

# Beginners Guide to Programming “Lesson 2”

By Michael Simpson

It's important that you read and understand last month's Lesson 1 before you proceed with this month's lesson. We are going to cover the For / Next and If / Then commands. With these two commands you have the ability to create very complex and intelligent constructs.

## For / Next Command

The For/Next loop allows you to create very efficient and simple loops. The syntax is very easy to follow, as you tell the command to do exactly what you want. To keep track of the number of loops, you supply a variable that you have previously defined in your program.

Let's take a closer look at the syntax of the For/Next command.

```
For VVV = SSS to EEEE
```

VVV = Variable to use as a counting index

SSS = Starting value for the loop

EEE = Ending value for the loop

```
Next
```

All the code between the For and Next statements will be executed during each cycle of the loop. The example shown in Figure 1 will count from 1 to 5, printing the index variable once for each repetition. Go ahead and run the simulator and step through each loop. Use the variable watch form that we went over last month to watch the idx variable.



**Figure 1**

The For/Next command is always checked before the actual loop is executed. This means if the index is out of range it will not execute even on the first pass. This is not true for all dialects of the Basic Language.

You can use the Step modifier to change the behavior of the For/Next loop. This will allow you to modify the amount the variable is advanced during each loop.

The syntax looks like this:

```
For VVV = SSS to EEEE Step AAA
```

VVV = Variable to use as a counting index

SSS = Starting value for the loop

EEE = Ending value for the loop

AAA = The amount to increment the variable

```
Next
```

Figure 2 shows the Step modifier in action. Instead of incrementing the variable by the default value of 1, it will now be incremented by 2.



**Figure 2**

As shown in Figure 3, the Step modifier can also be used to tell the For/Next command to count backwards. We need to do a few things differently to count backwards. First, we need to change the order of the start and end values. We also need to use a negative value for the step value. When counting backwards you must always use the Step modifier.

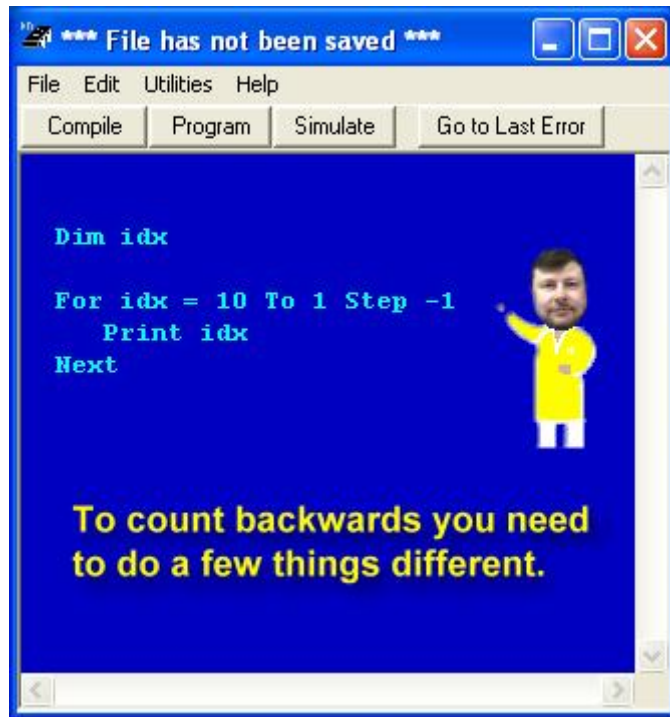


Figure 3

You may use variables in the place of the start and end index values as shown in Figure 4. This allows you to make some on-the-fly changes to your loop conditions while your program is running.

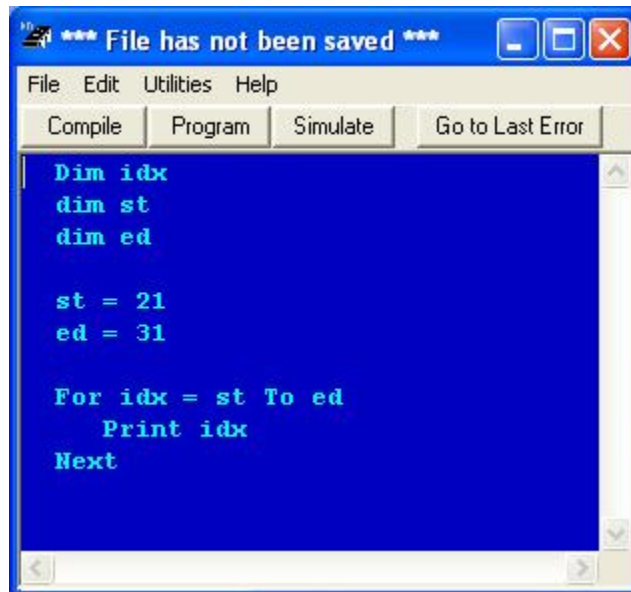


Figure 4

You may change the value of the index variable while the loop is executing. The condition of the index variable is checked once when the For/Next command is started and once each time the Next statement is encountered. If you change the value of the index variable it will be evaluated when the Next statement is reached and if it has reached the value of the end index value it will not loop. Run the program in Figure 5 using the simulator and watch the program as you single step through your code.

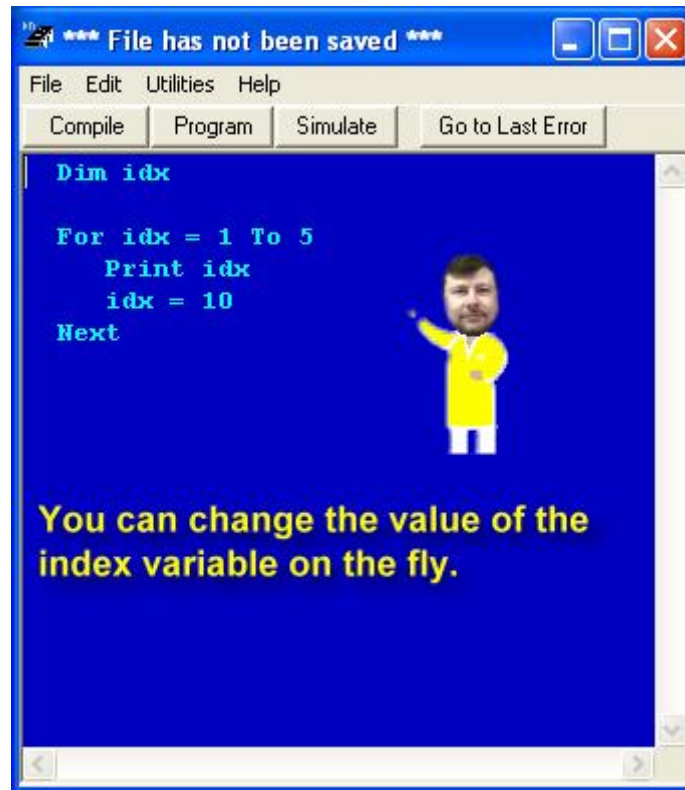


Figure 5

If you use variables for your index values, you may also change the end index on the fly as well, as it will be evaluated each time the Next statement is encountered.

### **If / Then / Else Command**

The If / Then command allows us to do real programming. Your program can start to make decisions based on various conditions while your program is running.

The basic syntax for the If / Then command looks like this:

If EXPRESION Operator EXPRESION then

EXPRESSION = Variable or math expression to evaluate  
Operator = The kind of evaluation to perform.

Endif

There are six valid operators that may be used to compare the two expressions in the If / Then command. They are as follows:

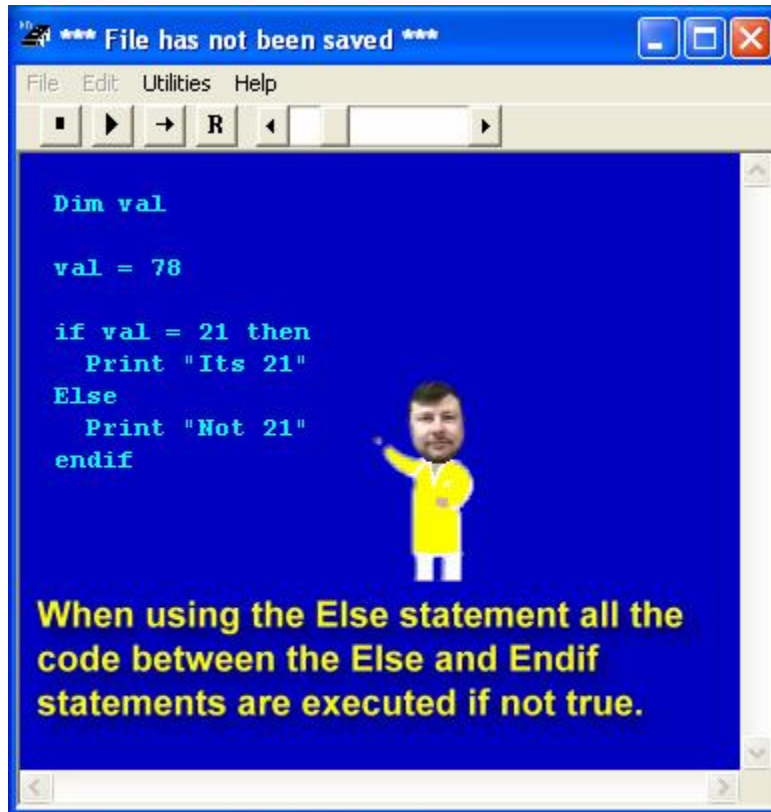
- = Equal
- < Less than
- > Greater than
- < Less than
- <> Not Equal to
- >= Greater than Equal to
- <= Less than Equal to

If the evaluation of the two expressions is deemed true, all the code between the Then and Endif statements will be executed as shown in Figure 6.



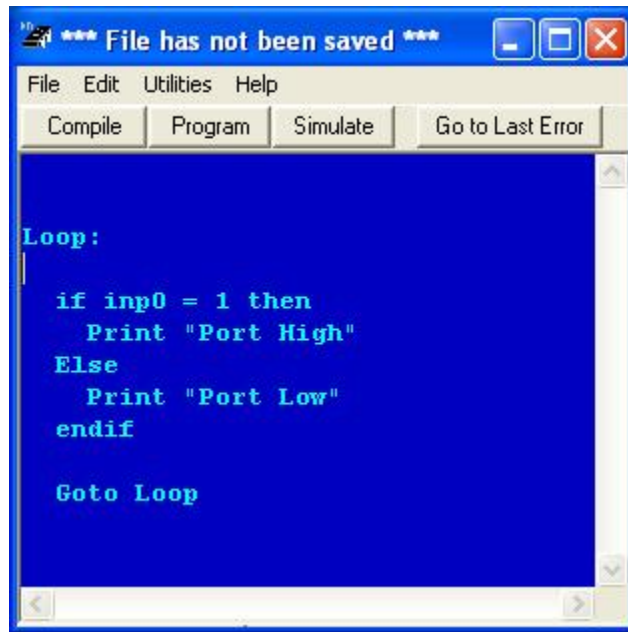
Figure 6

You can use the optional Else statement to execute a block of code if the evaluation of the expressions is false. Looking at the example in Figure 7, the program will print "Its 21" or "Not 21" depending on the value of the val variable.



**Figure 7**

Many times we need to execute code based on the condition of an IO port. Simulate the program shown in Figure 8. Double click the P0 field on the simulator form to change the input value of IO port 0. Notice that as you change the port, the program executes a different program block of code.

A screenshot of a software development environment window. The title bar reads "File has not been saved" with three asterisks on either side. The menu bar includes "File", "Edit", "Utilities", and "Help". Below the menu bar are four buttons: "Compile", "Program", "Simulate", and "Go to Last Error". The main text area has a blue background and contains the following code:

```
Loop:
    if inp0 = 1 then
        Print "Port High"
    Else
        Print "Port Low"
    endif

    Goto Loop
```

**Figure 8**

On many microcontrollers the actual syntax for the If / Then statements may vary slightly from chip to chip. For instance some microcontrollers don't allow you to actually execute blocks of code. You must branch to a subroutine instead.

## **Delay Commands**

There are three delay commands that may be used with the Athena compiler. These three commands are used to create the various delays your program may need.

### **Pause**

The Pause command allows you to create a delay of 1 to 255 milliseconds. The example in Figure 9 demonstrates a 100ms delay when blinking an LED connected to port 0. Simulate the program and watch the simulator form.



**Figure 9**

## **Pauseus**

The Pauseus command allows you to create a delay of 3 to 765 microseconds. The value you supply is 1 to 255, but each unit is 3 microseconds in length. A value of 100 would create a 300 microsecond delay. The example in Figure 10 creates a 300 microsecond pulse every 20 milliseconds.

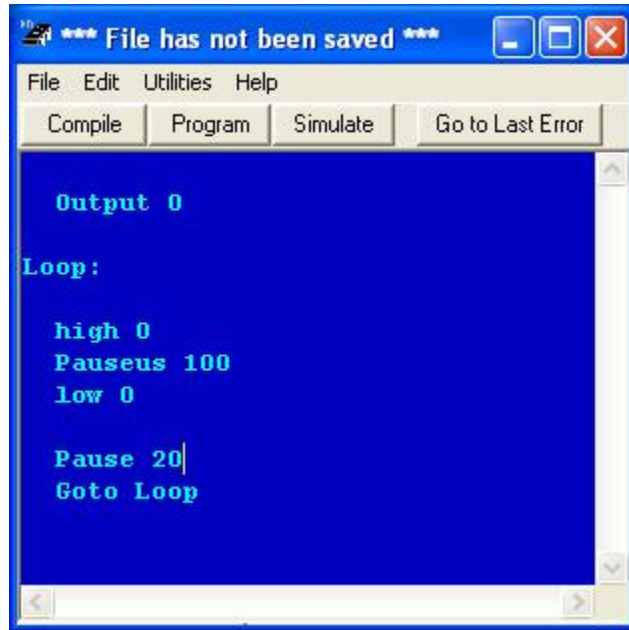


Figure 10

### LongPause

Because the Athena is an 8-bit microcontroller it only deals with values between 0 and 255. In order to perform longer delays, the LongPause command is used. We could use the Pause command 4 times to create a one second delay as shown in Figure 11. But by using the LongPause command, you save code space and combine the 4 Pause commands into a single operation as shown in Figure 12.

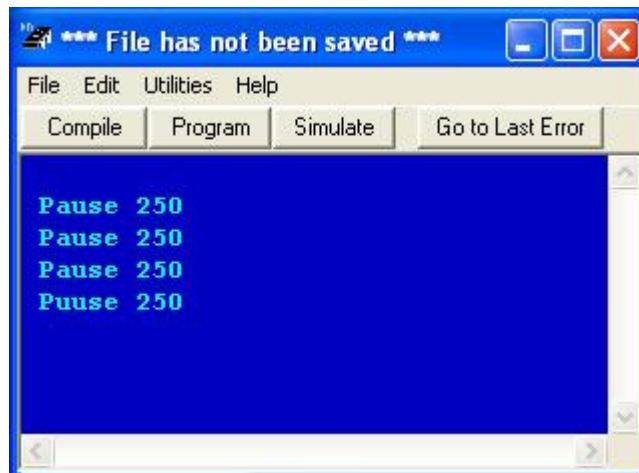
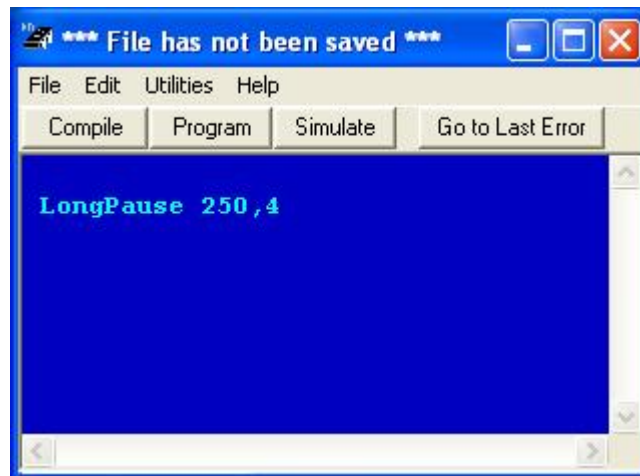


Figure 11



**Figure 12**

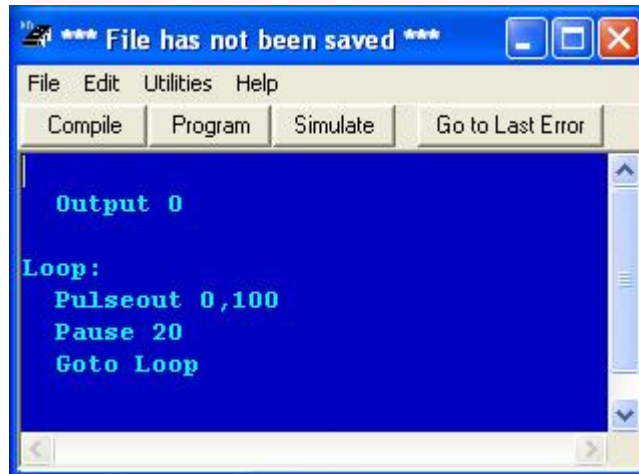
As you interface to various devices, you will find the delay commands invaluable. Many devices need predetermined delays between commands and singles in order to properly interface with them.

## **Pulses**

You saw how we could use the Pauseus command to create short pulses. To make things easier, there are commands that will allow you to create a pulse on an IO port using a single command.

### **Pulseout**

The Pulseout command allows you to create a very short duration pulse on an IO port. You call the command passing both the port and delay in 3us resolution. The command takes into account the current state of the IO port and pulses it for the given delay, then returns it to its original state. The example in Figure 13 creates a 300 microsecond pulse every 20 milliseconds.

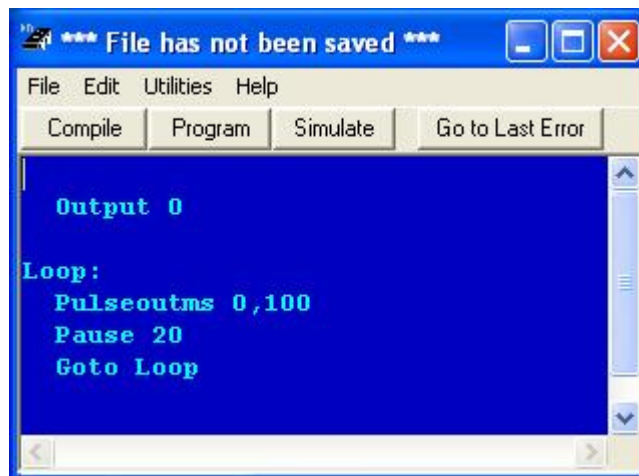


```
*** File has not been saved ***
File Edit Utilities Help
Compile Program Simulate Go to Last Error
Output 0
Loop:
Pulseout 0,100
Pause 20
Goto Loop
```

**Figure 13**

### **Pulseoutms**

The Pulseoutms command works in the same way as the Pulseout command but at a much slower rate. The example in Figure 14 creates a much slower pulse every 20 milliseconds.



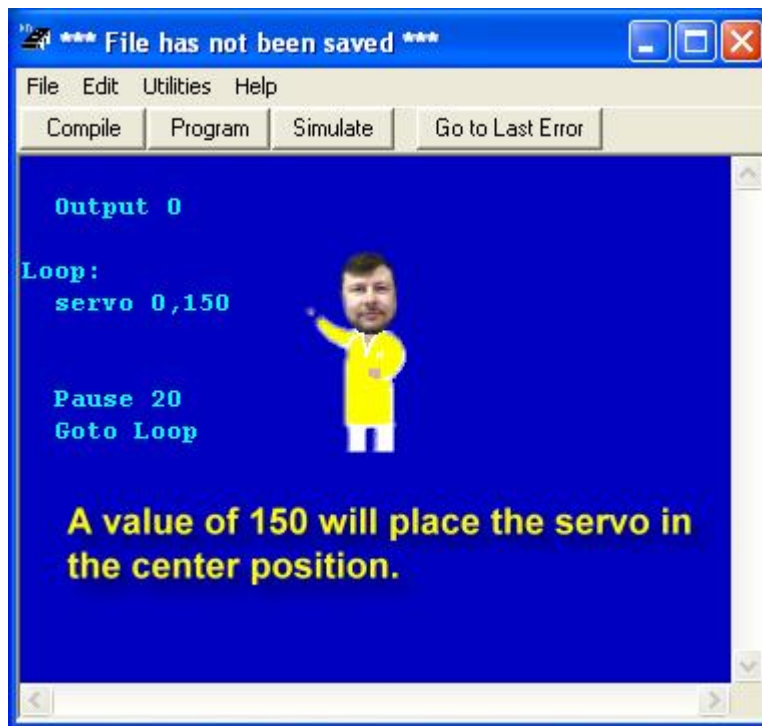
```
*** File has not been saved ***
File Edit Utilities Help
Compile Program Simulate Go to Last Error
Output 0
Loop:
Pulseoutms 0,100
Pause 20
Goto Loop
```

**Figure 14**

One example of using a pulse is when telling a standard servo to move to a particular position. A servo works by detecting a 1000 to 2000 microsecond pulse. If the pulse is at 1500 microseconds, the servo will be positioned at its center position. There must be a 20 millisecond delay between the pulses or the servo will not function properly. A problem arises because we don't have a pulse command that falls within the range needed for controlling a servo. To solve this problem, the Athena compiler has a Servo command.

## Servo

When using the Servo command you pass the port number and pulse value. The pulse value is in 10 microsecond units to make things easy to calculate. This gives the command a 10us to 2550us range, which is perfect for most servos. The example shown in Figure 15 shows how simple it is to use the servo command. You need to make sure you call the servo command at regular intervals of 20 milliseconds or so.



The screenshot shows a software window titled "File has not been saved" with a menu bar (File, Edit, Utilities, Help) and buttons for Compile, Program, Simulate, and Go to Last Error. The main area has a blue background with the following text:

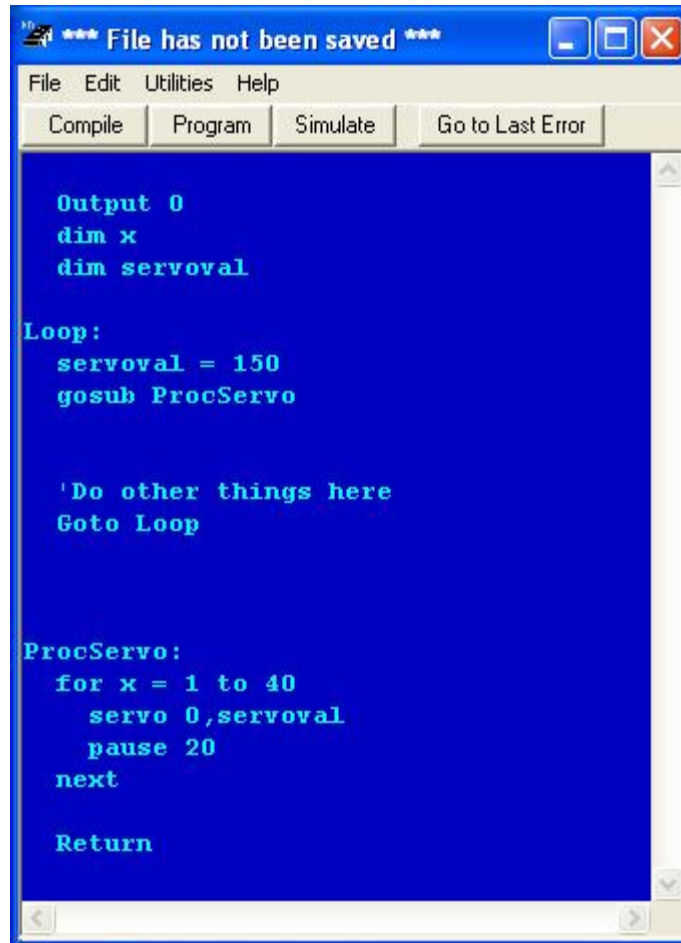
```
Output 0
Loop:
  servo 0,150

Pause 20
Goto Loop
```

A cartoon character in a yellow lab coat is pointing at the code. Below the code, the text reads: "A value of 150 will place the servo in the center position."

Figure 15

You can stop the pulses to a servo but you need to give the servo time to reach its destination before you stop the pulses. The program in Figure 16 does just that. You set the value in the servoval variable, then make a call to ProcServo routine using the gosub command.



```
Output 0
dim x
dim servoval

Loop:
servoval = 150
gosub ProcServo

'Do other things here
Goto Loop

ProcServo:
for x = 1 to 40
servo 0,servoval
pause 20
next

Return
```

**Figure 16**

## Summary

This month we covered the For / Next and If / Then commands in detail. We also looked at the various ways of creating delays and pulses on a microcontroller.

If you can answer the following questions, then you understand this month's lesson and can move on to the next lesson with confidence.

1. What are the three parameters needed to create a For / Next loop?
2. What modifier is needed to cause the For / Next command to count backwards?
3. Create a program that counts backwards from 100 to 10 at intervals of 10.

4. What command statement is used to cause a code block to be executed when an If / Endif command is false?
5. Create a program that counts backwards from 100 to 10 at intervals of 10. When the index is equal to 50, have the program print "Match", otherwise print the index.
6. What command is best used to create a 5 second delay? How would it be used?
7. Create a program that pulses IO ports 0,1,13 and 14 in sequence. Place these in a loop so that it runs continuously.

## **Answers**



**Dont cheat!  
Im watching you.**

1. Variable, Start, End

2. Step

3.

```
dim idx
for idx = 100 to 10 step -10
  print idx
next
```

4. Else

5.

```
dim idx
for idx = 100 to 10 step -10
  if idx = 50 then
    print "Match on 50"
  else
    print idx
  endif
next
```

6. LongPause, LongPause 250,20

7.

```
Output 0
Output 1
Output 13
Output 14
Loop:
Pulseouts 0,100
Pulseouts 1,100
Pulseouts 13,100
Pulseouts 14,100
goto Loop
```

Next month we will cover branching and bit manipulation commands. Remember, you can pick up a free copy of the Athena compiler/simulator at:

[www.kronosrobotics.com](http://www.kronosrobotics.com).

The program installs a complete manual and several sample programs.