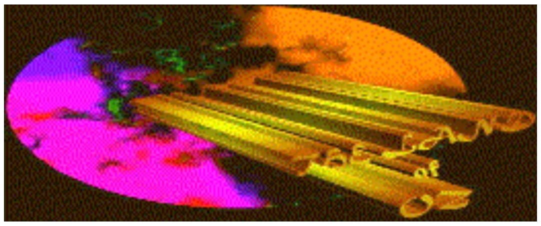
DAS_Tools For Windows
A Clarion For Windows ToolBox



General Information

ISBN 0-933735-00-6

Tin Man Software Corporation
P.O. Box 48823
Wichita, KS 67201-8823
(316) 264-3830
Homepage: http://www2.southwind.net/~tinman
@email: tinman@southwind.net
tinmanatoz@aol.com
74541.144@compuserve.com

Amierica Online: Tinmanatoz
Compuserve: 74541,144

Reichenberger Development Incorporated
Compuserve: 71324,1560

This is an updated version of a LIB we wrote for Clarion For Dos Version 3.
It has been updated for Clarion For Windows Version 1.5001 with new features added.

Introduction

DAS_Tools For Windows is a compilation of functions and procedures to assist the developer in handling the difficult tasks of creating applications using the Clarion For Windows environment. The Library is made up of both 16 and 32 bit compiled LIB files as well as Utility, Extension and Procedure Templates. In this version we have included Time, Date and String routines from the Clarion 3.1 version of DasTools. These functions were very popular in the DOS libraries and just make it very simple to manipulate date and time data, see "Included Routines" below for examples.

We also enhanced the package by adding a few of our favorite extensons templates that we thought you would enjoy. These include the DAS_Multi_Limiter Extension Template which limits the number of workstations in your EXE at one time. We also included the DAS_Demo Procedure Extension template which allows you to distribute a DEMO version of your product with very little effort on your part.

Our Application and Dictionary Documentation Utility templates are standard equipment in our toolbox. We felt that these were too essential to just be included as extras. All the information which is hidden inside the APP and DCT file can now be revealed and put to work for you while you are developing an APP or undertaking the difficult task of writting documentation.

The DASDEMO.EXE as well as the DASDEMO.APP are included, not only to instantly show off our products, but to be ripped apart, trashed and reinvented so as to demonstrate for you the implementation and ease of use of the DAS_Tools for Windows features.

We think you will be very happy with DAS_Tools For Windows. Our beta testers are rigorously testing all DAS_Tools For Windows features across Windows 3.1, Windows 3.11, WFWG 3.11, Windows 95 and Windows NT platforms. We feel very confident in the stability of our product as we release it to the public. Tinman Software Corp gaurantees your satisfaction with our products or your money back.

As always, we encourage any, and all, comments regarding our products. Should you have any ideas for a new feature you would like added to DAS_Tools For Windows, just let us know.

Adding DASTOOLS to the CW15 environment

REGISTER your templates in the CW15 environment.

1. Open CW15
2. Select Setup Menu
3. Select Template Registry
   (If you are updateing from an earlier version of DASTOOLS)
   - Scroll down and highlight CLASS DAS_ToolBox
   - Select the Unregister button
4. Select the Reigister Button
5. Go to the CW15\DASTOOLS directory
6. Double Click on the DASTOOLS.TPL file.

Adding DASTOOLS to your application

1. Open your application
2. From the Application Tree select Global
3. Select Extensions
4. Select Insert
5. Highlight DAS_ToolBox
6. Click the Select Button
7. Check if you need the ASCII file driver added to your Project file.
   (The ToolBox requires the ASCII File driver to be included in the Project, to prevent possible duplication errors at compile time, we have made it an option to have the ToolBox automatically add this driver to your project.)
8. Click OK

This is all there is to it.
Now you can start using any of the following functions or procedures.

----- Date Routines -----

Validate Date - DAS_DateValid

Validate the date passed to be a valid date (1/1/1801 - 12/31/2099).

Prototype:
  DAS_DateValid(LONG)

Parameters:

```
    ChkDate   = DAS_DateValid(Datein)

    Datein      = Long containing the date to check in Clarion
                    format.

    ChkDate   = Ushort to Return status the Clarion ERRORCODE() is
          also posted and may be checked.

          0 - Valid Date
          1 - Invalid Date

    Datein        = Date in Clarion format.
```

Example:

```
ChkDate   Ushort

  CODE

    ChkDate   = DAS_DateValid(Today())

    If DAS_DateValid(Today()) Then     !Date Valid
       .....Some Code      !No Do Something
    .
```

Get Day Of The Week - DAS_DayOfWk

Returns the day of the week for the date passed. Day is returned
as a string in long or short format. If the date passed is invalid
the Clarion ERRORCODE() is posted and a null string is returned.
If option is omitted long form is returned and if date is omitted
today's date will be used.

Prototype:
```
  DAS_DayOfWk(BYTE,LONG),STRING
```

Parameters:
```
    DayString = DAS_DayOfWk(Option,Datein)

    Option    = 0 - Return short form (Mon, Tues, Etc.)
                  1 - Return long form (Monday, Tuesday, Etc.)

    Datein    = Date in Clarion Format.

    DayString = String to return day.
```

Example:

```
Daystring STRING(9)
OtherDate Long

  CODE

  Daystring = DAS_DayOfWk(0,TODAY())   !Return short form
  Daystring = DAS_DayOfWk(1,TODAY())   !Return long form

  Daystring = DAS_DayOfWk(1,OtherDate) !Return long form
```

Get Day Of The Week Number - **DAS_DayOfWkn**

Returns the day of the week for the date passed. Day is returned as a ushort. If the date passed is invalid the Clarion ERRORCODE() is posted and a zero is returned. If date is omitted, todays date will be used.

Prototype:
```
  DAS_DayOfWkn(LONG),USHORT
```

Parameters:
```
    DayShort  = DAS_DayOfWkn(Datein)

    Datein    = Date in Clarion Format.

    DayShort  = Ushort to return day as follows.

           0 - Sunday
           1 - Monday
           2 - Tuesday
           3 - Wednesday
           4 - Thursday
           5 - Friday
           6 - Saturday
```

Example:

```
DayShort  USHORT
OtherDate Long

  CODE

  DayShort = DAS_DayOfWkn(TODAY())     !Return Day Number

  DayShort = DAS_DayOfWkn(OtherDate)   !Return Day Number

  If ERRORCODE() Then              !Error In Date
    .... Some Code             !Yes
  Else
    .... Some Code             !No
  .
```

Is Day A Day Of The Week - **DAS_IsaWkDay**

Checks to see if day of date passed is a week day. If day is a week day, the number of the week day is returned as a ushort. If the date passed is invalid the Clarion ERRORCODE() is posted and a zero is returned. If date is omitted today's date will be used.

Prototype:
```
  DAS_IsaWkDay(LONG),USHORT
```

Parameters:
```
    DayShort  = DAS_IsaWkDay(Datein)

    Datein    = Date in Clarion Format.

    DayShort  = Ushort to return status as follows.

              0 - Not a Week Day
              1 - Monday
              2 - Tuesday
              3 - Wednesday
              4 - Thursday
              5 - Friday
```

Example:

```
DayShort  USHORT
OtherDate Long

  CODE

  DayShort = DAS_IsaWkDay(OtherDate)   !Week Day

  If ERRORCODE() Then                  !Error In Date
    .... Some Code             !Yes
  Else
    .... Some Code             !No
  .
```

Is Year A Leap Year - **DAS_LeapYear**

Checks to see if year of date passed is a leap year. Status is returned as a ushort. If the date passed is invalid, the Clarion ERRORCODE() is posted and a zero is returned. If date is omitted, today's date will be used.

Prototype:
```
  DAS_LeapYear(LONG),USHORT
```

Parameters:
```
    DayShort  = DAS_LeapYear(Datein)

    Datein    = Date in Clarion Format.

    DayShort  = Ushort to return status as follows.

              0 - Not A Leap Year
              1 - A Leap Year
```

Example:

```
DayShort  USHORT
OtherDate Long

  CODE

  DayShort = DAS_LeapYear(TODAY())        !Leap Year

  DayShort = DAS_LeapYear(OtherDate))     !Leap Year

  If ERRORCODE() Then                   !Error In Date
    .... Some Code                      !Yes
  Else
    .... Some Code                      !No
  .
```

Is Date Last Day Of The Month - **DAS_IsLastDy**

Checks to see if the day of date passed is the last day of the
month. Status is returned as a ushort. If the date passed is
invalid the Clarion ERRORCODE() is posted and a zero is returned.
If date is omitted, today's date will be used.

Prototype:
  DAS_IsLastDy(LONG),USHORT

Parameters:
    DayShort = DAS_IsLastDy(Datein)

    Datein   = Date in Clarion Format.

    DayShort = Ushort to return status as follows.

      0 - Not A Last Day
      1 - Is Last Day

Example:

```
DayShort  USHORT
OtherDate Long

  CODE

  DayShort = DAS_IsLastDy(TODAY())        !Last Day

  DayShort = DAS_IsLastDy(OtherDate))     !Last Day

  If ERRORCODE() Then                   !Error In Date
    .... Some Code                      !Yes
  Else
    .... Some Code                      !No
  .
```

Get Number Of Days In Month - **DAS_DaysInMn**

Returns the days in the month for the date passed. Days is returned as a ushort. If the date passed is invalid the Clarion ERRORCODE() is posted and a zero is returned. If date is omitted, today's date will be used.

Prototype:
```
  DAS_DaysInMn(LONG),USHORT
```

Parameters:
```
    DayShort  = DAS_DaysInMn(Dateincl)

    Dateincl  = Date in Clarion Format.

    DayShort  = Ushort to return days.
```

Example:

```
DayShort  USHORT
OtherDate Long

  CODE

    DayShort = DAS_DaysInMn(TODAY())        !Return Days In Month

    DayShort = DAS_DaysInMn(OtherDate))     !Return Days In Month

    If ERRORCODE() Then           !Error In Date
         .... Some Code           !Yes
    Else
         .... Some Code           !No
    .
```

Get Day Of The Year Number - **DAS_DayOfYr**

Returns the day of the year for the date passed. Day is returned as a ushort. If the date passed is invalid the Clarion ERRORCODE() is posted and a zero is returned. If date is omitted, today's date will be used.

Prototype:
```
  DAS_DayOfYr(LONG),USHORT
```

Parameters:
```
    DayShort  = DAS_DayOfYr(Dateincl)

    Dateincl  = Date in Clarion Format.

    DayShort  = Ushort to return day.
```

Example:

```
DayShort  USHORT
```

```
OtherDate Long

  CODE

    DayShort = DAS_DayOfYr(TODAY())    !Return Day Number

    DayShort = DAS_DayOfYr(OtherDate)) !Return Day Number

    If ERRORCODE() Then           !Error In Date
         .... Some Code           !Yes
    Else
         .... Some Code           !No
    .
```

Get Week Of The Year Number - **DAS_WeekOfYr**

Returns the week of the year for the date passed. Week is returned as a ushort. If the date passed is invalid the Clarion ERRORCODE() is posted and a zero is returned. If date is omitted, today's date will be used.

Prototype:
```
  DAS_WeekOfYr(LONG),USHORT
```

Parameters:
```
  DayShort  = DAS_WeekOfYr(Datein)

  Datein    = Date in Clarion Format.

  DayShort  = Ushort to return week.
```

Example:

```
DayShort    USHORT
OtherDate   Long

  CODE

  DayShort = DAS_WeekOfYr(TODAY())        !Return Week Number

  DayShort = DAS_WeekOfYr(OtherDate)      !Return Week Number

  If ERRORCODE() Then                     !Error In Date
    .... Some Code                        !Yes
  Else
    .... Some Code                        !No
  .
```

Get Quarter Of The Year - **DAS_QtrOfYr**

Returns the quarter of the year for the date passed. Quarter is returned as a ushort. If the date passed is invalid the Clarion ERRORCODE() is posted and a zero is return. If date is omitted, today's date will be used.

Prototype:
```
 DAS_QtrOfYr(LONG),USHORT
```

Parameters:
```
   DayShort  = DAS_QtrOfYr(Datein)

   Datein    = Date in Clarion Format.

   DayShort  = Ushort to return quarter.
```

Example:

```
DayShort  USHORT
OtherDate Long

  CODE

  DayShort = DAS_QtrOfYr(TODAY())         !Return Quarter Number

  DayShort = DAS_QtrOfYr(OtherDate)       !Return Quarter Number

  If ERRORCODE() Then                     !Error In Date
    .... Some Code                !Yes
  Else
    .... Some Code                !No
  .
```

Get Days Left In The Year - **DAS_DaysLfYr**

Returns the days left in the year from the date passed. Days is returned as a ushort. If the date passed is invalid the Clarion ERRORCODE() is posted and a zero is returned. If date is omitted, today's date will be used.

Prototype:
```
 DAS_DaysLfYr(LONG),USHORT
```

Parameters:
```
   DayShort  = DAS_DaysLfYr(Dateincl)

   Dateincl  = Date in Clarion Format.

   DayShort  = Ushort to return days.
```

Example:

```
DayShort  USHORT
OtherDate Long

  CODE
  DayShort = DAS_DaysLfYr(TODAY())     !Return Days

  DayShort = DAS_DaysLfYr(OtherDate)   !Return Days

  If ERRORCODE() Then            !Error In Date
    .... Some Code               !Yes
  Else
    .... Some Code               !No
  .
```

Get Formatted Date - **DAS_FmatDate**

Returns a formatted date for the date passed. Date is returned as a string in long or short format. If the date passed is invalid the Clarion ERRORCODE() is posted and a null string is returned. If option is omitted long form is returned and if date is omitted, todays date will be used.

Prototype:

```
  DAS_FmatDate(BYTE,LONG),STRING
```

Parameters:

```
    DateStr   = DAS_FmatDate(Option,Datein)

    Option    = 0 - Return short form (Mon Jan 1,1980)
                1 - Return long form (Monday August 15,1980)

    Datein    = Date in Clarion Format.

    DateStr   = String to return date.
```

Example:

```
DateStr   STRING(30)
OtherDate Long

  CODE

  DateStr = DAS_FmatDate(0,TODAY())    !Return short form

  DateStr = DAS_FmatDate(1,TODAY())    !Return long form

  DateStr = DAS_FmatDate(1,OtherDate)    !Return long form
```

Get Month Of Year - **DAS_MonOfYr**

Returns the month of the year for the date passed. Month is returned as a string in long or short format. If the date passed is invalid the Clarion ERRORCODE() is posted and a null string is returned. If option is omitted long form is returned and if date is omitted, today's date will be used.

Prototype:

```
  DAS_MonOfYr(BYTE,LONG),STRING

 Parameters:
   MonthStr  = DAS_MonOfYr(Option,Datein)

   Option    = 0 - Return short form (Jan, Feb, etc.)
               1 - Return long form (June, August, etc.)

   Datein    = Date in Clarion Format.

   MonthStr  = String to return Month

 Example:

MonthStr  String(10)
OthMonth  Long

  CODE

  MonthStr = DAS_MonOfYr(0,MONTH(TODAY())) !Return short form

  MonthStr = DAS_MonOfYr(1,MONTH(TODAY())) !Return long form

  MonthStr = DAS_MonOfYr(1,OtherMonth) !Return long form
```

Get Julian Date - **DAS_JulDate**

Returns a string containing the Julian date of the date passed.
Julian date is returned as 'YY.DDD' If the date passed is invalid
the Clarion ERRORCODE() is posted and a null string is returned. If
option is omitted long form is returned and if date is omitted,
today's date will be used.

```
 Prototype:
  DAS_JulDate(LONG),STRING

 Parameters:
   JulString = DAS_JulDate(Datein)

   Datein    = Date in Clarion format.

   JulString = String to Return Julian Date

 Example:

JulString STRING(6)
OtherDate Long

  CODE

  JulString = DAS_JulDate(TODAY())

  JulString = DAS_JulDate(OtherDate)
```

Find Age Of Dates - DAS_FindAge

Returns a string containing the elapsed time between the dates passed as ' xx Years, x Months xx Days'.

Prototype:
```
  DAS_FindAge(LONG,LONG),STRING
```

Parameters:
```
    DayString = DAS_FindAge(LowDate,HighDate)

    LowDate   = Start date in Clarion date format.

    HighDate  = End date in Clarion date format.

    DayString = String to return age.
```

Example:

```
DayString STRING(35)
LowDate   Long
HighDate  Long

  CODE

  Daystring = DAS_FindAge(LowDate,HighDate)

  Daystring = DAS_FindAge(LowDate,ToDay())
```

Convert Clarion Date To DOS Format - DAS_CDateDos

Returns a ushort containing the Date in DOS format converted from Clarion format. If the date passed is invalid the Clarion ERRORCODE() is posted and zero is returned. If date is omitted, today's date will be used.

Prototype:
```
  DAS_CDateDos(Date),USHORT
```

Parameters:
```
   DosShort  = DAS_CDateDos(ClaDate)

   ClaDate   = Long(Date) to convert.

   DosShort  = Ushort containing the Date in DOS file format.
```

Example:

```
ClaDate   Date
DosShort  USHORT

   CODE
```

```
  DosShort  = DAS_CDateDos(ClaDate)
```

Convert DOS Date Format To Clarion - **DAS_DosDateC**

Returns a long(Date) containing the DOS Date converted to Clarion format.

Prototype:

```
  DAS_DosDateC(USHORT),Date
```

Parameters:

```
   ClaDate   = DAS_DosDateC(DosShort)

   ClaDate   = Long(Date) to return convert Date.

   DosShort  = Ushort containing the DOS Date.
```

Example:

```
ClaDate   Date
DosShort  USHORT

  CODE

  ClaDate = DAS_DosDateC(DosShort)
```

Get Clarion Date - **DAS_Date**

Returns the date in Clarion format from the month, day and year passed.

Prototype:

```
  DAS_Date(BYTE,BYTE,<SHORT>),LONG
```

Parameters:

```
  ClaDate   = DAS_Date(Monthin,Dayin,Yearin)

  ClaDate   = Long to return date in Clarion format.

  Monthin   = Month of year.

  Dayin     = Day of month.

  Yearin    = Year, if omitted the current year is use.
```

Example:

```
 ClaDate    Long

  CODE

  ClaDate   = DAS_Date(11,01,1990)       !Get Date
```

Find Work Days - DAS_Workdays

Returns the number of working days between dates.

Prototype:
```
  DAS_WORKDAYS(LONG,LONG,<BYTE>,<BYTE>,<BYTE>,<BYTE>,<BYTE>, |
            <BYTE>,<BYTE>),LONG
```

Parameters:
```
  WORKDAYS  = DAS_WORKDAYS(LOWDATE,HIGHDATE,SAT,SUN,MON,TUE,WED,THU,FRI)

  LOWDATE   = Start date in Clarion date format.

  HIGHDATE  = End date in Clarion date format.

  SAT - FRI = Set each day to work day or not. If ommited then that
              day is set to a work day.  1 - Yes / 0 - No

  WORKDAYS  = Long to return number of days.
```

Example:

```
WORKDAYS    LONG
LOWDATE     LONG
HIGHDATE    LONG

  CODE

  WORKDAYS  = DAS_WORKDAYS(LOWDATE,TODAY(),0,0)     !FIND WORK DAYS
                                                         !SAT & SUN NON WORK
!DAYS

  WORKDAYS  = DAS_WORKDAYS(LOWDATE,TODAY())         !ALL DAYS WORK DAYS
```

Get Fiscal Quarter Of The Year - DAS_Fiscalqtr

Returns the quarter of the year for the date passed based on the start of the fiscal year passed.

Prototype:
```
  DAS_FISCALQTR(LONG,<LONG>),LONG
```

Parameters:
```
  FISCALQT  = DAS_FISCALQTR(DATEIN,FISCALSRT)

  FISCALQT  = Number of quarter.

  DATEIN    = Date in Clarion Format.

  FISCALSRT = Start of fiscal year in clarion date format, if
              omitted start of current year will used.
```

Example:

```
FISCALQT    LONG

  CODE

  FISCALQT = DAS_FISCALQTR(TODAY(),DATE(03,01,YEAR(TODAY()))
```

Find Busniess Days - DAS_Busdays

Returns a string containing the nubmer of busniess days between
the two dates.

Prototype:
```
  DAS_BUSDAYS(LONG,<LONG>),LONG
```

Parameters:
```
  BUSDAYS   = DAS_BUSDAYS(LOWDATE,HIGHDATE)

  LOWDATE   = Start date in Clarion date format.

  HIGHDATE  = End date in Clarion date format, if omitted today is used.

  BUSDAYS   = Long to return number of days.
```

Example:

```
BUSDAYS       LONG
LOWDATE      LONG
HUGHDATE    LONG

  CODE

  BusDays   = DAS_BusDays(LowDate,HighDate)

  BusDays   = DAS_BusDays(LowDate,ToDay())
```

Find Next Busniess Day - DAS_Nextbday

Returns the date of the next business day at start date plus
number of days.

Prototype:
```
  DAS_NEXTBDAY(<LONG>,<LONG>),LONG
```

Parameters:
```
  BUSDATE   = DAS_NEXTBDAY(STRDATE,NUMDAYS)

  STRDATE   = Start date in Clarion date format, if omitted
                   todays date will be used.

  NUMDAYS   = Number of days ahead, if omitted 1 will be used.

  BUSDATE   = Date of next business day in clarion format.
```

Example:

```
BUSDATE     LONG

  CODE

  BUSDATE   = DAS_NEXTBDAY(TODAY(),15)     !FIND NEXT BUSINESS DAY 15
                                             !DAYS FROM NOW.
```

Find Number Of Days - **DAS_Numdays**

Returns the total numbers of days between two dates.

Prototype:
```
  DAS_NUMDAYS(LONG,<LONG>),LONG
```

Parameters:
```
  NUMDAYS   = DAS_NUMDAYS(LOWDATE,HIGHDATE)

  LOWDATE   = Start date in Clarion date format.

  HIGHDATE  = End date in Clarion date format, if omitted today is used.
  NUMDAYS   = Long to return number of days.
```

Example:

```
NUMDAYS     LONG
LOWDATE     LONG
HIGHDATE    LONG

  CODE

  NUMDAYS   = DAS_NUMDAYS(LOWDATE,HIDATE)
```

----- Time Routines -----

Delay Program - **DAS_Delay**

Cause a delay in the program for ?? seconds.

Prototype:
```
  DAS_Delay()
```

Parameters:
```
  DAS_Delay(Secs)

    Secs = Number of seconds 1 - 60.
```

Example:

```
  CODE
  DAS_Delay(5)                !Delay for 5 seconds

  DAS_Delay(20)               !Delay for 20 seconds
```

Find Time Between Dates And Time - **DAS_FindTime**

Returns a string containing the elapsed time between the dates and time passed as 'xx Hours, xx Minutes xx Seconds'.

Prototype:
```
 DAS_FindTime(LONG,LONG,LONG,LONG),STRING
```

Parameters:
```
   TimString = DAS_FindTime(LowDate,LowTime,HighDate,HighTime)

   LowDate   = Start date in Clarion date format.

   LowTime   = Start time in Clarion time format.

   HighDate  = End date in Clarion date format.

   HighTime  = End time in Clarion time format.

   TimString = String to return time.
```

Example:

```
TimString STRING(35)
LowDate   Long
LowTime   Long
HighDate  Long
HighTime  Long

 CODE

 TimString = DAS_FindTime(LowDate,LowTime,HighDate,HighTime)

 TimString = DAS_FindTime(LowDate,LowTime,TODAY(),CLOCK())
```

Validate Time - **DAS_TimeValid**

Validate the time passed to be a valid time (Midnight - 23:59:59).

Prototype:
```
 DAS_TimeValid(LONG)
```

Parameters:
```
   ChkTime   = DAS_TimeValid(Timein)

   Timein    = Long containing the time to check in Clarion
          format.

   ChkTime   = Ushort to Return status the Clarion ERRORCODE() is
```

also posted and may be checked.

    0 - Valid Time
    1 - Invalid Time

Timein = Time in Clarion format.

Example:

```
Timein    LONG
ChkTime   Ushort

  CODE

     Chk_Time = DAS_TimeValid(Timein)

     If DAS_TimeValid(Timein) Then         !Time Valid
       .....Some Code                      !No Do Something
     .
```

Convert Clarion Time To DOS Format - **DAS_CTimeDos**

Returns a ushort containing the Time in DOS format converted from Clarion format. If the time passed is invalid the Clarion ERRORCODE() is posted and zero is returned. If time is omitted, the current time will be used.

Prototype:

```
  DAS_CTimeDos(LONG),USHORT
```

Parameters:

```
   DosShort  = DAS_CTimeDos(ClaTime)

   Clatime   = Time in Clarion format.

   DosShort  = Ushort containing the time in DOS file format.
```

Example:

```
ClaTime   TIME
DosShort  USHORT

   CODE

   DosShort  = DAS_CTimeDos(ClaTime)
```

Convert DOS Time Format To Clarion - **DAS_DosTimeC**

Returns a long(time) containing the DOS time converted to Clarion format.

Prototype:

```
  DAS_DosTimeC(USHORT),TIME
```

Parameters:
  ClaTime   = DAS_DosTimeCl(DosShort)

  Clatime   = Long(time) to return convert time.

  DosShort = Ushort containing the DOS time.

Example:

```
ClaTime   TIME
DosShort  USHORT

  CODE

  ClaTime = DAS_DosTimeC(DosShort)
```

Get Clarion Time - DAS_Time

Returns the time in Clarion format from the hours, minutes, seconds and hundredths of seconds passed.

Prototype:
  DAS_Time(BYTE,<BYTE>,<BYTE>,<BYTE>),LONG

Parameters:
  ClaTime   = DAS_Time(Hrsin,Minsin,Secsin,Hundin)

  ClaTime   = Long to return time in Clarion format.

  Hrsin   = Hours.

  Minsin   = Minutes.

  Secsin   = Seconds.

  Hundin   = Hundredths of seconds.

Example:

```
ClaTime     Long

  CODE

  ClaTime   = DAS_Time(11,01,19)          !Get Time
```

Is Time AM - DAS_IsTimeAM

Checks to see if time passed is in the AM.

Prototype:
  DAS_IsTimeAM(LONG),USHORT

```
Parameters:
  AMShort   = DAS_IsTimeAM(Timein)

  Timein    = Time in Clarion Format.

  AMShort   = Ushort to return status as follows.
              0 - Not AM
              1 - AM
```

```
Example:

  CODE

  If DAS_IsTimeAM(Clock()) Then          !Time in AM
    .......Some Code
  .

  If ERRORCODE() Then                    !Error In Time
    .... Some Code               !Yes
  Else
    .... Some Code               !No
  .
```

Is Time PM - DAS_IsTimePM

Checks to see if time passed is in the PM.

```
Prototype:
  DAS_IsTimePM(LONG),USHORT

Parameters:
  PMShort   = DAS_IsTimePM(Timein)

  Timein    = Time in Clarion Format.

  PMShort   = Ushort to return status as follows.
              0 - Not PM
              1 - PM
```

```
Example:

  CODE

  If DAS_IsTimePM(Clock()) Then          !Time in PM
    .......Some Code
  .

  If ERRORCODE() Then                    !Error In Time
    .... Some Code               !Yes
  Else
    .... Some Code               !No

  .
```

Get Hours - DAS_Hours

Returns the hours part of the time passed.

Prototype:
```
DAS_HOURS(LONG),LONG
```

Parameters:
```
Hours     = DAS_HOURS(Timein)

Hours     = The hours part of the time.

Timein    = Time in Clarion time format.
```

Example:

```
CODE

If DAS_HOURS(Clock()) = 12            !Is it noon
  ...... some code
End
```

Get Minutes - DAS_Minutes

Returns the minutes part of the time passed.

Prototype:
```
DAS_MINUTES(LONG),LONG
```

Parameters:
```
Minutes   = DAS_MINUTES(Timein)

Minutes   = The minutes part of the time.

Timein    = Time in Clarion time format.
```

Example:

```
CODE

If DAS_MINUTES(Clock()) = 5                        !If 5 minutes passed
  ...... some code
End
```

Get Seconds - DAS_Seconds

Returns the seconds part of the time passed.

Prototype:
```
DAS_SECONDS(LONG),LONG
```

Parameters:
```
Seconds   = DAS_SECONDS(Timein)
```

Seconds = The seconds part of the time.

Timein = Time in Clarion time format.

Example:

```
  CODE

  If DAS_SECONDS(Clock()) = 15            !If 15 seconds passed
    ...... some code
  End
```

Get Hundreds Of Seconds - DAS_Hundreds

Returns the hundreds of seconds part of the time passed.

Prototype:
  DAS_HUNDREDS(LONG),LONG

Parameters:
  Seconds = DAS_HUNDREDS(Timein)

  Seconds = The hundreds of seconds part of the time.

  Timein = Time in Clarion time format.

Example:

```
Hseconds        Long

  CODE

  Hseconds = DAS_HUNDREDS(CLOCK))
```

Get Time Elapsed In Hundreds Of Seconds - DAS_Elapsed

Returns the elasped time in hundreds of seconds.

Prototype:
  DAS_ELAPSED(LONG,LONG,<LONG>,<LONG>),LONG

Parameters:
  ELAPSED = DAS_ELAPSED(LOWDATE,LOWTIME,HIDATE,HITIME)

  ELAPSED = Elasped time in hundreds of seconds.

  LOWDATE = Staring date.

  LOWTIME = Staring time.

  HIDATE = Ending date, if omitted today date will be used.

HITIME   = Ending time, if ommited current time will be used.

Example:

```
LOWDATE     LONG
LOWTIME     LONG
ELAPSED     LONG

  CODE

  ELAPSED = DAS_ELAPSED(LOWDATE,LOWTIME,TODAY(),CLOCK())
```

Delay Program Untill Date & Time - DAS_Waituntil

Cause a delay in the program untill the date and time passed
is reached. with option to display message, and return
if key pressed.

Prototype:
```
  DAS_WAITUNTIL(LONG,LONG,<BYTE>,<BYTE>,<STRING>),LONG
```

Parameters:
```
   DAS_WAITUNTIL(DATEIN,TIMEIN,SHOWCNT,KEYPRESS,MSGOUT)
```

DATEIN   = Date to wait untill.

TIMEIN   = Time of day to wait untill.

SHOWCNT = True show count screen.

KEYPRESS = True return status of key press

MSGOUT   = Optional message to display max 40 bytes long,
also showcnt must be ture for message to be
displayed.

Example:

```
  CODE

  DAS_WAITUNTIL(TODAY()+1,CLOCK(),1,1)        !WAIT TILL NEXT DAY SAME
                                                  !TIME
```

<u>----- String Routines -----</u>

Capitalized First Letter Of Each Word - DAS_CapsStr

Returns a string containing the first letter of each word in the
string passed capitalized.

Prototype:
```
  DAS_CapsStr(STRING,<STRING>),STRING
```

Parameters:
  CapString = DAS_CapsStr(Stringin,Exception)

  Stringin = The string to capitalized.

  Exception = A string containing words not to capitalized.
  Ex. (and of to the you) this parameter is
  optional.

  CapsStrString = String to Return Capitalized String.

Example:

```
CapString STRING(80)
Exception STRING('and of the to too is')

  CODE

  CapString =    DAS_CapsStr('this is a test of a string being
                 capitalized')

  CapString =    DAS_CapsStr('test of a string being
                 capitalized',Exception)
```

Return First and Last Name - **DAS_Parse**

Parses out the Salute into First and Last names. Name is passed back as a string. If Type is omitted, First_Name is returned.

Prototype:
  DAS_Parse(String,Byte),String

Parameters:
  First_Name = DAS_Parse(Salute,Type)

  Salute = Field containing Salute.

  Type = 0 - Returns First_Name from Salute.
  = 1 - Returns Last_Name from Salute.

  First_Name = First Name stripped of salutation (ie..Mr., Mrs, etc)

  Last_Name = Last Name

Example:

```
First_Name        String(20)
Last_Name         String(20)

 CODE
 First_Name     = DAS_Parse('Mr. Gary L. Reichenberger',0)     ! Returns 'Gary L.'
 Last_Name      = DAS_Parse('Mr. Gary L. Reichenberger',1)     ! Returns 'Reichenberger'
```

Return Salute - DAS_Salute

Return a proper Salute from the strings passed.

Prototype:
DAS_Salute(String,String,String),String

Parameters:
Salute = DAS_Salute(First,Init,Last)

First	= Variable containing first name
Init	= Variable containing middle initial
Last	= Variable containing last name
Salute	= String to return salute.

Example:

```
Salute      String(45)
First       String(20)
Init        String(5)
Last        String(20)

 CODE
 Salute = DAS_Salute('Gary','L','Reichenberger') ! Returns 'Gary L. Reichenberger'

 Salute = DAS_Salute('Gary',,'Reichenberger') ! Returns 'Gary Reichenberger'

 First    = 'Gary'
 Init     = 'L'
 Last     = 'Reichenberger'
 Salute = DAS_Salute(First,Init,Last) ! Returns 'Gary L. Reichenberger'
```

----- Message Routine -----

Display a Message Box - DAS_MsgFunc

Displays a message box with passed parameters.

Prototype:
DAS_MsgFunc(<String>,String,<String>,<String>,<Byte>,<Byte>),Byte

Parameters:
CallFunc = DAS_MsgFunc(Title2,Message1,Message2,Message3,YesNo,Beepit)

CallFunc	= Byte to return
Title2	= Text to appear in Title Bar of Window
Message1	= Message Line 1

Message2	=	Message Line 2
Message3	=	Message Line 3
YesNo	=	Select Button Choice 0 = Yes and No 1 = Continue and Halt 2 = Go and Stop
Beepit	=	Select Bell Choice 0 = No Bell 1 = Bell

Example:

```
CallFunc         BYTE
Title2            STRING(50)
Message1         STRING(50)
Message2         STRING(50)
Message3         STRING(50)
YesNo            BYTE
BeepIt           BYTE

 CODE
 Title2          = 'Message Box'
 Message1        = 'Message Text Line 1'
 Message2        = 'Message Text Line 2'
 Message3        = 'Message Text Line 3'
 YesNo           = 1
 Beepit          = 1

 CallFunc = DAS_MsgFunc(Title2,Message1,Message2,Message3,YesNo,Beepit)
```

This displays a message box with the 'Continue' and 'Halt' buttons and the Bell will sound.

Returns 0 if 'Halt' button is pressed
Returns 1 if 'Continue Button is pressed

----- Warning Routine -----

Display a Warning Box - DAS_Warning

Displays a warning box with passed parameters.

Prototype:

```
DAS_Warning(<String>,<String>,<String>,<Byte>)
```

Parameters:

```
DAS_Warning(Message1,Message2,Message3,Beepit)
```

Message1	=	Message Line 1

```
            Message2      =  Message Line 2

            Message3      =  Message Line 3

            Beepit          =  Select Bell Choice
                               0 = No Bell
                               1 = Bell
```

Example:

```
Message1          STRING(50)
Message2          STRING(50)
Message3          STRING(50)
BeepIt            BYTE

 CODE
 Message1         = 'Message Text Line 1'
 Message2         = 'Message Text Line 2'
 Message3         = 'Message Text Line 3'
 Beepit           = 1

 DAS_Warning(Message1,Message2,Message3,Beepit)
```

This just displays a warning box with the 'Close' button and the Bell will sound. Nothing is returned.

----- Error Save Routine -----

Traps Error Information to ERROR.LOG file - DAS_ErrorSave

Gathers information on error and appends to the ERROR.LOG file.

Prototype:

```
 DAS_ErrorSave(String,String,Short,String)
```

Paramaters:

```
 DAS_ErrorSave(ErrFile,ErrName,ErrNum,Memo)
       ErrFile     = ErrorFile
       ErrName  =  Error()
       ErrNum    =  ErrorCode()
       Memo      =  User defined Memo
```

Example:

```
 CODE
 DAS_ErrorSave('Procedure1', Error(), Errorcode(), 'Error Updateing Form')
```

Output: to ERROR.LOG file

```
              DATE                Today()
              TIME                CLOCK()
              USER                DAS::USERNAME
              ERRORCODE()    ERRNUM
```

ERROR()	ERRNAME
ERRORFILE	Procedure1
PROGRAM	DAS::PGMNAME
LOCATION	DAS::PROCNAME
USER MEMO	Error Updateing Form

----- Utility Templates -----

An assortment of Utilities which make your job a little easier.

To use any of the following Utility Templates, follow these steps:
(Remember...You must Register the DAS_Tools Templates to the CW15 Environment)

1. Load your APP into the CW15 environment.
2. View the Application Tree.
3. Press 'Ctrl-U', This is the HOT KEY to the 'Select Utility' Menu
4. Double Click any of the Utility Templates under 'Class DAS_Tools'
5. To view the files produced by these templates, simply load the

DAS_APPDoc - DAS_Application Document Utility.

(Export Application Information to [Application Name].TIA file)

Creates a complete listing of all Controls placed in every Procedure. Very neatly organized and includes control types, control use variables and any TIP, HLP or MSG information available which would help to identify it.

DAS_DicDoc - DAS_Dictionary Document Utility.

(Export Dictionary Information to [Dictionary Name].TID file)

Creates a very useful Dictionary Documentation file which can be quickly inserted into any .CLW file as source code without any editing. Includes File and Key information as well as Relationship Information.

DAS_ProcCallTree - Produces a Procedure Calling Tree.

(Export Procedure Calling Tree List to [Application Name].TRE file)

Creates a very simply Application Calling Tree. Nothing real elaborate yet.

----- Extension Templates -----

DAS_Multi_Limiter (Application Extension Template)

The DAS_Multi_Limiter Extension restricts the number of users in your APP at one time.

We designed this Template to enable us to distribute single or multiple user applications to our clients. You simply enter the Maximum Users you require in the 'Maximum Users' entry field and specify the unique file which will be used by the DAS_Multi_Limiter code. At runtime the APP will automatically log each user into the system and out of the system when they exit the program. Once the Maximum User Count has been reached, new users will not be allowed into the APP until another user exits the APP.

To add DAS_Multi_Limiter to your APP simply follow the directions below..

1. Load your APP into the CW15 Environment.
2. View the Application Tree
3. Click on the 'Global' Button
4. Click on the 'Extensions' Button
5. Click on the 'Insert' button
6. Highlight the DAS_Multi_Limiter Template under the 'Class DAS_Tools' area.
7. Click on the 'Select' Button

To activate the DAS_Multi_Limiter Feature

1. Enter the number of Maximum Users you require in the 'Maximum Users' entry field.
   (If you leave the 'Maximum Users' set to 0, No limitation code will be generated.)
2. Optionally specify a unique file name for DAS_Multi_Limiter to use for this application.
3. Check if you need the Topspeed file driver added to your Project file.
   (The ToolBox requires the Topspeed File driver to be included in the Project, to prevent possible duplication errors at compile time, we have made it an option to have the ToolBox automatically add this driver to your project.)

DAS_Demo (Procedure Extension Template)

The DAS_Demo Extension allows the developer to easily distribute a demonstration version of any APP. Simply add the Extension Template to any Procedure which already has the Standard BrowseUpdateButtons Control placed in the window and fill in the blanks and you are finished.

We took a little different approach when we sat down and designed the DAS_Demo Template. We thought the developers would be proud of their work and would want to continue showing it off even after the Demonstration Period was over. So we came up with a Procedure Extension Template which is only called when the INSERT button is pressed. This essentially allows the enduser full access to your program even after the maximum record count has been met. They just can't add any more records until they delete some.

Our approach allows the developer to place the DAS_Demo template anywhere it is appropriate in his APP and as often as the developer requires. For example... you may want to limit the ZIPCODE database to only 100 records, the STATE database to only 20 records and the INVOICE database to only 10 records. Simply add the template to the appropriate Browse Procedures and enter the unique Datafile and Maximum Record Count which applies to each individual Procedure. I think you will like our approach.

To add DAS_Demo to your APP, simply follow the directions below...

1. Load your APP into the CW15 Environment.
2. View the Application Tree
3. Highlight the Procedure which you would like to add the DAS_Demo code to.
4. Click on the 'Properties' Button
5. Click on the 'Extensions' Button
6. Click on the 'Insert' Button
7. Highlight the DAS_Demo Template under the 'Class DAS_Tools' area.
8. Click on the 'Select' Button

To activate the DAS_Demo Feature

1. Enter the 'Datafile to Check:'
   (If the contents of the Datafile to Check is empty, the feature will be disabled.)

2. Enter the 'Maximum Records Allowed' in this Datafile to Check.
3. Optionally enter the Message you want displayed when the Demonstration period is over.
4. Click on the 'OK' Button

DAS_VarFilePath (Application Extension Template)

DAS_VarFlePath simplifies the use of Global Variables for File Names and File Paths. Just fill in the blanks and write the Procedure or Routine (if you specify this method) to define the Path Variable you specify. This Procedure you create may be as simple as getting the path from a datafile or as unique as creating a point and click procedure to define the Path Variable.

File Information:
Enter a variable for the Path Variable which will be prepended to the data file name or names which you associate with this variable.

Procedure or Routine to Call:
Choices are:

Procedure....This calls a Procedure which you create to preset the path variable prior to defining each file 'NAME'.
Routine........This calls a DO Routine which you create to preset the path variable prior to defining each file 'NAME'.
.INI File.........Pulls the path variable from the *default*.INI file at the section/entry specified.

.INI File:
Specify the section/entry to search for the file path in *default*.INI file.

To add DAS_VarFilePath to your APP simply follow the directions below..
1. Load your APP into the CW15 Environment.
2. View the Application Tree
3. Click on the 'Global' Button
4. Click on the 'Extensions' Button
5. Click on the 'Insert' button
6. Highlight the DAS_VarFilePath Template under the 'Class DAS_Tools' area.
7. Click on the 'Select' Button

DAS_DateFix (Procedure Extension Template)

Fixes the 2 digit year for the year 2000. With the year 2000 coming around the corner, many data entry personell are worried about changing to posting a 4 digit year. This template allows the developer to enter a Year Limit (or cut off year) like 1980. When any date is entered into an entry field the code is generated to check if it should be adjusted for the year 2000 or left alone. Any date entered prior to the cut off year will adjusted for the year 2000.

Example:

Suppose a Year Limit was set to 1980.
01/01/15 will be converted to 01/01/2015
01/01/81 will be converted to 01/01/1981

To add DAS_DateFix to your APP, simply follow the directions below...
1. Load your APP into the CW15 Environment.
2. View the Application Tree

3. Highlight the Procedure which you would like to add the DAS_DateFix code to.
4. Click on the 'Properties' Button
5. Click on the 'Extensions' Button
6. Click on the 'Insert' Button
7. Highlight the DAS_Date Template under the 'Class DAS_Tools' area.
8. Click on the 'Select' Button

DAS_ApplStats (Application Extension Template)

Simply adds a quick document section which allows the developer to enter his copyright, company name, time stamp compile, version, release, build number and comments into the code.

To add DAS_AppStats to your APP simply follow the directions below..
1. Load your APP into the CW15 Environment.
2. View the Application Tree
3. Click on the 'Global' Button
4. Click on the 'Extensions' Button
5. Click on the 'Insert' button
6. Highlight the DAS_AppStats Template under the 'Class DAS_Tools' area.
7. Click on the 'Select' Button

DAS_ProcComments (Procedure Extension Template)

This is an easier way to add comment to your individual procedures.

To add DAS_ProcComments to your APP, simply follow the directions below...
1. Load your APP into the CW15 Environment.
2. View the Application Tree
3. Highlight the Procedure which you would like to add the DAS_ProcComments code to.
4. Click on the 'Properties' Button
5. Click on the 'Extensions' Button
6. Click on the 'Insert' Button
7. Highlight the DAS_ProcComments Template under the 'Class DAS_Tools' area.
8. Click on the 'Select' Button

DAS_SplashTime (Procedure Extension Template)

Utilizes the timer feature before executing the specified action. When the specified seconds have passed the program will execute the specified action you defined. This action can be as simple as closing the window or you can call a procedure or insert source code.

To add DAS_SplashTime to your APP, simply follow the directions below...
1. Load your APP into the CW15 Environment.
2. View the Application Tree
3. Highlight the Procedure which you would like to add the DAS_SplashTime code to.
4. Click on the 'Properties' Button
5. Click on the 'Extensions' Button
6. Click on the 'Insert' Button

 7. Highlight the DAS_SplashTime Template under the 'Class DAS_Tools' area.
 8. Click on the 'Select' Button

<u>----- Control Templates -----</u>

DAS_SplashMouse (Window Control Template)

Puts a Splash Region inside a window. When the specified mouse movement is detected the program will execute the specified action you defined. This action can be as simple as closing the window or you can call a procedure or insert source code.

To add DAS_SplashMouse to your APP, simply follow the directions below...
 1. Load your APP into the CW15 Environment.
 2. View the Application Tree
 3. Highlight the Procedure which you would like to add the DAS_SplashMouse code to.
 4. Click on the 'Properties' Button
 5. Click on the 'Window' Button
 6. Click on the 'Control Template' Icon (lower right icon in the Control Window or ToolBox Window)
 7. Highlight the DAS_SplashMouse Template under the 'Class DAS_Tools' area.
 8. Click on the 'Select' Button
 9. Place the mouse cursor where you would like the splash box to be and click the mouse.
10. A Splash Region will appear on the screen, place and resize to your satisfaction.
11. Press the ENTER key to bring up the Region Properties screen.
12. Go to the Actions Tab and Specify mouse movement activity and the specify the action to take.
    (This action can be closeing the window or calling a procedure)

<u>----- Code Templates -----</u>

DAS_MessageBox (Code Template)

This just makes building the Message Boxes a little easier. Just insert this code template in the desired embed point and fill in the blanks. You may select to use the Clarion Standard Message Box or the DAS_MsgFunc window.

To add DAS_MessageBox to your APP simply follow the directions below..
 1. Load your APP into the CW15 Environment.
 2. View the Application Tree
 3. Click on the 'Properties' Button
 4. Click on the 'Embeds' Button
 5. Highlight the Embed Point you want.
 6. Click on the 'Insert' button
 7. Highlight the DAS_MessageBox Template under the 'Class DAS_Tools' area.
 8. Click on the 'Select' Button
 9. Just fill in the blanks and press the 'OK' button.