# SDWb_Lua 倒计时功能使用说明

Lua 脚本提供了 32 个 ID 为 0-31 的超时定时器，利用 Lua 脚本里的这些超时定时器，可以极大程度丰富 SDWb 串口屏的功能。本案例所介绍的倒计时功能就是其中的一种典型应用。整个过程无需用户单片机参与、无需串口指令控制。

## 一．案例功能介绍

### 1.1 界面功能

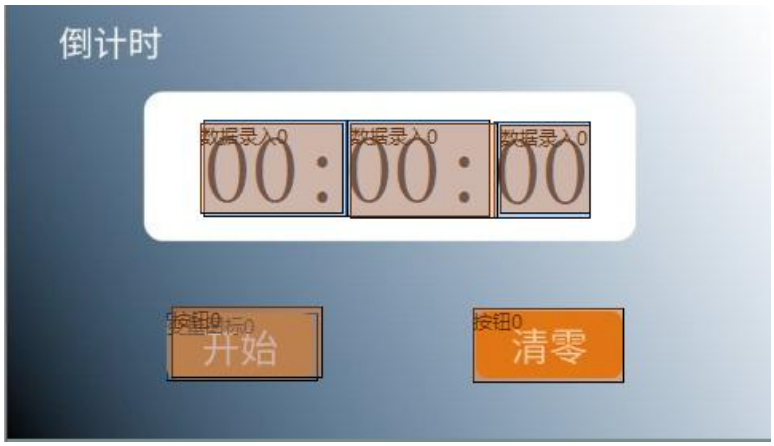


图 1 倒计时界面

倒计时界面如图 1 所示，界面上有时、分、秒三个参数可以设置，设置完后点击“开始/暂停”按钮控制倒计时开启和关闭。当按钮处于“开始”状态下，点击该按钮，如果设置的时间不为 0，则倒计时开始，同时该按钮切换成“暂停”状态。按钮处于“暂停”状态时，点击该按钮倒计时暂停，同时该按钮切换为“开始”状态。点击“清零”按钮，倒计时停止，并且将时间显示框内的时间数值清零。

### 1.2 控件设计

如图 1 所示，倒计时的时、分、秒的实时显示与参数录入，用到了数据变量显示控件和数据录入控件组合实现。首先使用数据录入控件录入数据，再通过数据变量控件将录入的数据显示出来，此过程无需 Lua 脚本参与。本案例时、分、秒三个变量的地址分别为 0x0001、0x0002、0x0003。

“开始/暂停”按钮用于控制倒计时的开始以及暂停，清零按钮控制倒计时清零。这两个按钮都需要设置按键键码，以便在触摸事件发生时，能够执行 Lua 中的触摸回调函数，通过页面编号以及按键键码，可以在 Lua 代码中定位到对应的按钮控件。

“开始/暂停”按钮以及清零按钮的键码设置如图 2 和图 3 所示。除此之外，在任何时刻点击修改倒计时时间都会让倒计时暂停，所以设置时、分、秒的数据录入控件也设置了键码分别为 3、4、5。触发这些录入控件后 Lua 脚本需要执行的代码比较简单，这里不做说明，用户自行查看 Lua 脚本里的注释。


图 2 开始/暂停按钮键码

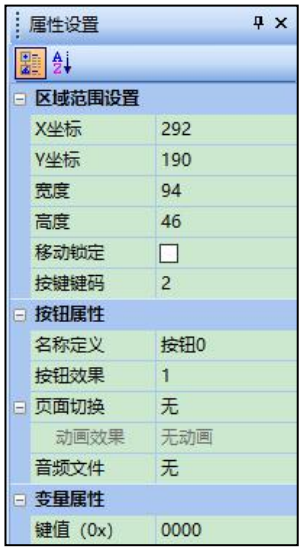


图 3 清零按钮键码


图 4 变量图标控件属性

因为“开始/暂停”按钮有“开始”和“暂停”两种状态，由一个按钮控制。所以在一个按钮下方放置了一个变量图标来指示按钮现在所处的状态。控件属性如图 4 所示，其变量地址设置为 0x0004，按照该设置往 0x0004 地址里写入 0x0000 时显示 0 号图标库文件的 1 号图标，即“开始”图标，往地址里写入 0x0001 时显示 0 号图标库文件的 2 号图标，即“暂停”图标。

### 1.3 流程设计

倒计时案例功能包括倒计时的开始和暂停以及清零，这里主要置介绍倒计时从设置完后点击“开始/暂停”按钮来控制倒计时开始和暂停的流程，“清零”按钮操作这里不做详细说明，用户自行参考 Lua 脚本的代码注释。其流程图如图 5 所示。用户设置完倒计时的时间后，点击“开始/暂停”按钮，将在 Lua 脚本中触发触摸回调函数来控制倒计时的开始以及暂停，之后再设置定时器，通过触发定时器回调函数来刷新倒计时时间显示以及控制蜂鸣器鸣响 5 次提示倒计时结束。

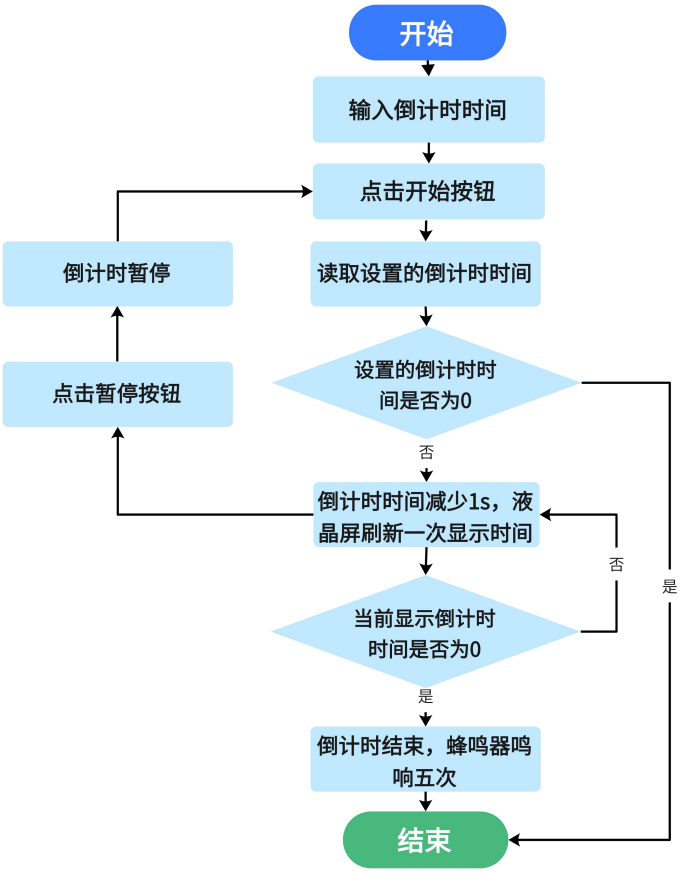


图 5 倒计时开始与暂停流程图

## 二．初始化回调函数

该案例中初始化回调函数主要用于设置串口相关参数。

```
--初始化回调函数
function callback_init()
    --设置串口屏模式为VGUS协议，波特率115200，串口格式8N1
    com_set_work_mode(0,0,115200,4)
end
```

该案例也可以使用默认设置，不调用初始化回调函数。

## 三．触摸回调函数

设置完参数后点击”开始/暂停”按钮，如图 2 所示，“开始/暂停”按钮是同一个按键，键码为 1，所以触发按键后串口屏开始调用触摸回调函数 callback_touch (pic_id,key_code,touch_state)，在回调函数下用 if 语句来判断是否触发该按键，当满足条件时开始执行 if 语句里的代码。

```
--当“开始/暂停”按钮点击并抬起，执行以下代码
```

```
if pic_id==0 and key_code==1 and touch_state==2 then
```

### 3.1 定义标志位，触发按钮对标志位赋值，区分按钮功能

当“开始/暂停”按钮点击并抬起执行以下代码：

```
--标志位flag 加1
flag=flag+1
```

在脚本的开头定义了一个标志位 flag，初始值为 0。每次触发“开始/暂停”按钮标志位将加 1，flag 为 1 时执行倒计时开始操作，flag 为 2 执行倒计时暂停操作同时将标志位清零。

### 3.2 控制倒计时开始

当标志位 flag 为 1 时，执行嵌套在 if 语句下的代码：

```
--判断标志位flag 的值，当标志位的值为1 时执行以下代码
if flag==1 then
```

#### 3.2.1 读取当前时间，并分别赋值给定义的时、分、秒的变量

```
--读取设置的倒计时时间，并赋值给hour，minute，second 变量
vgus_vp_read (0x0001,3,time_table)
--读取读取当前设置时间，将读取的时间分别赋值给定义的hour，minute，second 变量
hour=time_table[1]
minute=time_table[2]
second=time_table[3]
--计算设置的倒计时对应的总秒数
second_total=hour*3600+minute*60+second
```

用于显示以及录入时间的变量控件地址分别为 0x0001，0x0002，0x0003。通过读变量存储器函数 vgus_vp_read (vp_addr, read_len, read_table)，读取当前倒计时时间并赋值给定义的数据表 time_table 第 1-3 个元素。然后把这三个元素的值跟别赋给定义的 hour，minute，second 三个变量，分别对应显示时间的时，分，秒。接下来计算设置的倒计时时间对应的总秒数，这里定义一个变量 second_total 来表示该参数。

#### 3.2.2 判断是否执行开始倒计时操作

当计算出的倒计时时间对应的总秒数不为 0 时，倒计时开始，将执行以下代码：

```
--判断设置倒计时时间是否为0，非0 则开始倒计时，否则不进行任何操作，并将标志位flag 清零
if second_total>0 then
```

```
    --开启定时器0 ，定时模式为周期模式，计数方式为向上计数，定时器超时时间为1000ms
    vgus_timer_start(0,1,0,1000)
    --“开始/暂停”按钮位置显示暂停图标
    vgus_vp_var_write(0x0004,0,1)
```

首先启动定时器 0，定时模式为周期模式，计数方式为向上计数，定时器超时时间为 1000ms，意味着每秒刷新一次倒计时，倒计时时间每次减少 1 秒。与此同时，通过 vgus_vp_var_write(vp_addr, var_type,var_value)函数将“开始/暂停”位置的按钮下的图标修改为“暂停”图标，因为此时处于倒计时“开始”状态，再次点击按钮倒计时将暂停，所以这里的图标需要显示“暂停”图标。

当倒计时时间对应的总秒数为 0 时执行以下代码：

```
  else
     flag=0
```

总秒数为 0，无法进行倒计时，所以只需要将标志位 flag 清零即可。

### 3.3 控制倒计时暂停

当标志位 flag 为 2 时表示倒计时暂停，将执行以下代码：

```
if flag==2 then
     --停止定时器0
     vgus_timer_stop(0)
     --标志位清零
     flag=0
    --“开始/暂停”按钮位置显示“开始”图标
     vgus_vp_var_write(0x0004,0,0)
 end
```

当标志位为 2，将执行暂停倒计时操作，只需要停止定时器 0，然后将标志位 flag 清零，同时处于倒计时“暂停”状态时，再次点击按钮倒计时将重新开始。所以通过 vgus_vp_var_write(vp_addr, var_type,var_value)函数将“开始/暂停”下的图标改为“开始”图标。

## 四．定时器回调函数

当开启定时器后，串口屏通过 Lua 脚本调用定时器回调函数 callback_timer(timer_id)，在回调函数下使用 if 语句来判断开启的是哪一个定时器，满足条件后开始执行嵌套在 if 语句里的代码。这里只对定时器 0 嵌套的代码进行详细说明，定时器 1

嵌套的代码是在倒计时结束后执行一个每次触发蜂鸣器鸣响 1000ms 的操作。代码相对简单，且注释比较详细，这里就不再说明。

```
--开启定时器0后，执行以下代码
if timer_id==0 then
```

**4. 1 计算下一次显示的倒计时时间数值。**

这里采用的是 24 小时制时间格式，计算下一次显示倒计时的时间数值首先要在定时器启动后，每次在当前倒计时时间对应的总秒数基础上减少 1 秒：

```
second_total=second_total-1
```

**4.2 刷新倒计时显示时间数值**

刷新倒计时显示时间需要判断下一次显示的倒计时时间对应的总秒数是否减为 0，不为 0 则把总秒数转换为时、分、秒格式，为 0 则将倒计时时间对应的时、分、秒数值都显示为 0，同时关闭定时器 0，结束倒计时。

**4.2.1 下一次倒计时时间对应的总秒数大于 0**

下一次倒计时时间对应的总秒数大于 0 执行以下代码：

```
--判断下一次倒计时的总秒数是否大于0
if second_total>0 then
    hour=second_total//3600
    minute=second_total%3600//60
    second=second_total%3600%60
    --将每次计算出的时、分、秒数值重新赋值给数据标的第1-3个元素
    time_table={hour,minute,second}
```

当总秒数大于 0 时，将总秒数转换成时、分、秒，再分别赋值给 hour，ninute，second 变量。再将这三个变量赋值给先前定义的数据表 time_table。

**4.2.2 下一次倒计时时间对应的总秒数等于 0**

下一次倒计时时间对应的总秒数等于 0 执行以下代码：

```
--时、分、秒都为0执行以下操作：
else
    --倒计时时间为0
    time_table={0,0,0}
    --关闭定时器0
    vgus_timer_stop(0)
    --标志位清零
    flag=0
    --“开始/暂停”按钮位置显示为“开始”图标
```

```
    vgus_vp_var_write(0x0004,0,0)
    --蜂鸣器鸣响时间设置为 100ms，0x0a=10，单位为 10ms,10x10ms 为 100ms。
    beep_table[1]=0x0a
    --控制蜂鸣器鸣响 1 次
    vgus_reg_write(02,1,beep_table)
    --开启定时器 1，定时模式为周期模式，计数方式为向上计数，定时器超时时间为
1000ms
    vgus_timer_start(1,1,0,1000)
end
```

计算出倒计时总秒数为 0，代表倒计时结束，通过停止定时器 vgus_timer_stop(timer_id)函数来停止定时器 0。倒计时结束后“开始/暂停”按钮也应当恢复为“开始”状态，所以将标志位置 0，同时通过以数值方式写入变量存储器函数 vgus_vp_var_write(vp_addr, var_type,var_value)将 0x0004 的地址的数值改为 0，对应显示“开始”图标。同时设置蜂鸣器鸣响时间为 100ms，触发一次蜂鸣器，然后开启定时器 1 控制蜂鸣器鸣响 4 次，一共响 5 次，每次间隔 1s。

#### 4.2.3 将计算出的倒计时时间刷新显示

将上述两种情况下计算得到的时、分、秒在屏幕上刷新显示，其代码如下：

```
--将计算出来新的时间刷新显示出来
vgus_vp_write(0x0001,3,time_table)
```

将变量 time_table 的第 1-3 个元素的数值写入到 0x0001-0x0003 地址中，倒计时的时间将对应在屏幕中实时显示出来。

以上就是实现倒计时功能的基本步骤，官网可以下载完整的案例工程，包括 Lua 脚本代码和界面工程。控件的使用说明、寄存器功能说明都可以参考文档《VGUS 串口屏用户开发指南》。Lua 脚本函数可以参考文档《基于 VGUS 的 Lua 脚本使用说明。