应用文档

SIGLENT 鼎阳

编程示例

使用 Kotlin 从 XE 系列示波器中检索数据

Copyright © 2019 Siglent Technologies, All Rights Reserved

访问 www.siglent.com 获得更多鼎阳仪器信息

联系鼎阳： market@siglent.com 或 拨打免费电话 400-878-0807

SDS 系列示波器均具有远程编程和数据采集功能。 它们可以轻松集成到许多自动测试环境中，以简化测试期间的设置和数据采集。

我们的一位有用的客户开发了一个很好的编程示例，旨在使用 Kotlin 设置和检索来自 SIGLENT SDS1202X-E 示波器的数据，Kotlin 是一个免费的开源编码环境（此处更多关于 Kotlin）。

该代码使用 LAN 连接和打开的套接字。

感谢 Chris Welty 的代码！

代码如下：

```
**
* License: 3-Clause BSD
*
* Copyright 2018 Chris Welty
*
* Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:
* 1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
* 2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
* 3. Neither the name of the copyright holder nor the names of its contributors may be used to endorse or promote products derived from this software without
```

specific prior written permission.

*

* THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS “AS IS” AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING,

* BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO

* EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR

* CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR

* PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,

* OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE

* POSSIBILITY OF SUCH DAMAGE.

*/

```
package scope

import java.io.BufferedWriter

import java.io.OutputStreamWriter

import java.io.Serializable

import java.net.Socket

/**
```

```
* 创建一个从 Siglent 1202X-E 下载的波形
*/
class Waveform(val vDiv: Double, val vOffset: Double, val tDiv: Double, val tOffset:
Double, val data: ByteArray) : Serializable {
    val xs: DoubleArray
get() = DoubleArray(data.size, { i -> i * tDiv * 14 / data.size + tOffset – tDiv * 7 })
    val ys: DoubleArray
get() = DoubleArray(data.size, { i -> data[i] * vDiv / 25 – vOffset })
    companion object {
/**
* 下载在示波器屏幕上显示的波形
*/
fun download(): Waveform {
Socket("192.168.1.222", 5025).use { socket ->
    println("connected to " + socket.inetAddress)
val output = BufferedWriter(OutputStreamWriter(socket.getOutputStream(),
Charsets.US_ASCII))
//由于套接字可以返回二进制数据，我们不能使用 InputStreamReader
//将字节翻译成字符。SCPI 通常使用 US ASCII。
val input = socket.getInputStream()
/**
*开始读取，直到遇到\ n。
```

```
*字节数字转换为数字（ASCII）。
*/
fun readLine(): String {
val sb = StringBuilder()
while (true) {
val c = input.read()
when (c) {
-1, '\n'.toInt() -> return sb.toString()
else -> sb.append(c.toChar())
}
}
}
    /**
* 读取字节
* 字节不会转换为字符
*/
fun readBytes(n: Int): ByteArray {
val result = ByteArray(n)
var i = 0
while (i < n) {
i += input.read(result, i, n – i)
}
```

```
return result
}
    fun writeLine(string: String) {
output.write(string)
output.write("\n")
output.flush()
}
/**
* 读取数据响应
* 比如 "C1:VDIV 1.00E+00V".
* 此函数提取"1.00E + 00"，将其转换为 double 类型，然后返回。
*/
fun readNumber() = readLine().split(" ")[1].dropLast(1).toDouble()
    writeLine("*IDN?")
println(readLine())
    //将响应格式重置为默认值，以便 readNumber（ ）运行
writeLine("CHDR SHORT")
    writeLine("C1:VDIV?")
val vDiv = readNumber()
    writeLine("C1:OFST?")
val vOffset = readNumber()
    writeLine("TDIV?")
```

```
val tDiv = readNumber()
    writeLine("TRDL?")
val tOffset = readNumber()
    // 请求所有的波形点
writeLine("WFSU SP,0,NP,0,F,0")
writeLine("C1:WF? DAT2")
    // 解析波形响应
val header = String(readBytes(21))
println("header is $header")
val length = header.substring(13, 21).toInt()
println("length is $length")
val data = readBytes(length)
readBytes(2)
//最后两个无效字节
    println("V/div = $vDiv; offset = $vOffset; t/div = $tDiv; tOffset = $tOffset")
    return Waveform(vDiv, vOffset, tDiv, tOffset, data)
}
}
}
}
```