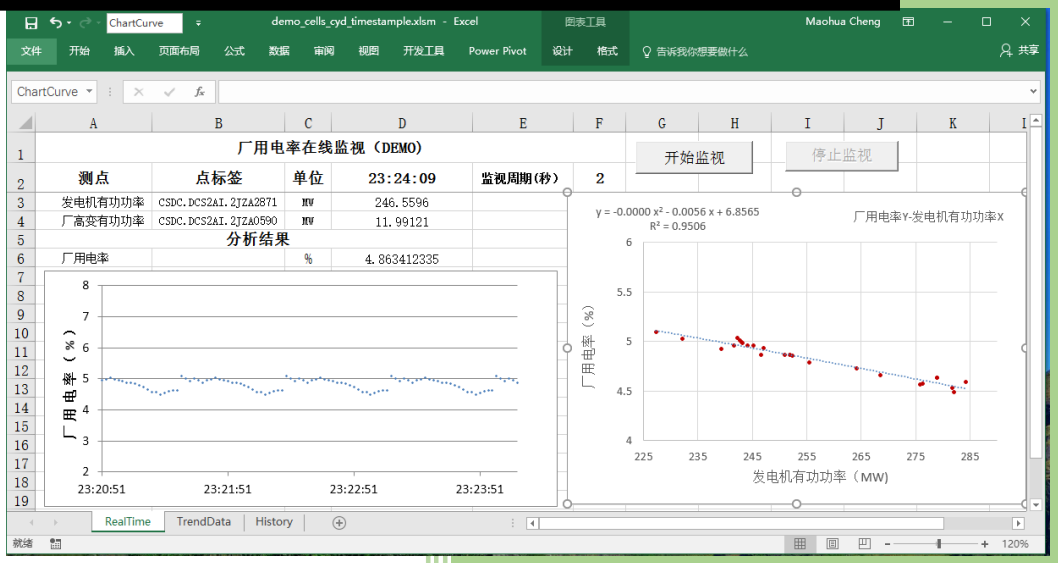
2017

工业过程在线监视 MS Excel 工作簿实例



程懋华

东南大学

2017-1-8

目录

一、 概述

MS Excel 是最常见电子表格软件。Excel VBA 使其成为一个最随手可及的软件开发环境。日常工作中使用 Excel VBA 开发软件，对提高工作效率和质量有很大的帮助，是工程师必备基本技能。

工业过程监视是工程师常见工作之一。通常使用监控软件系统，如各种 SCADA 等。这些系统提供了丰富功能，但是难以满足现场工程师的“即时”需求，这时，随手可及的 Excel 就是简单、快捷地实现监视工作的利器。

本文档以厂用电率在线监视为例，给出在线监视 Excel 软件的可定制标准模板，详细说明可定制模板各部分的代码及其技术。

示例模板设计原则是：技术简单、结构清晰，模块通用且不过度设计；功能完备、使用方便，成为可定制在线监视 Excel 软件标准模板。技术人员可以这个可定制标准模板为基础按需实现自己的在线监视软件。

1.1 运行环境

运行环境由：操作系统、办公软件、Python 及其软件包、Redis 数据库等组成。

操作系统：Windows10 64 位；

办公软件：Microsoft Excel2016 64 位；

Python 语言：Python3.5 64 位；

Python 软件包：

Pyredis:和 Redis 交互数据；

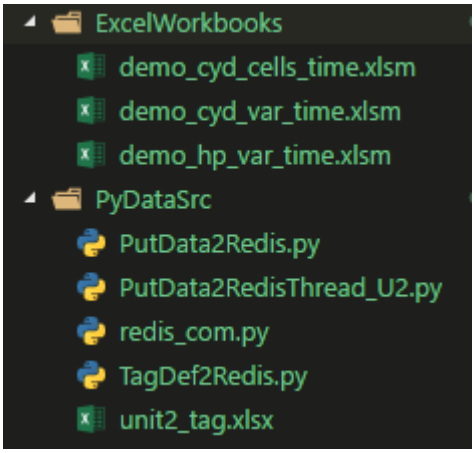
PyWin：Python extensions for Windows 提供 COM 支持；

Xlrd：读写 Excel 文件；

Redis：内存数据库，用于为 Excel 提供数据服务。

1.2 监视软件

监视示例软件由 2 类文件组成,分别存放在各章文件夹中：1）运行数据仿真-PyDataSrc；2）监视 Excel 工作簿:ExcelWorkbooks。



示例系统启动由 3 个步骤构成：

（一）初始化运行数据服务

1） 注册 Redis 数据交互 COM 组件：redis_com.py,只需注册一次；

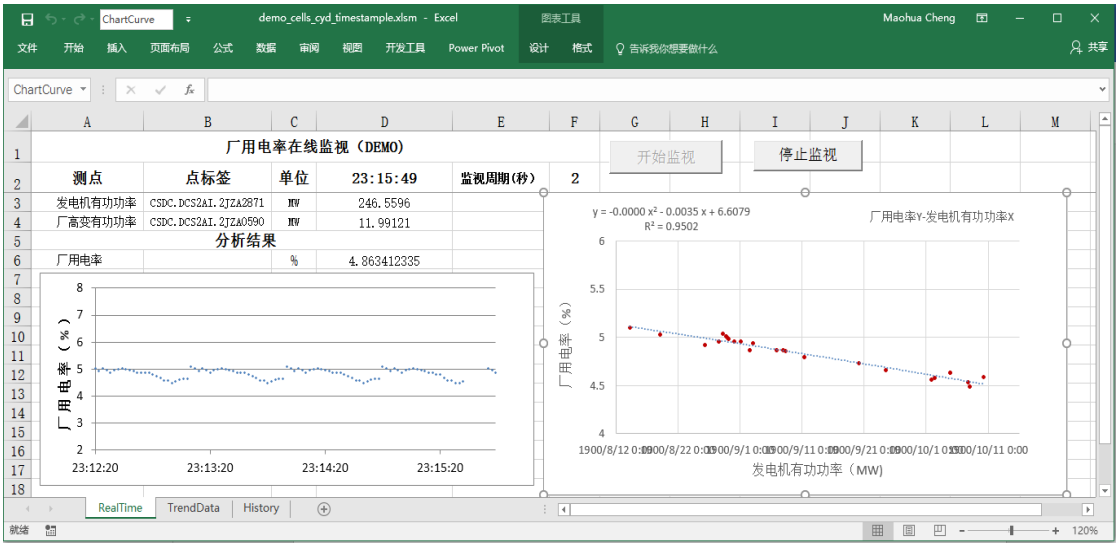
2） 在 Redis 中建立点表：TagDef2Redis.py,Redis 默认安装提供持久化，只需运行一次；

（二）运行数据仿真过程：

发送运行数据到 Redis： PutData2RedisThread_U2.py,从 unit2_tag.xlsx 提取数，发送到 Redis，仿真运行数据；

（三）在线监视：

启动在线监视工作簿：demo_cyd_cells_time.py/ demo_cyd_var_time.py



下面用三个小节分别说明各部分的软件技术。

二、 Redis 数据服务

在线监视模板示例用开源 Redis 数据库提供数据服务，构建实时运行数据源，以 MS Excel 为分析中心。模板的数据源和不同真实工作场景下的数据源无关，更具普遍性。

2.1 Python 环境

Python ： 从 Python 官网 https://www.python.org/downloads/windows/ 下载 Python3.5.2 安装。

Python 软件包：

1)Pywin32(Python for Windows (pywin32) Extensions)：

```
https://github.com/mhammond/pywin32
pip install pywin32
```

Note that if you want to use pywin32 for "system wide" features, such as registering COM objects or implementing Windows Services, then you must run the following command from an elevated command prompt:

```
python Scripts/pywin32_postinstall.py -install
```

2)Xlrd：pip install xlrd

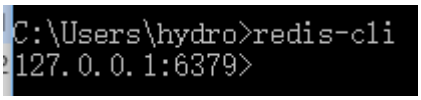
3)Pyredis: pip install redis

2.2 Redis 数据库

Redis 官方不提供 Windows 版本。但微软技术开源组开发维护了 Windows 版本的 Redis。可以从：

https://github.com/MSOpenTech/redis/releases

下载 msi 安装包安装。安装后在 DOS 命令行下，执行：



说明安装成功。

2.3 运行数据仿真

运行数据仿真程序由一个数据文件和 3 个 Python 程序构成。

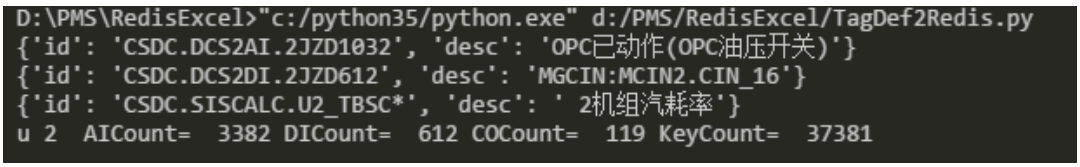
1） unit2_tag.xlsx：Excel 工作簿中有运行数据仿真的点信息和运行数据；示例采用中电投常熟电厂 2 号机组测点和数据文件

2） TagDef2Redis.py：从 unit2_tag.xlsx 读取点信息，在 Redis 中建立点的 Hash 键

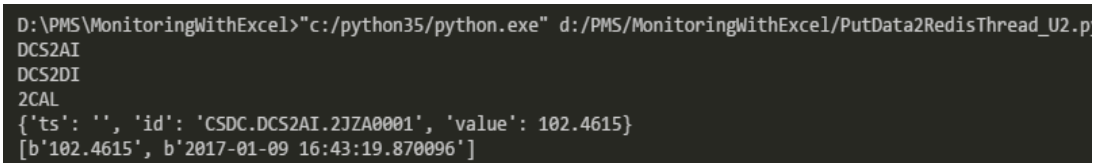
3） PutData2RedisThread_U2.py: 引用 PutData2Redis.py，从 unit2_tag.xlsx 读取点信息和运行数据向 Redis 发送仿真运行数据。

启动数据仿真：

1）运行：TagDef2Redis.py 在 Redis 中建立点表



2）运行： PutData2RedisThread_U2.py 向 Redis 发送数据



2.4 Redis 数据服务

2.4.1 数据 COM 组件

Excel 是基于 COM 组件模型的软件，使用 COM 组件和 Redis 交互数据很方便。基于 PyWin 用 Python 编写 COM 组件 redis_com.py 服务于和 Redis 的数据交互。

示例组件中的 get_snapshot 方法，用测点标签从取 Redis 中取数据。

```
import pythoncom
import redis
```

```
try:
    conn = redis.Redis('localhost')
except:
    pass

class RedisRealtime:
    _public_methods_ = ["get_snapshot"]
    _reg_progid_ = "Redis.Snapshot"
    _reg_clsid_ =  pythoncom.CreateGuid()

    def get_snapshot(self, tagid):
        htag=conn.hmget(tagid, 'value', 'ts')
        return float(htag[0].decode())

if __name__ == "__main__":
    print("Registering COM server...")
    import win32com.server.register
    win32com.server.register.UseCommandLine(RedisRealtime)
```

注册组件到操作系统

>python Redis_COM.py

```
D:\PMS\RedisExcel>"c:/python35/python.exe" d:/PMS/RedisExcel/redis_com.py
Registering COM server...
Requesting elevation and retrying...
Registering COM server...
Registered: Redis.Snapshot
```

组件注册后，就可以在 Excel 中使用了。

从操作系统注销组件，执行

>python Redis_COM.py -unregister

```
D:\PMS\MonitoringWithExcel>"c:/python35/python.exe" d:/PMS/MonitoringWithExcel/PutData2RedisThread_U2.py
                                                    DCS2AI
Point: CSDC.DCS2AI.2JZA0017 [b'1073742000.0', b'2017-01-09 17:18:55.535355']
Point: CSDC.DCS2AI.2JZA0017 [b'1073742000.0', b'2017-01-09 17:18:57.561179']
Point: CSDC.DCS2AI.2JZA0017 [b'1073742000.0', b'2017-01-09 17:18:59.527951']
```

2.4.2 数据服务测试

Excel 工作簿模块 TestModel 中的函数 testgetsnapshot()用于测试 COM 组件。

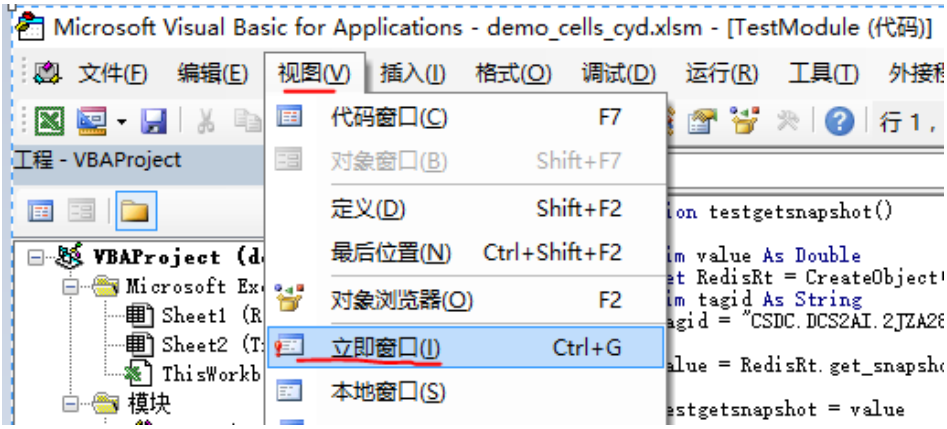
```
Function testgetsnapshot()

    Dim value As Double
    Set RedisRt = CreateObject("Redis.Snapshot")
    Dim tagid As String
```

```
    tagid = "CSDC.DCS2AI.2JZA1308"
    value = RedisRt.get_snapshot(tagid)
    testgetsnapshot = value

    Debug.Print (value)

End Function
```

运行子过程时：

Debug.Print (value)

的结果将显示在“立即窗口”



“立即窗口”显示的数据说明，COM 组件注册成功，可以被 VBA 调用。

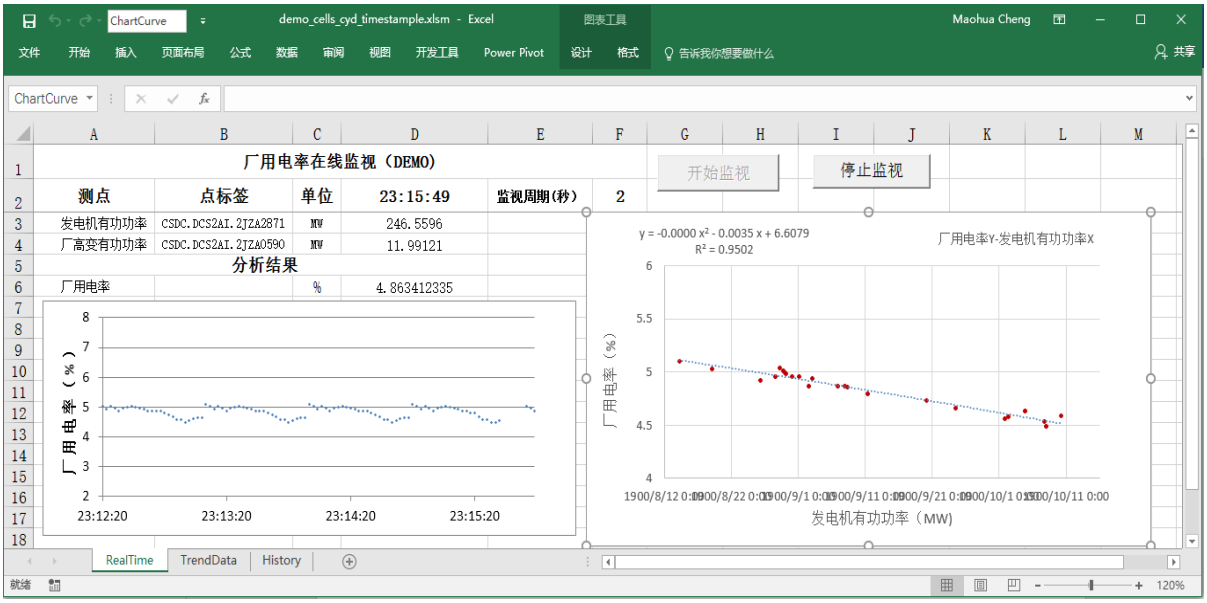
三、 监视 Excel 工作簿

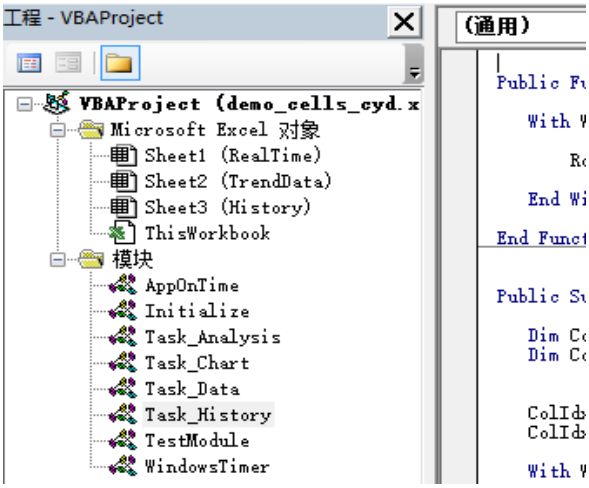
实例提供了 2 个过程监视 Excel 工作簿 demo_cyd_cells_time.xls、demo_cyd_var_time.xls，对应使用单元格（cells）、变量(var)为图形数据源的不同实现方案。

过程监视 Excel 工作簿有三个表单和若干 VBA 模块构成。

1）表单：用户操作、趋势线数据源和历史数据表单；

2）VBA 模块：实现数据采集、分析和存储等功能。





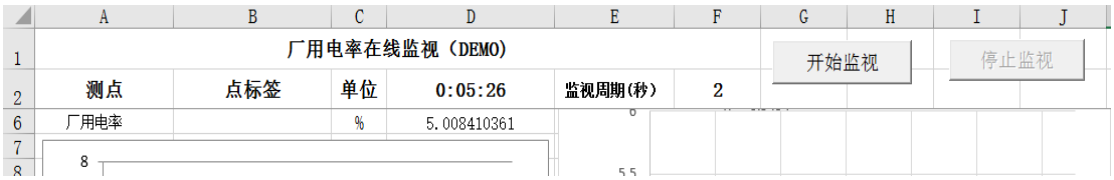
3.1 Excel 表单

3.1.1 用户操作表单

Excel 表单默认名称是“sheet*”，命名表单反映工作内容，用户操作表单命名为 RealTime。这个表单中用三类元素：数据单元格、趋势线、操作控件，提供启动、停止监视和显示实时数据等功能。

Excel2016 默认显示功能区，在线监视时，建议“折叠功能区”，获得更大显示区域。

表单冻结第 1-2 行，其中是测量点属性定义和操作控件等重要表单元素。



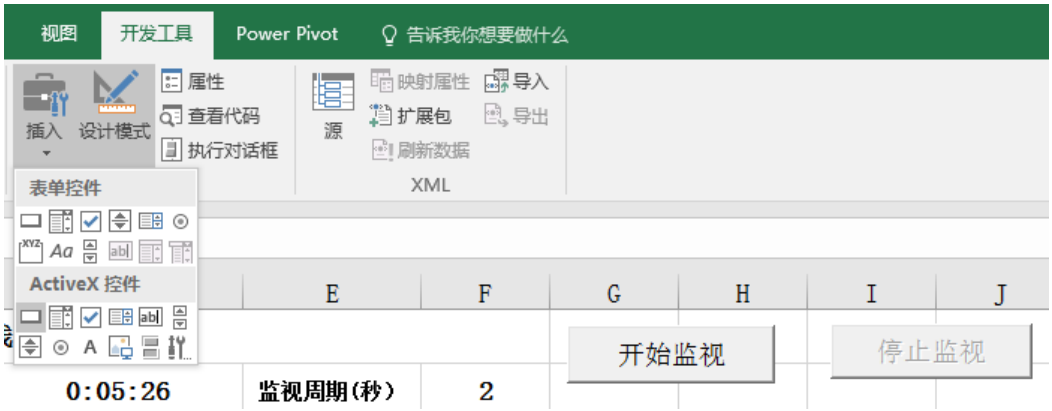
表单第二行是测量数据点属性定义。各数据点属性值从第三行开始，按照一定的顺序，在下面逐个按行提供。

需要注意的是：表单名、属性顺序、数据点顺序确定后不要轻易改变。。因为 VBA 根据这些信息所在表单的单元格坐标，取、写数据，如果改动这些信息的位置，需要修改相应程序源码。如 Task_Data 模块 CollectData 方法使用单元格坐标取测点标签信息：

```
With Worksheets(sheetname)
  For i = RowIdxBeg To RowIdxBeg + TagCount - 1

    RowIdxCur = i
    tag = .Cells(RowIdxCur, ColIdxTag).value

    .Cells(RowIdxCur, ColIdxVal).value = getsnapshot(tag)
  Next i

End With
```

表单中的“操作控件”是 Active 控件。Active 控件可直接放在表单中，然后，对控件动作进行编程。（操作控件可实现的方式还有：菜单、表单控件等，这些控件对应的程序相对复杂，Excel 版本兼容性不好，不建议使用）。

Active 控件可在开发工具的“设计模式”下修改属性、编写动作响应代码：



然后，编写启、停监视事件响应代码：

```
Private Sub StartCmdBtn_Click()
   '从表单取监视周期
   cRunIntervalSeconds = Worksheets("RealTime").Range("F2")

   '历史数据表单中新数据行号
   RowIdxNull = RowIdxNullHistorySheet("History")



   Call StartOnlineMonitor  'App 定时器

   'Call StartTimer          'Windows 定时器

   monitoring = True

   StartCmdBtn.Enabled = False
   StopCmdBtn.Enabled = True
End Sub

Private Sub StopCmdBtn_Click()

   Call StopOnlineMonitor ' App 定时器

   'Call EndTimer         'Windows 定时器

   monitoring = False

   StartCmdBtn.Enabled = True
   StopCmdBtn.Enabled = False

End Sub
```

3.1.2 动态数据线数据源表单

动态数据线数据源表单 TrendData，其中存放有固定数量的当前运行和分析数据。再作为动态数据线数据源的同时，也可用于监视当前时段的数据。

	A	B	C
1	时间	发电机有功功率	厂用电率
2	2017/1/4 23:20:51	244.1162037	4.958097824
3	2017/1/4 23:20:53	243.1229977	4.98397519
4	2017/1/4 23:20:55	242.3231944	5.037903923
5	2017/1/4 23:20:57	243.0790972	4.990498977
6	2017/1/4 23:20:59	245.1532986	4.956641416
7	2017/1/4 23:21:01	247.0956944	4.937439219

监视时，表单中数据实时更新，相应的时序、关系数据线动态显示。动态时间线实现代码稍多，有关内容在“四、动态数据线”节中单列。

3.1.3 历史数据表单

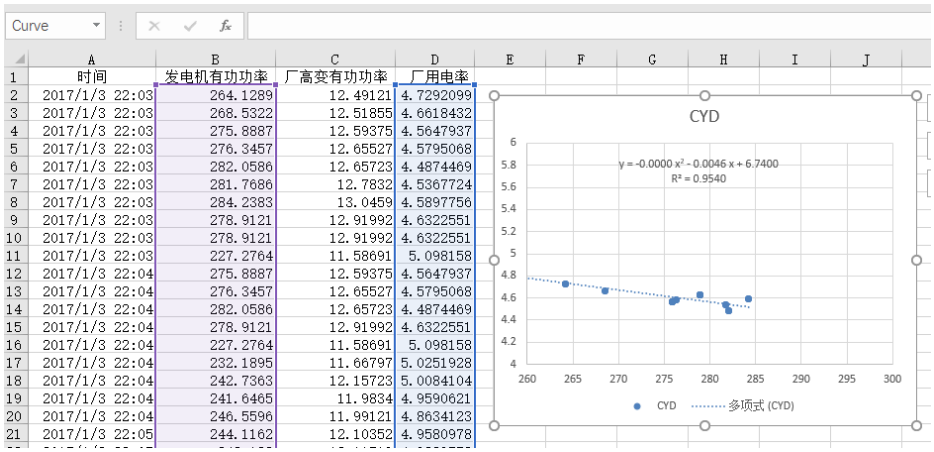
用户存储历次监视历史数据表单 History。每次启动监视时，程序自动从最后一个历史记录数据行的下一行开始。

	A	B	C	D
1	时间	发电机有功功率	厂高变有功功率	厂用电率
2	2017/1/3 21:46:24	284.24	13.05	4.589775551
3	2017/1/3 21:46:26	284.24	13.05	4.589775551
4	2017/1/3 21:46:28	284.24	13.05	4.589775551
5	2017/1/3 21:46:30	284.24	13.05	4.589775551

用户可使用历史数据，做离线分析。如用历史数据做：

厂用电率=f(发电机有功率)

关系分析：



	A	B	C	D
1	时间	发电机有功功率	厂高变有功功率	厂用电率
2	2017/1/3 22:03	264.1289	12.49121	4.7292099
3	2017/1/3 22:03	268.5322	12.51855	4.6618432
4	2017/1/3 22:03	275.8887	12.59375	4.5647937
5	2017/1/3 22:03	276.3457	12.65527	4.5795068
6	2017/1/3 22:03	282.0586	12.65723	4.4874469
7	2017/1/3 22:03	281.7686	12.7832	4.5367724
8	2017/1/3 22:03	284.2383	13.0459	4.5897756
9	2017/1/3 22:03	278.9121	12.91992	4.6322551
10	2017/1/3 22:03	278.9121	12.91992	4.6322551
11	2017/1/3 22:03	227.2764	11.58691	5.098158
12	2017/1/3 22:04	275.8887	12.59375	4.5647937
13	2017/1/3 22:04	276.3457	12.65527	4.5795068
14	2017/1/3 22:04	282.0586	12.65723	4.4874469
15	2017/1/3 22:04	278.9121	12.91992	4.6322551
16	2017/1/3 22:04	227.2764	11.58691	5.098158
17	2017/1/3 22:04	232.1895	11.66797	5.0251928
18	2017/1/3 22:04	242.7363	12.15723	5.0084104
19	2017/1/3 22:04	241.6465	11.9834	4.9590621
20	2017/1/3 22:04	246.5596	11.99121	4.8634123
21	2017/1/3 22:05	244.1162	12.10352	4.9580978

3.2 VBA 模块

监视 Excel 工作簿中，有 8 个 VBA 模块。这些模块可以分为两类：监视任务无关通用模块和任务相关模块。

1）任务无关模块：

Initialize：初始化监视软件

AppOntime：APP 定时器,定时运行监视任务

WindowsTimer：系统定时器

Testmodule：开发中的一些测试方法（见 2.4.2）

2）任务相关模块：

Task_Data：实时数据

Task_History：历史数据

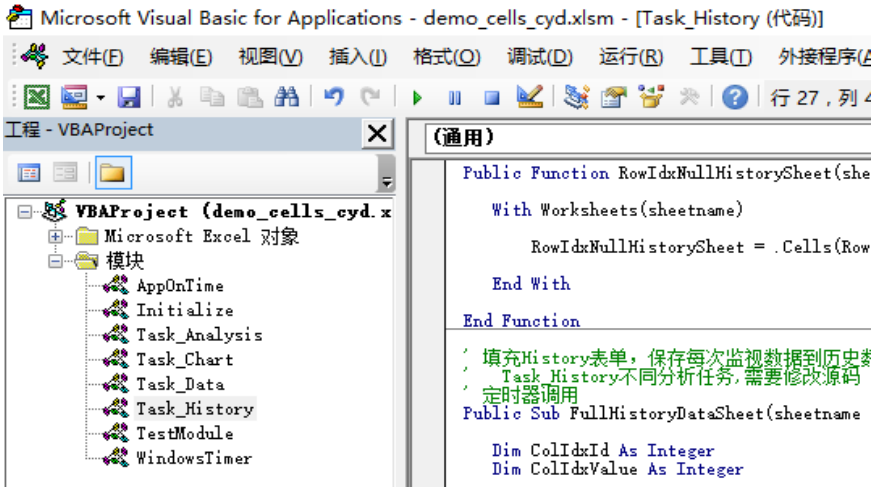
Task_Analysis：计算分校

Task_Chart：见“四 动态数据线”

任务相关模块分别对应程序的数据层，分析层、用户层（UI）。和任务相关模块在分析任务改变时，需要修改以满足需要。为了便于辨识模块类型，任务相关模块都以“Task_”为前缀命名，名称为“Task_*”。

在修改任务相关模板实现定制任务时，不要改模板名称，也不要改模块内方法和函数的名称，只改其中的代码。

定制任务时需要修改的代码，源码中都加了注释说明。



3.2.1 工作簿初始化

工作簿初始化模块 Initialize，有全局变量和 Sub Auto_Open()、Sub Auto_close()两个特殊方法。这两个方法在工作簿打开和关闭时被自动调用，用于完成工作簿的初始化和结束工作。

```
’  全局量
Public monitoring As Boolean ’监视状态
Public cRunIntervalSeconds As Double ’ 监视周期，从单元格
Worksheets("RealTime").Range("F2")获取

Public IdIdx As Integer  ’IdIdx 连续采集的数据组数序号，数值最大到：
容许的最大显示点数
Public RowIdxNull As Integer ’历史数据表单中追加新数据的行号

’特殊方法：打开 Excel 时自动调用
Sub Auto_Open()

  ’1 定义于 AppOnTime 模块，用于标识：监视状态
  monitoring = False
   ’2 本次连续采集的数据组数序号
  IdIdx = 0
  ’ 3 设置时序线 X 轴上下限 - 时间轴
  With
Sheets("RealTime").ChartObjects("ChartTrend").Chart.Axes(xlCategory)
    .MinimumScale = Now
    .MaximumScale = Now + TimeSerial(0, 3, 20)
 End With
   ’4 按钮恢复初始状态
   Sheets("RealTime").OLEObjects("StartCmdBtn").Object.Enabled =
True
   Sheets("RealTime").OLEObjects("StopCmdBtn").Object.Enabled =
```

```
False

End Sub

'特殊方法：退出 Excel 时自动调用
Sub Auto_close()

  '1 清除动态趋势线用数据
   Worksheets("TrendData").Range("A2:C101").ClearContents

   '2 自动保存当前工作簿
   ActiveWorkbook.Save
End Sub
```

3.2.2 定时循环

定时循环可用 Application.Ontime 和 Windows Timer 定时器。操作系统 Windows Timer 定时器时的 id，是 VBA 的公共变量。如果运行中发生什么异常，电脑会丢了计时的 id，这样计时器就关不掉了，Excel 会死锁。原则上不用操作系统定时器时，用 Application.ontime。纯 Application.ontime 只能最快 1 秒钟执行一次，需要速度快于 1 秒的才有必要使用操作系统定时器

VBA 中提供了 2 种定时器的模块代码。监视工作通常不会快一 1 秒，为程序稳健起见，实例中使用 Application.Ontime 定时器（定时循环代码在 AppOntime 模块）

循环周期参数定义在 RealTime 表单中。

E	F
监视周期(秒)	2

```
Private Sub StartCmdBtn_Click()
  cRunIntervalSeconds = Sheets("RealTime").Range("F2")
```

```
Public Sub StartOnlineMonitor()

   RunWhen = Now + TimeSerial(0, 0, cRunIntervalSeconds)
```

如需要改变周期，修改 Sheets("RealTime").Range("F2")中的数值。

模块 AppOnTime 源码

```
'
' 使用 Application.OnTime 定时执行任务
'
'  适用于周期大于 1 秒任务
```

```
'
'   三个表单名称改变后，需修改下面代码，使用当前表单名称
'

Dim RunWhen As Double

Public Sub StartOnlineMonitor()


  '1 从 Redis 取数据到表单，Task_Data 对应的表单会因为任务不同而改变，这时需修改 CollectData 源码
  Call CollectData("RealTime")

  '2 从表单取数据计算，然后，将结果返回表单， Task_Analysis 不同分析任务,需改源码
  Call calPerformance("RealTime")

  '3 填充 TrendData 表单，自动更新图形，Task_Chart 不同分析任务,需改源码
  Call FullTrendDataSheet("TrendData")

  '4 填充 History 表单，保存历史数据，Task_History 不同分析任务,需修源码
  Call FullHistoryDataSheet("History")


  Sheets("RealTime").Range("D2").value = Now

  RunWhen = Now + TimeSerial(0, 0, cRunIntervalSeconds)

  Application.OnTime RunWhen, "StartOnlineMonitor"

End Sub

'分析任务无关，不需要改动
Public Sub StopOnlineMonitor()

    If (monitoring = True) Then
      Application.OnTime RunWhen, "StartOnlineMonitor", , False

    End If

End Sub
```

3.2.3 实时数据

模块 Task_Data 中提供 getsnapshot 函数和 CollectData，用于从 Redis 提取运行数据，写到表单中。

```
'调用 COM 组件，从 Redis 取数据，被本模块的 CollectData 调用
Public Function getsnapshot(ByVal tag As String)

    Dim value As Double
    Set RedisRt = CreateObject("Redis.Snapshot")

    value = RedisRt.get_snapshot(tag)

    getsnapshot = value

End Function


'AppOnTime 模块调用，定时执行，从 Redis 取数据到表单 RealTime
Public Sub CollectData(sheetname As String)

   Dim RowIdxBeg As Integer ' 第一个数据点行号
   Dim ColIdxTag As Integer ' 数据点标签的列号
   Dim ColIdxVal As Integer ' 数据点运行数据的列号
   Dim RowIdxCur As Integer
   Dim TagCount As Integer '总点数

   '从表单取数据，不同任务的表单设计不同，需要修改相应的位置坐
标

   RowIdxBeg = 3
   ColIdxTag = 2
   ColIdxVal = 4
   TagCount = 2

  With Worksheets(sheetname)
     For i = RowIdxBeg To RowIdxBeg + TagCount - 1

       RowIdxCur = i
       tag = .Cells(RowIdxCur, ColIdxTag).value

       .Cells(RowIdxCur, ColIdxVal).value = getsnapshot(tag)
     Next i
```

```
   End With

End Sub
```

3.2.4 历史数据

模块：Task_History，用于存储用户历次监视数据。每次启动监视时，程序自动从最后一个记录数据行的下一行开始。

1）RowIdxNullHistorySheet:检测最后一个记录数据行的下一行号；
2）FullHistoryDataSheet:存储监视数据
3）ClearHistorySheet:用于清除历史数据

```
Public Function ClearHistorySheet()

   Sheets("History").Rows.Clear
   Sheets("History").Cells(1, 1) = " 时间"
   Sheets("History").Cells(1, 2) = " 发电有功"
   Sheets("History").Cells(1, 3) = " 厂用电"
   Sheets("History").Cells(1, 4) = " 厂用电率"
End Function

' 检测最后一个记录数据行的下一行号
Public Function RowIdxNullHistorySheet(sheetname As String)

   With Worksheets(sheetname)

       RowIdxNullHistorySheet = .Cells(Rows.Count,
"A").End(xlUp).Offset(1, 0).Row ' 表示为最后一行为空白的行号

   End With

End Function

' 填充 History 表单，保存每次监视数据到历史数据表单
'   Task_History 不同分析任务,需要修改源码
' 定时器调用
Public Sub FullHistoryDataSheet(sheetname As String)

   Dim ColIdxId As Integer
   Dim ColIdxValue As Integer


   ColIdxId = 1
```

```
      ColIdxValue = 2

      With Worksheets(sheetname)

         .Cells(RowIdxNull, ColIdxId).value = Now
         .Cells(RowIdxNull, ColIdxValue).value = cyd.power '趋势线
   的数据 -任务不同需要改
         .Cells(RowIdxNull, ColIdxValue + 1).value = cyd.gb
         .Cells(RowIdxNull, ColIdxValue + 2).value = cyd.rcyd

           RowIdxNull = RowIdxNull + 1

      End With

   End Sub
```

3.2.5 分析模块

分析模块 Task_Analysis，从数据表单提取数据，计算分析，将结果写回表单

```
'Option Explicit

' 分析模块,基本元素

'  1 分析对象的数据结构定义
'  2 分析对象的内存变量
'  3 计算分析
'
'  calPerformance 被 AppOnTime 调用

Public Type plantpower
   power As Double '发电机有功功率
   gb As Double    '高厂变有功率
   rcyd As Double  '厂用电率
End Type

Public cyd As plantpower

Sub calCyd(cyd As plantpower)
  cyd.rcyd = 100 * cyd.gb / cyd.power
End Sub

'  calPerformance 被 AppOnTime 调用
Public Sub calPerformance(sheetname As String)
  Dim RowIdxBeg As Integer
```

```
    Dim ColIdxVal As Integer

    RowIdxBeg = 3

    ColIdxVal = 4

    With Worksheets(sheetname)

        cyd.power = .Cells(RowIdxBeg, ColIdxVal).value
        cyd.gb = .Cells(RowIdxBeg + 1, ColIdxVal).value

        Call calCyd(cyd) '调用计算模块

       .Cells(6, ColIdxVal).value = cyd.rcyd

    End With

End Sub
```

四、 动态数据线

动态数据线是程序中代码较多部分。动态数据线实现方案很多。完全程序、手工或者程序和手工结合都可以生成需要的目标。

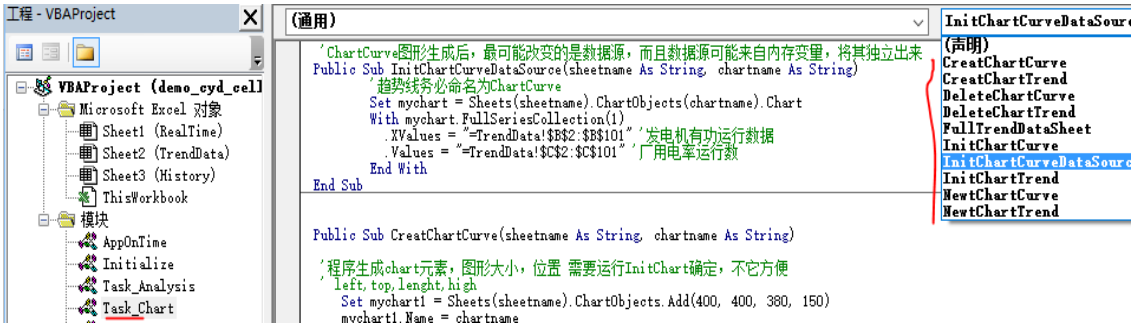
表单 RealTime 中有：时序和关系数据线，两个数据线。Task_Chart 中这两线的程序生成代码，用于程序生成基本图形，然后，手工细调的方式生成目标图形。这样可以减少代码复杂度，也能发挥 Excel 环境下，图形可视化定制方便的优点，更方便灵活地满足需求。

动态数据线的数据源可以是表单单元格（cells），也可以数组变量。示例中用使用了 TrendData 中表单单元格作为数据源，有关代码位于 Task_Chart 模块。

Task_chart 模块中生成和配置图形的方法 NewChartTrend、NewChartCurve，只需运行一次即可生成图形。

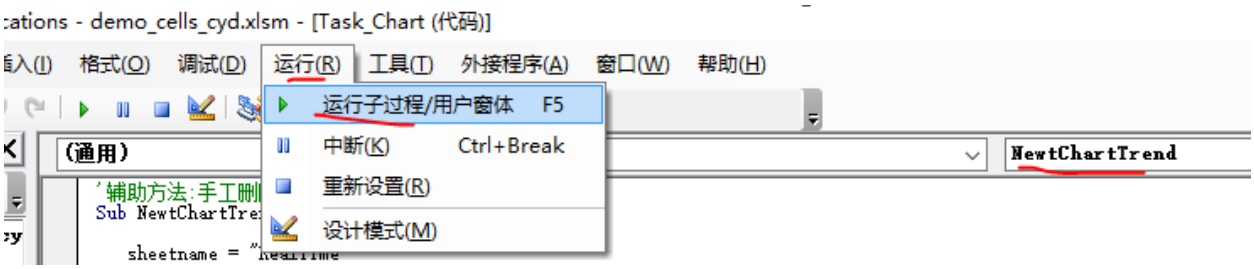
图形和和具体任务目标有关，定制时需要修改有关方法的代码，但是不要改方法名称，只改代码。

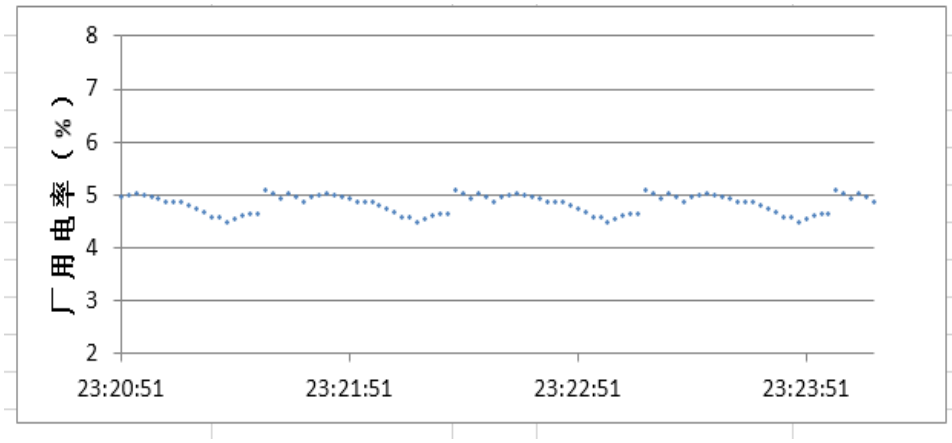
和任务有关需要代码修改的主要场景是：任务相关的数据源单元格不同。模块中提供了相应的方法，便于改变数据源。



4.1 时序趋势线

时序趋势线 ChartTrend 是表单 Realtime 中的嵌入式图形。用子程序 NewtChartTrend 首次生成基本图形，然后，进入 Realtime 表单手工细化调整。（也可完全手工给出时序趋势线）。





建立 ChartTread 的代码如下：

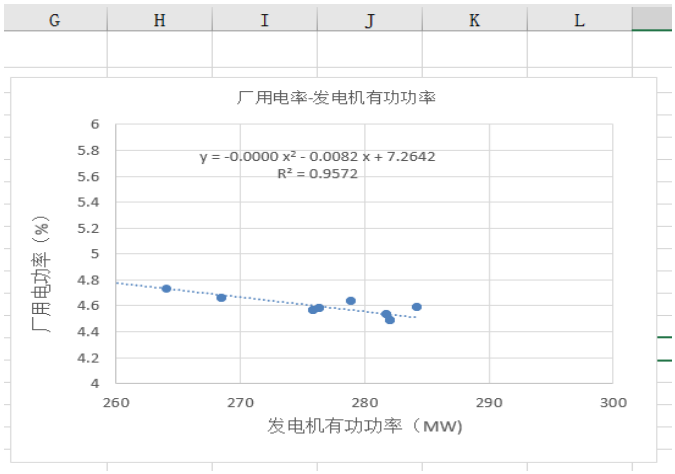
```
'============================================================
'
'   辅助方法 1 将图形创建独立出来，方便通过“运行”->"运行子过程"
'
'   程序生成 chartREend 基本图形，图形建立后，可手工调整大小、位置
'
'============================================================================

' 删除表单中的图形，运行子程序，建立新图形
Sub NewtChartTrend()

    Dim sheetname    As String
    Dim chartname   As String

    sheetname = "RealTime"
    chartname = "ChartTrend"

    Call DeleteChartTrend(sheetname, chartname)

    Call CreatChartTrend(sheetname, chartname)
    Call InitChartTrend(sheetname, chartname)
```

```
End Sub

Public Sub CreatChartTrend(sheetname As String, chartname As String)

    Set mychart = Worksheets(sheetname).ChartObjects.Add(320, 50, 380, 150)
' left,top,lenght,high
    mychart.Name = "ChartTrend"

End Sub


Public Sub DeleteChartTrend(sheetname As String, chartname As String)
   Set mychart = Worksheets(sheetname).ChartObjects(chartname)
   mychart.Delete
End Sub

 '初始化图形 ChartTrend"
Public Sub InitChartTrend(sheetname As String, chartname As String)

      Set mychart = Worksheets(sheetname).ChartObjects(chartname).Chart

      '1 图标类型 绘制散点图
      mychart.ChartType = xlXYScatter

      mychart.HasLegend = False

      '2 加数据序列
      mychart.SeriesCollection.NewSeries
      mychart.SeriesCollection(1).MarkerStyle = 2  '图标的样式和大小
      mychart.SeriesCollection(1).MarkerSize = 2

      '***  数据来自表单
      mychart.FullSeriesCollection(1).XValues = "=TrendData!$A$2:$A$101"  '
时间数据
      mychart.FullSeriesCollection(1).Values = "=TrendData!$C$2:$C$101"'
厂用电率运行数据

       '3 设置主 Y 轴（数值轴 xlValue, 主 xlPrimary)
      With mychart.Axes(xlValue, xlPrimary)
         .MinimumScale = 4
          MaximumScale = 7
         .MajorUnit = 0.5                    '设置轴的单位，即刻度
```

```
        .HasTitle = True                    'Y 轴注释
        .AxisTitle.Text = "厂用电率（%）"
    End With

    ' 设置字体与图表大小*
    With mychart.Axes(xlValue,
xlPrimary).AxisTitle.Format.TextFrame2.TextRange.Font
      .BaselineOffset = 0
      .Bold = msoTrue
      .Italic = msoFalse
      .Spacing = 3
    End With

    '4 设置 X 轴(xlCategory 轴)上下限
    With mychart.Axes(xlCategory)
        .MinimumScale = Now
        .MaximumScale = Now + TimeSerial(0, 3, 20)
        .MajorUnit = TimeSerial(0, 1, 0)    '设置轴的单位，即刻度
        .TickLabels.NumberFormatLocal = "[$-x-systime]h:mm:ss AM/PM"
    End With

End Sub
```

4.2 关系数据线

关系数据线命名为 ChartCurve。手工在 Realtime 表单中，嵌入散点图，图形中有一个数据系列，一个 2 次多项式趋势线。配置纵、横坐标轴的上、下限、坐标轴标题等信息，用程序给定数据系列数据源的方式实现图形定义。软件使用中可根据需要调整显示效果（也可用 Task_Chart 中的 NewtChartCurve()和生成时序线一样方式实现）。



ChartCurve 图形生成后，最可能会变的是数据源，将其独立为一个方法：

```
Public Sub InitChartCurveDataSource(sheetname As String, chartname As
```

```
String)
     Set mychart = Sheets(sheetname).ChartObjects(chartname).Chart
      With mychart.FullSeriesCollection(1)
        .XValues = "=TrendData!$B$2:$B$101" '发电机有功运行数据
        .Values = "=TrendData!$C$2:$C$101" '厂用电率运行数
      End With
End Sub
```

4.3 数据源

4.3.1 单元格数据源

两个数据线的数据源自 TreadData 表单的 cells。如上 4.1 代码所示，可在图形生成时设定。

	A	B	C
1	时间	发电机有功功率	厂用电率
2	2017/1/4 23:20:51	244.1162037	4.958097824
3	2017/1/4 23:20:53	243.1229977	4.98397519
4	2017/1/4 23:20:55	242.3231944	5.037903923
5	2017/1/4 23:20:57	243.0790972	4.990498977

图形形态定义好后，基本不会改变异变的是其数据源，所以，Task_Chart 提供 2 个方法为两个数据线的设定数据源。

图形定义指定数据源自表单的 cells 后，定时监视循环调用 FullTrendDataSheet 更新 cells 中数据，图形自动更新。

下面代码中逻辑关系比较复杂的是对采集数据开始时间和时间轴上下限的对应关系的判断和调整。

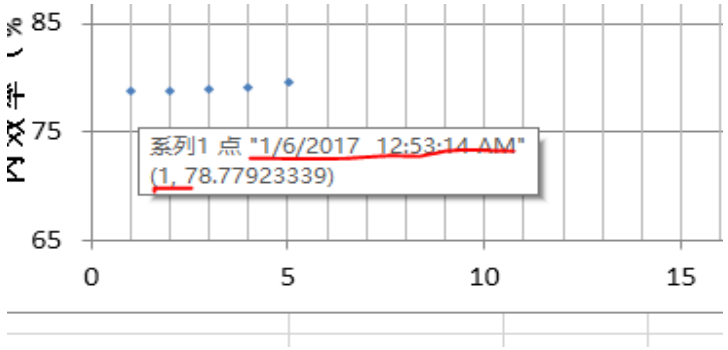
```
' 填充数据源表单，更新数据线,sheetname 是图形用数据源所在表单
Public Sub FullTrendDataSheet(sheetname As String)

   Dim RowIdxBeg As Integer
   Dim ColIdxTime As Integer '时间标签列号
   Dim ColIdxValue As Integer ' 第一个数据点数据列号

   Set mychart = Sheets("RealTime").ChartObjects("ChartTrend").Chart

   RowIdxBeg = 1
   ColIdxTime = 1
   ColIdxValue = 2

   If (IdIdx <= Uplimt) Then
```

```
    IdIdx = IdIdx + 1
    With Worksheets(sheetname)
     .Cells(RowIdxBeg + IdIdx, ColIdxTime).value = Now '时间标签
     .Cells(RowIdxBeg + IdIdx, ColIdxValue).value = cyd.power '需要修改
     .Cells(RowIdxBeg + IdIdx, ColIdxValue + 1).value = cyd.rcyd '需要修改
    End With
  End If

   If (IdIdx < Uplimt) Then '保证首次开始监视的时间为时间轴的最小值
       If (mychart.Axes(xlCategory).MinimumScale <
Sheets("TrendData").Range("A2").value) Then
          mychart.Axes(xlCategory).MinimumScale =
Sheets("TrendData").Cells(2, 1).value
          mychart.Axes(xlCategory).MaximumScale =
Sheets("TrendData").Cells(2, 1).value + (TimeSerial(0, 3, 20)) '这个最大数
据点和采样间隔有关
       End If

       If (mychart.Axes(xlCategory).MaximumScale <
Sheets("TrendData").Cells(RowIdxBeg + IdIdx, ColIdxTime).value) Then
          mychart.Axes(xlCategory).MaximumScale =
Sheets("TrendData").Cells(RowIdxBeg + IdIdx, ColIdxTime).value
           mychart.Axes(xlCategory).MinimumScale =
Sheets("TrendData").Cells(RowIdxBeg + IdIdx, ColIdxTime).value -
(TimeSerial(0, 3, 20))
       End If
  End If

  If (IdIdx > Uplimt) Then

    '  向上一行移动数据
    For k = 1 To Uplimt
        With Sheets("TrendData")
            .Cells(RowIdxBeg + k, ColIdxTime).value = .Cells(RowIdxBeg + k
+ 1, ColIdxTime).value '时间标签
            .Cells(RowIdxBeg + k, ColIdxValue).value = .Cells(RowIdxBeg +
k + 1, ColIdxValue).value
            .Cells(RowIdxBeg + k, ColIdxValue + 1).value
= .Cells(RowIdxBeg + k + 1, ColIdxValue + 1).value
        End With
    Next

    '  当前新数据在最后一行
    With Worksheets(sheetname)
```

```
        .Cells(RowIdxBeg + Uplimt, ColIdxTime).value = Now '时间标签
        .Cells(RowIdxBeg + Uplimt, ColIdxValue).value = cyd.power  '需修改
        .Cells(RowIdxBeg + Uplimt, ColIdxValue + 1).value = cyd.rcyd  '需修改
      End With
      With mychart.Axes(xlCategory)
          .MinimumScale = Sheets("TrendData").Cells(2, ColIdxTime).value
          .MaximumScale = Sheets("TrendData").Cells(101, ColIdxTime).value
      End With

    End If
End Sub
```

4.3.2 数组变量数据源

Chart 的 XValues 数据源是日期数组变量时，如果直接使用日期数值，作为图形数据点的数据，数据点接受后的信息异常：



数据点（x,y）中的 x 不是预期的“日期数据”，而是数据点“序号”，时间变成了数据点的“标签”（Label）。

msgroups.net 上面对问题：“Line chart - time series - category axis date format (2007 VBA)"的回答：

http://msgroups.net/microsoft.public.excel.charting/line-chart-time-series-catego/232668

中给出解释：

> Make sure that **all of the cells** which **look** like **dates** to you are **real numeric dates.** It only takes on to be entered as text for Excel to ignore the time scale setting.

原因在于：数据源在 cells 中时，VBA 中内部对来自工作表元素的日期，都自动视为日期对应的实数处理。现在用日期类型变量，VBA 接受到的变量数值是日期类型数值，VBA 没有自动将“日期”变量转换到对应“实数”的机制，需要程序显式将“日期”数据转换为对应“实数”。有关代码如下：

1）定义时间数组为 Variant 类型

```
Public curtime(20) As Variant
```

2）指定 xlCategory 的数据源，类型和显示格式等

```
ActiveChart.FullSeriesCollection(1).XValues = curtime
```

```
    ActiveChart.Axes(xlCategory).CategoryType = xlTimeScale

    ActiveChart.Axes(xlCategory).MinimumScale = Now
    ActiveChart.Axes(xlCategory).MaximumScale = Now + TimeSerial(0, 3, 20)
'设置轴的单位，即刻度
ActiveChart.Axes(xlCategory).MajorUnit = TimeSerial(0, 1, 0)
ActiveChart.Axes(xlCategory).TickLabels.NumberFormatLocal = "h:mm:ss"
```

3）转换日期为实数给 curtime 数组表示时间

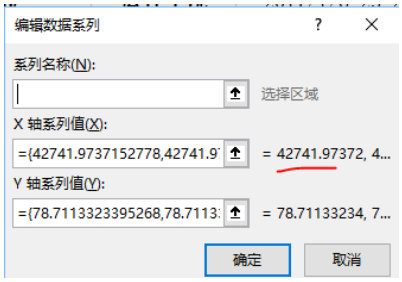
日期数据不能用如下代码，直接使用时间

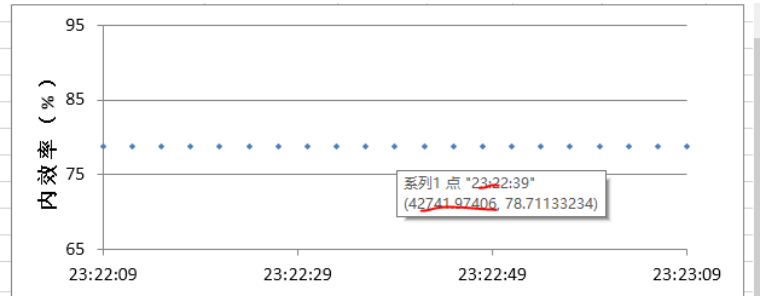
```
curtime(flag) = Now
```

要用 CDbl 将其转换为实数，这是关键所在。

```
curtime(flag) = CDbl(Now)
```

转换后，数据点（x,y）中的 x 是预期日期实数，图形按照格式定义正确显示。





详细代码，见使用日期数组变量作为数据源的工作簿：demo_cyd_var_time.xlsm。

小结

本文档以厂用电率为例，给出了工业过程监视 Excel 工作簿实现技术。通过学习本文档及示例 Excel 监视软件：1）可以掌握使用 Excel VBA 编程实现在线监视的软件技术；2）以该示例为通用分析 Excel 模板，快捷定制监视软件。

参考文献

[1] Excel Home. 别怕,Excel VBA 其实很简单(第 2 版)[M].北京：北京大学出版社. 2016.07

[2] Excel Home. http://www.excelhome.net/

[3] John Walkenbach. Excel 2013 Power Programming with VBA[M]. John Wiley & Sons, Inc. USA. s2013

[4] Getting Started with VBA in Office 2010
https://msdn.microsoft.com/library/office/ee814735(v=office.14)
[5] Excel VBA Programming
http://www.homeandlearn.org/the_excel_vba_editor.html
[6] James Ma Weiming. Mastering Python for Finance, Chapter 10 Excel with Python[M]. P291-304. Packt Publishing. 2015
[7] Mark Hammond. Python for Windows Extensions.
https://sourceforge.net/projects/pywin32/
[8] 李子骅. Redis 入门指南[M]. 北京：人民邮电出版社. 2013
[9] Guido van Rossum, Python development team. Python Tutorial.
https://docs.python.org/tutorial/index.html