Communicating with SLICE Products using the Serial Command API in a Python Script

Vescent

May 9, 2022

1 Introduction

One of the most compelling features of the SLICE product line is its ability to interface easily with computers using the SLICE serial command API. These commands, found on the web pages for each respective SLICE product, are typically sent through an SSH program such as Tera Term or PuTTY. However, it's also possible to write simple python scripts to automate SLICE control. This application note provides the framework for creating custom python scripts to easily send and receive data from SLICE.

2 Code

The provided code contains the `slice_send` function, which will allow communication with SLICE products. It takes a COM port, and a serial command from the SLICE API as inputs, then returns the output of the command. To create your own script, simply copy and paste the code below. You may need to install pyserial on your machine for the code to compile, since pyserial is not included with the default pyhton installation. An example of how to use the code is provided after the function definitions.

Download:
slice_send.py

```
import time
import serial


def slice_send(ComPortNum, CommandInput, bDisplay = True):

    '''
    Function that sends a serial string command to SLICE Box:
    ---------------------------------------------------------

    Input: Com Port Number[int], CommandInput[str]
    Output: Response from SLICE. converted to all caps, and leading and trailing
    whitespaces removed.

    Notes:
    -Read buffer always emptied!
    -COM Port is opened/closed each time function is run

    '''

    lastComTime = 0

    # generate the inter-communication delay:
    # In some instances there is not enough delay between communication which
    # may cause test script functonality issues. This section of the SendSlice module
    # will add delay if needed.
    minInterComDuration = 0.25   # this is the minimum time between calls to this
    function before additional sleep delay is added.
    interComDuration = time.time() - lastComTime
    reqdDelay = max(0, minInterComDuration-interComDuration)
    time.sleep(reqdDelay)
```

```
30
31
32      #Open Port w/ SLICE COM Default Settings
33      SliceOutputReturn = 'EMPTY!'
34      SliceSer = serial.Serial(port='COM'+str(ComPortNum), baudrate=115200, timeout=0.3,
        bytesize=8, write_timeout = 0)
35
36
37      #Define Command
38      MasterCommand = str(str(CommandInput) + '\r' )
39
40      #Send Commands/Close Port
41      SliceSer.write(MasterCommand.encode())
42
43      i=0
44      SliceOutputReturn = SliceSer.read(256) #Read Buffer, 256 is the Number of bytes to
        read on ser.read()
45      while len(SliceOutputReturn)==0 and i<20:
46          SliceOutputReturn = SliceSer.read(256) #Read Buffer, 256 is the Number of bytes
        to read on ser.read()
47          time.sleep(0.20)    #wait(SliceDelay)
48          i=i+1
49
50      SliceOutputReturn = SliceOutputReturn.upper().decode()
51      SliceOutputReturn = SliceOutputReturn.rstrip(" \r\n") #strip off the <CR> and <LF>
        and any spaces
52      SliceOutputReturn = SliceOutputReturn.lstrip()
53
54      #Return Output
55      if(bDisplay):
56          print(SliceOutputReturn)
57
58      #Close COM Port
59      SliceSer.close()
60      lastComTime = time.time()
61
62      return SliceOutputReturn
63
64
65
66  #Example:
67  COM_Port = 4
68  slice_send(COM_Port, '#VERSION') #Returns the System Controller's Firmware Version
69  slice_send(COM_Port, 'CONTROL? 1') #On a SLICE-DCC, returns the status of Channel 1
```

Listing 1: SLICE Python/Serial Command API interface functions

For more information, contact Vescent at:+1 (303) 296-6766, info@vescent.com, or visit our website at https://www.vescent.com