

## Scripting and its Benefits for Meridian



The Meridian is a versatile wire and harness analyzer. This all-in-one bench top tester can be used for many applications including testing wires, cables, and harnesses for opens, shorts, miswires, and testing components within a cable or fixture assembly. Scripting language is a way for engineers to customize the tester to their application needs.

### What is scripting?

A script language is a programming language that allows control of one or more software applications. "Scripts" are different from the core code of the application and are often generated or can be modified by the end-user. Scripting languages are almost always embedded in the applications they control.

Languages such as TCL and Lua were specifically designed as general purpose scripting languages that could be embedded in any application or used on their own. The Meridian has a standard option of using TCL and Lua, and is designed as a platform for both languages to run on when programming a test. Both scripting languages can be tailored to meet the needs of Meridian end-users.

### TCL and Lua

TCL is one of the smallest core languages of all scripting languages; it basically consists of a general comment, command, block syntax and evaluation semantics, but has no actual language primitives. It is a very powerful - but easy to learn - dynamic programming language.

```
proc power {base p} {  
    set result 1  
    while {$p > 0} {  
        set result [expr $result * $base]  
        set p [expr $p - 1]  
    }  
    return $result  
}
```

Figure 1: TCL Example

Lua is also very powerful, fast, lightweight, and embeddable. It combines simple procedural syntax with powerful data description constructs based on associative arrays and extensible semantics.

```

> a,b = 0,1
> while true do                                -- infinite loop
>> io.write(b, ", ")
>> a,b = b,a+b
>> if a>500 then break end                    -- exit the loop if the condition is true
>> end
1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, >

```

Figure 2: Lua Example

## Benefits

This being said, what are the benefits of using scripting languages with Meridian Wire and Harness Analyzer?



Figure 3: Meridian Test Program with Script Example 1

### Benefit 1: Customization

Every tester has rules on how a test sequence runs. When these rules don't meet an end-user's needs, scripting language can allow them to write their own tests that meets their specific requirements. Therefore, the end-user decides what will happen during test, and the tester understands how the end-user would like to test his products. Customization is a very powerful feature of the scripting, especially when running tests on products that have special needs.

For example, the normal sequence in a test is to run the low voltage test first, and the high voltage test after. However, if a customer has special needs to test high voltage first and the low voltage next, scripting could help operators achieve this goal.

Another example is if the customer would like to test a parallel diode with different voltage and current setting, scripting could help him program to the precise different voltage range. This customization helps improve the quality of testing results.

### Benefit 2: Control

Scripting gives the end-user more control over the entire testing process. When operators are plagued by interruptions due to abnormal testing requirements that aren't scripted, it disrupts the flow of production. Scripting allows end-users to efficiently plan and program their test to minimize errors and streamline the testing process, no matter how simple or complex it is.

### Benefit 3: Clear to set up easy

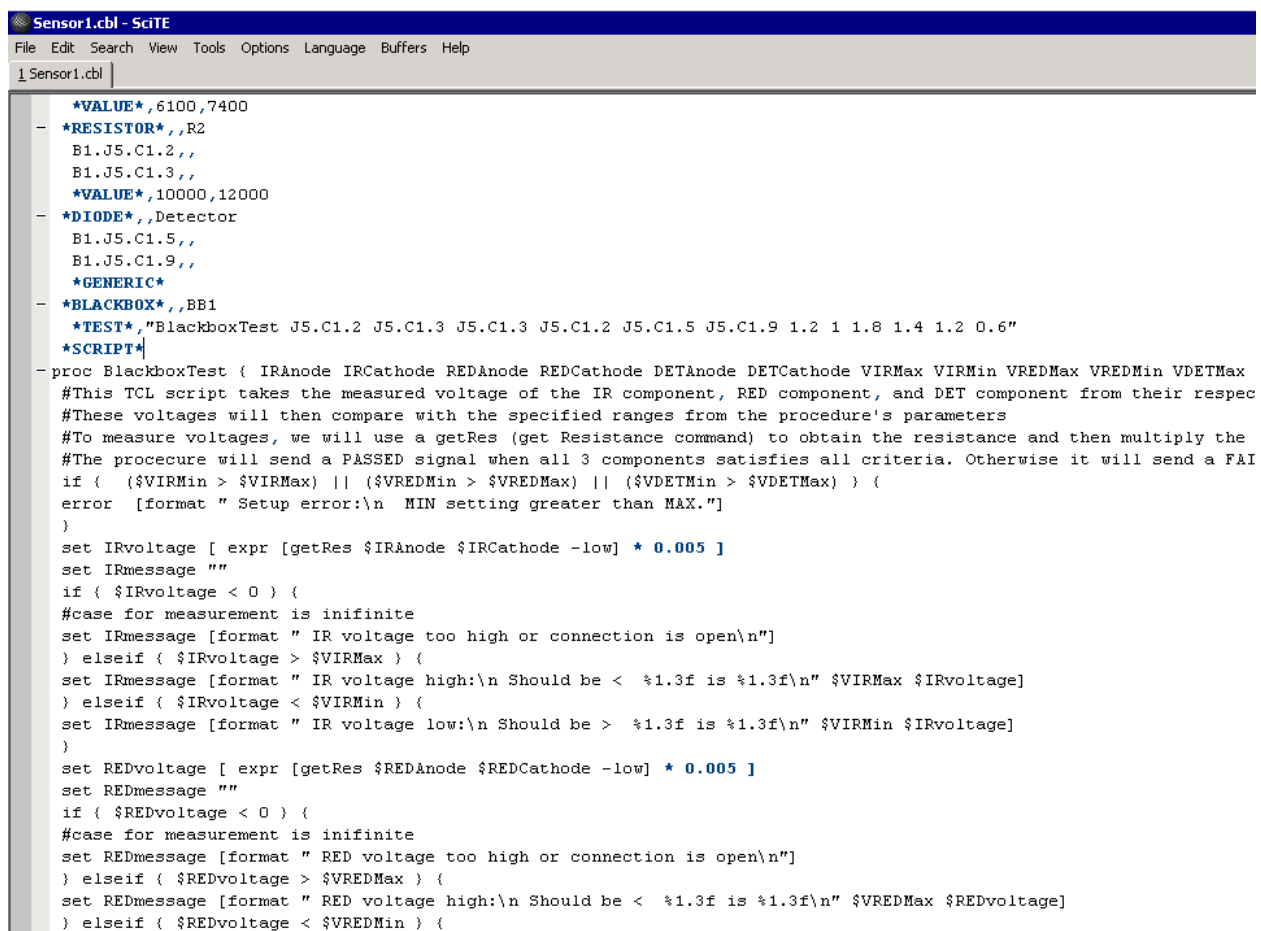
Even if the test is complex, setting it up doesn't have to be. Scripting makes a difficult job easier.

The Meridian is capable of testing a wide variety of products such as military connectors, cables with shield, embedded components, and other complex products. When test engineers prepare a program for such products, it often takes them many hours. With the help of scripting, test engineers can re-use the code to make such test programs simple.

### Benefit 4: Common

Both TCL and Lua are very common scripting languages for test engineers. They are widely used as industry standards for many industries.

Test engineers who work with the Meridian will have the ability to write a script with minimum to no training required, and the knowledge is transferable.



```

Sensor1.cbl - SciTE
File Edit Search View Tools Options Language Buffers Help
1_Sensor1.cbl
*VALUE*,6100,7400
- *RESISTOR*,R2
  B1.J5.C1.2,,
  B1.J5.C1.3,,
  *VALUE*,10000,12000
- *DIODE*,Detector
  B1.J5.C1.5,,
  B1.J5.C1.9,,
  *GENERIC*
- *BLACKBOX*,BB1
  *TEST*"BlackboxTest J5.C1.2 J5.C1.3 J5.C1.3 J5.C1.2 J5.C1.5 J5.C1.9 1.2 1 1.8 1.4 1.2 0.6"
  *SCRIPT*
- proc BlackboxTest ( IRNode IRCathode REDNode REDCathode DETNode DETCathode VIRMax VIRMin VREDMax VREDMin VDETMax
  #This TCL script takes the measured voltage of the IR component, RED component, and DET component from their respec
  #These voltages will then compare with the specified ranges from the procedure's parameters
  #To measure voltages, we will use a getRes (get Resistance command) to obtain the resistance and then multiply the
  #The procedure will send a PASSED signal when all 3 components satisfies all criteria. Otherwise it will send a FAI
  if { ($VIRMin > $VIRMax) || ($VREDMin > $VREDMax) || ($VDETMin > $VDETMax) } {
    error [format " Setup error:\n MIN setting greater than MAX." ]
  }
  set IRvoltage [ expr [getRes $IRNode $IRCathode -low] * 0.005 ]
  set IRmessage ""
  if { $IRvoltage < 0 } {
    #case for measurement is infinite
    set IRmessage [format " IR voltage too high or connection is open\n"]
  } elseif { $IRvoltage > $VIRMax } {
    set IRmessage [format " IR voltage high:\n Should be < %1.3f is %1.3f\n" $VIRMax $IRvoltage]
  } elseif { $IRvoltage < $VIRMin } {
    set IRmessage [format " IR voltage low:\n Should be > %1.3f is %1.3f\n" $VIRMin $IRvoltage]
  }
  set REDvoltage [ expr [getRes $REDNode $REDCathode -low] * 0.005 ]
  set REDmessage ""
  if { $REDvoltage < 0 } {
    #case for measurement is infinite
    set REDmessage [format " RED voltage too high or connection is open\n"]
  } elseif { $REDvoltage > $VREDMax } {
    set REDmessage [format " RED voltage high:\n Should be < %1.3f is %1.3f\n" $VREDMax $REDvoltage]
  } elseif { $REDvoltage < $VREDMin } {

```

Figure 4: Meridian Test Program with Script Example 2