

DRWavePar 波形分析用法

DRWavePar 波形分析控件提供一个计算信号波形参数的模块，可以对收到的采样信号进行波形参数计算。该控件为后台控件，在 GUI 界面上无显示。

在 Python 程序 Tkinter GUI 界面中创建一个 DRWavePar 波形分析控件的命令和语法格式如下：

```
hl=DRWavePar(option=value, ...)
```

该控件创建时只有一个可选的输入参数：

N：分析信号长度，默认值=0，即按收到的信号长度

返回值 hl 则是该控件创建后的句柄变量名，可以通过 hl 对创建后的控件进行进一步的设定。

DRWavePar 波形分析控件有 2 个属性函数，其功能定义如下：

hl.setValue2DX(dx,yarray)：设定信号的采样间隔和采样数组

par=hl.getArrayWavePar()：取返回的波形参数数组，排放顺序为：频率、周期、双峰值、有效值、方差、均值、翘度。

下面是一个全参数的 DRWavePar 波形分析控件创建实例：

```
hl=dsp.DRWavePar(0)
```

通常，可以选择设定部分选择参数的形式进行设定，其余可选参数采用默认值。创建实例如下：

```
hl=dsp.DRWavePar()
```

创建后可以通过返回的控件句柄对其属性函数进行操作。

DRWavePar 波形分析控件可以通过绑定的方式将信号的波形参数传递给回调函数或其他控件。获取识别的信号频率参数的绑定方法为：

```
hl.addCallBackVSFre(fun)
```

获取识别的信号周期参数的绑定方法为：

```
hl.addCallBackVST(fun)
```

获取识别的信号双峰值参数的绑定方法为：

```
hl.addCallBackVSPP(fun)
```

获取识别的信号有效值参数的绑定方法为：

```
hl.addCallBackVRMS(fun)
```

获取识别的信号方差参数的绑定方法为：

```
hl.addCallBackVSTd(fun)
```

获取识别的信号均值参数的绑定方法为：

```
hl.addCallBackVSMean(fun)
```

获取识别的信号翘度参数的绑定方法为：

```
hl.addCallBackVSKurtosis(fun)
```

被绑定的回调函数或另一控件的 set 函数是一个单值函数，收到的是指定的信号波形参数。回调函数形式为：

```
def setValueSingle(v):  
  
    ...
```

当采用 DRPython 进行图形化编程时，控件的创建代码和直连绑定代码的生成均可以由机器实现；控件创建只需要将所需控件拖拽到 GUI 布局区，而直连绑定则需要在 GUI 布局区的控件间进行连线操作。DRWavePar 波形分析控件可以做为字符串或单数值输出控件，在连线状态下在控件上点击鼠标，则弹出端选择；选择完成后，点击另一控件，则弹出输入端选择；若输出端和输入端类型一致，则完成连线。输出参数为波形分析结果，包括单值的识别出的信号频率(addCallBackVSFreAmp)，单值的识别出的信号周期(addCallBackVST)，单值的识别出的信号双峰值(addCallBackVSPP)，单值的识别出的信号有效值(addCallBackVSRMS)，单值的识别出的信号标准差(addCallBackVSFreStd)，单值的识别出的信号均值(addCallBackVSMean)，单值的识别出的信号翘度(addCallBackVSKurtosis)。



DRWavePar 波形分析控件也可以做为单数/字符串输入控件，在连线状态下当上一控件已选输出函数后，在控件上点击鼠标，则弹出连线选择菜单；选择完成后，若输出端和输入端类型一致，则完成连线。输入参数为单值+一维数组的采样信号(addValue2DX)。



下面是 DRWavePar 波形分析控件的创建和应用示例小程序，功能是在 GUI 界面上插入 1 个按钮控件，1 个信号发生器控件、1 个 DRWavePar 波形分析控件、一个用户自定义回调函数和 1 个 Plot 曲线控件，然后分别通过直连绑定的方式，用按钮控件控制信号发生器的启/停、显示信号波形曲线和 print 识别的波形参数：

```
===引用的包  
import tkinter as tk
```

```

import numpy as np
import drvi.drviControlls as dr
import drvi.drviDSP as dsp

def muserSingle(v):
    global F
    Fs=5000
    N=512
    F=F+10
    t=np.arange(N)*(1/Fs)
    y=0.5*np.sin(2*np.pi*F*t)
    mPlot.setValue2D(t,y)
    mWavePar3.setValue2DX(1/Fs,y)
    par=mWavePar3.getArrayWavePar()
    print(par)

#==创建 Tkinter GUI 窗口
win= tk.Tk()
win.geometry('900x500')
win.config(bg="#ddeeef")
#==插入按钮控件
mButton1=dr.DRButton(win, 56, 14, 100, 40)
#==插入波形分析控件
mWavePar3=dsp.DRWavePar()
#==插入曲线控件
mPlot=dr.DRPlot(win, 58, 91, 700, 210)
#=直连绑定
mButton1.addCallBackSingle(muserSingle)
#==全局变量==
F=0
#==主程序循环
win.mainloop()

```

pyqt 版本中的用法说明:

DRVI 有继承自 Tkinter 和继承自 pyqt 两个不同库,除引用的 GUI 库(Tkinter/pyqt)和框架程序略有不同外,DRVI 部分控件的创建、布局和连线是完全相同的。因均继承自 Numpy 和 Scipy,该函数在两个版本中用法无区别。