R&R Report Writer *Technical Bulletin*

Time Arithmetic in R&R Report Writer® for DOS®

Product: R&R Report Writer for DOS **Oper Sys:** DOS
Version: N.A.

Summary

Time arithmetic is useful in applications that need to measure the duration of events. Time arithmetic is used in this example to analyze a support call history. The results help in planning and budgeting support services to maintain a 100% call service rate.

When you call for support, your registration is retrieved using the serial number as a key. When the call ends, the starting and ending times are recorded in a database. Several interesting statistics can be produced using this data, including total call duration, average call length, and average time between calls.

Wall Data Support Rep Time Analysis
June 3, 1998

Start	End	Hours	Minutes	Sec's	Duration	Lag Time
10:02:40	10:26:16	0.39	23.60	1416	00:23:36	
10:40:42	10:59:11	0.31	18.48	1109	00:18:29	00:14:26
11:08:15	11:10:53	0.04	2.63	158	00:02:38	00:09:04
11:25:17	11:38:50	0.23	13.55	813	00:13:33	00:14:24
13:32:36	13:34:37	0.03	2.02	121	00:02:01	01:53:46
14:38:28	14:44:51	0.11	6.38	383	00:06:23	01:03:51
14:54:46	15:05:16	0.18	10.50	630	00:10:30	00:09:55
15:35:44	15:43:25	0.13	7.68	461	00:07:41	00:30:28
15:46:30	15:50:28	0.07	3.97	238	00:03:58	00:03:05
15:51:32	15:52:27	0.02	0.92	55	00:00:55	00:01:04
15:52:57	15:53:53	0.02	0.93	56	00:00:56	00:00:30
16:07:00	16:10:52	0.06	3.87	232	00:03:52	00:13:07
16:32:06	17:10:21	0.64	38.25	2295	00:38:15	00:21:14
========	========	======	=======	=====	========	========
10:02 AM	5:10 PM	2.21	132.78	7967	02:12:47	04:54:54

First call started at 10:02 AM
Last call ended at 5:10 PM
Total time on phone 02:12:47 (2.21 hours, 132.78 minutes, 7967 seconds)
Average call duration 10.21 minutes
Average time between calls 21.23 minutes
This support person is semi-retired!

Figure 1. Time Analysis Report

Producing a Report Using Time Arithmetic

Figure 1 shows an example of the type of report you can produce using time arithmetic. All you need to produce this type of report is a database with two fields, one containing the starting time and one containing the ending time. In R&R Report Writer, time values are represented as character strings in the format HH:MM:SS.

Seven user-defined functions (UDFs) were used to perform the calculations in this report. We'll discuss them individually as they apply to fields on the report.

The first UDF is named AmPm() and is used to format the starting time of the first call and the ending time of the last call. Note two features of the AmPm() function: the seconds are omitted and hours after 12 noon are converted to twelve-hour format. Here is the function definition.

```
Declaration:  AmPm(c_tstr)
Formula:  STR(IIF(MOD(VAL(SUBSTR(tstr,1,2)),12),
                  MOD(VAL(SUBSTR(tstr,1,2)),12),12),2)+
                  SUBSTR(tstr,3,3)+' '+
                  IIF(VAL(SUBSTR(tstr,1,2))<12,"AM","PM")
```

The AmPm() function takes one input, a time string, which is named tstr in the formula. The formula is printed on seven lines for readability. The first five lines convert the hour to a two-digit numeric string between 1 and 12 using the MOD() function to convert the hour to a number between 0 and 11, and the IIF() function to change 0 to 12. The sixth line appends a colon, the two digit minute, and a space. The last line appends "AM" or "PM" depending on the hour.

The AmPm() function is used in the sample report to format the earliest call starting time and the latest call ending time. So, we first need to calculate these times before applying the AmPm() function. The /Field Total Create command is used to calculate these times as follows.

```
Frst_Start = Grand Minimum of START_TIME
Last_End       = Grand Maximum of END_TIME
```

Next, two calculated fields are created to apply the AmPm() function to the above fields. Here are the calculated field definitions for the fields that appear on the report.

```
AmPm_Start = AmPm(Frst_Start)
AmPm_End     = AmPm(Last_End)
```

The middle three columns on the report calculate the number of hours, minutes, and seconds between the start and end times. Both hours and minutes are displayed with a two-digit fraction. (Keep in mind that 0.50 hours represents 30 minutes, not 50 minutes.) To calculate these time differences, it is easiest to convert both time values to common units—the number of seconds

since the start of the day. So, we first create a UDF named TimTos() to convert a time string to seconds.

```
Declaration:  TimTos(c_tstr)
Formula:  VAL(LEFT(tstr,2))*3600+VAL(SUBSTR(tstr,4,2))*60+
              VAL(RIGHT(tstr,2))
```

The formula appears on three lines for readability. The first line multiplies the value of the hour portion of the string by 3600, the number of seconds per hour. The second line multiplies the value of the minutes string by 60, the number of minutes per hour. The last line just computes the value of the seconds string. All three seconds values are then added together to give the result.

Now that we can convert a time string to a number of seconds, we can create UDFs to compute a time difference in hours, minutes, and seconds. Here are the three UDF definitions.

```
Declaration:  HrsBtwn(c_tstr1,c_tstr2)
Formula:  (TimTos(tstr2)-TimTos(tstr1))/3600

Declaration:  MinBtwn(c_tstr1,c_tstr2)
Formula:  (TimTos(tstr2)-TimTos(tstr1))/60

Declaration:  SecBtwn(c_tstr1,c_tstr2)
Formula:  TimTos(tstr2)-TimTos(tstr1)
```

To calculate the total time spent on the phone, in either hours, minutes, or seconds, you need to create a calculated field and a total field. The calculated field contains the call duration in the desired units for each call record. The total accumulates the value from each record. Therefore, create the following three pairs of calculated and total fields.

```
Hours     = HrsBtwn(START_TIME,END_TIME)
Tot_Hours = Grand Sum of Hours

Minutes = MinBtwn(START_TIME,END_TIME)
Tot_Mins  = Grand Sum of Minutes

Seconds         = SecBtwn(START_TIME,END_TIME)
Tot_Secs  = Grand Sum of Seconds
```

Another way to express the duration of a phone call is using the time format. For example, instead of representing a call duration as 2.21 hours, it can be represented also as 02:12:47. As you may have guessed, this calculation requires another UDF, which is named TimBtwn().

The TimBtwn() function works by calculating a time difference in seconds and formatting the result in the normal time string format. To simplify the computation, we created a separate UDF, which is named TimStr(), to apply the time string format to a value representing the number of seconds since the beginning of the day. Here are the two UDF definitions.

```
Declaration:  TimStr(n_sec)
Formula:  STR(INT(sec/3600),2)+":"+STR(INT(MOD(sec,3600)/60),2)+":"+
                    STR(INT(MOD(MOD(sec,3600),60)),2)

Declaration:  TimBtwn(c_tstr1,c_tstr2)
Formula:  TimStr(TimTos(tstr2)-TimTos(tstr1))
```

It is now a simple step to create a calculated field that computes the time difference and formats it as a time string. In our sample report, the field is defined as follows:

```
Duration = TimBtwn(START_TIME,END_TIME)
```

If you try this calculation, you find that the resulting time string does not contain leading zeros. For example, instead of getting 01:02:03, you'll get 1: 2: 3.

This may be acceptable to you, but we will tell you how to add leading zeros in case you prefer leading zeros to spaces.

The StrReplace() UDF, which replaces all occurrences of one substring with another, can be used to replace all spaces in a time string with zeros. Here is a modified TimStr() function that includes leading zeros.

```
Declaration:  TimStr(n_sec)
Formula:  StrReplace(STR(INT(sec/3600),2)+":"+
                  STR(INT(MOD(sec,3600)/60),2)+":"+
                  STR(INT(MOD(MOD(sec,3600),60)),2),
                  ' ','0')
```

Try it again and you will see that the time string still doesn't have leading zeros—when we designed the StrReplace() function, we did not anticipate replacing spaces, so we TRIMed them off. However, the StrReplace() function can easily be modified to handle spaces correctly. For reference, here is the complete and corrected definition of the StrReplace() function and its companion _ssReplace() function.

```
Declaration:  _ssReplace(c_string,c_old,c_new)
Formula:  IIF(old$string,_ssReplace(STUFF(string,AT(old,string),
                    LEN(old),new),old,new),string)

Declaration:  StrReplace(c_string,c_old,c_new)
Formula:  IIF(old$TRIM(string),_ssReplace(TRIM(string),old,new),
                    string)
```

Now the Duration calculated field will correctly format the time string with leading zeros. The same format is applied to the total duration by creating a calculated field with the following definition.

```
Tot_Dur = TimStr(Tot_Secs)
```

This expression takes the total duration of all phone calls in seconds, which we previously calculated in the Tot_Secs field, and applies the time string format using the TimStr() function.

Now let's discuss lag time. In our example, lag time is defined as the time between the end of the previous call and the start of the next call. This calculation illustrates the use of the PREVIOUS() function. Here is the calculated field definition.

```
Lag_Time = TimBtwn(PREVIOUS(END_TIME),START_TIME)
```

Note that the first record on the report has no lag time because there is no previous call. This is accomplished by creating a count of phone calls by support rep, and changing the Lag_Time expression to return a blank string when the count is equal to 1. If the sample report were grouped by support rep, the following total field and modified Lag_Time expression would be used.

```
Rep_Count = REP Sub-Count of START_TIME
Lag_Time  = IIF(Rep_Count>1,TimBtwn(PREVIOUS(END_TIME),
                    START_TIME),"")
```

In order to calculate the average lag time, we must also calculate the lag time in seconds for each call. The following calculated field and total field together produce the average lag time in minutes.

```
Lag_Mins = MinBtwn(PREVIOUS(END_TIME), START_TIME)
Avg_Lag = Grand Average of Lag_Mins
```

```
Header                      Wall Data Support Rep Time Analysis
Header                              ^mmmmmmmm dd, yyyy
Header
Header   Start      End        Hours     Minutes     Sec's    Duration   Lag Time
Header
Body     'XXXXXXX   'XXXXXXX   ###.##    ####.##     #####    'XXXXXXX   'XXXXXXX
Summary  ========   ========   ======    =======     =====    ========   ========
Summary  'XXXXXXX   'XXXXXXX   ####.##   #####.##    ######   'XXXXXXX   'XXXXXXX
Summary
Summary  First call started at 'XXXXXXX
Summary  Last call ended at 'XXXXXXX
Summary  Total time on phone 'XXXXXXX (##.## hours, ###.## minutes, ####
         seconds)
Summary  Average call duration ####.## minutes
Summary  Average time between calls ####.## minutes
Summary  This support person is 'XXXXXXXXXXXXXX
```

Figure 2. Time Analysis Report Layout

In the summary area there are two other calculations we have not discussed: the average call time in minutes and a text message rating the customer support rep's performance for the day. The average call time in minutes is a straight-forward total defined as follows.

```
Avg_Dur = Grand Average of Minutes
```

The last line of the report illustrates how a message can be tailored based on the value in another field. Support reps that spend six hours or more per day on the phone are rated as "working hard". Those who spend four hours or more are rated as "hardly working"; those spending two hours or more are rated "semi-retired", as in our example, and reps who spend fewer than two hours on the phone are rated "useless." (The rep in this example isn't really that bad - June 3rd was just a slow day!) Here's the calculated field definition.

```
Rating = IIF(Tot_Hours>=6,"working hard",
                IIF(Tot_Hours>=4,"hardly working",
                        IIF(Tot_Hours>=2,"semi-retired",
                                "useless")))
```