# DateTime Library

**mrfearless**

**Jun 29, 2022**

# CONTENTS:


**1 Introduction** **1**
- 1.1 Overview . . . 1
- 1.2 Installation & Setup . . . 2
- 1.3 Building . . . 3
- 1.4 Contributing . . . 5
- 1.5 Frequently Asked Questions . . . 7

**2 DateTime Functions x86** **9**
- 2.1 DTDateFormat . . . 9
- 2.2 DTDateStringsCompare . . . 11
- 2.3 DTDateStringsDifference . . . 12
- 2.4 DTDateTimeStringToDwordDateTime . . . 13
- 2.5 DTDateTimeStringToJulianMillisec . . . 15
- 2.6 DTDateTimeStringToUnixTime . . . 16
- 2.7 DTDateTimeStringsDifference . . . 17
- 2.8 DTDay . . . 19
- 2.9 DTDayOfWeek . . . 19
- 2.10 DTDwordDateTimeToDateTimeString . . . 20
- 2.11 DTDwordDateTimeToJulianMillisec . . . 22
- 2.12 DTDwordDateTimeToUnixTime . . . 23
- 2.13 DTDwordDateToJulianDate . . . 24
- 2.14 DTDwordTimeToMillisec . . . 25
- 2.15 DTGetDateTime . . . 26
- 2.16 DTIsLeapYear . . . 26
- 2.17 DTJulianDateToDwordDate . . . 27
- 2.18 DTJulianMillisecToDateTimeString . . . 28
- 2.19 DTJulianMillisecToDwordDateTime . . . 29
- 2.20 DTMillisecToDwordTime . . . 31
- 2.21 DTMonth . . . 31
- 2.22 DTUnixTimeToDateTimeString . . . 32
- 2.23 DTUnixTimeToDwordDateTime . . . 33
- 2.24 DTYear . . . 35

**3 DateTime Functions x64** **37**
- 3.1 DTDateFormat . . . 37
- 3.2 DTDateStringsCompare . . . 39
- 3.3 DTDateStringsDifference . . . 40
- 3.4 DTDateTimeStringToJulianMillisec . . . 41
- 3.5 DTDateTimeStringToQwordDateTime . . . 42
- 3.6 DTDateTimeStringToUnixTime . . . 45

3.7 DTDateTimeStringsDifference . . . 46
3.8 DTDay . . . 47
3.9 DTDayOfWeek . . . 47
3.10 DTGetDateTime . . . 48
3.11 DTIsLeapYear . . . 49
3.12 DTJulianDateToQwordDate . . . 50
3.13 DTJulianMillisecToDateTimeString . . . 51
3.14 DTJulianMillisecToQwordDateTime . . . 52
3.15 DTMillisecToQwordTime . . . 54
3.16 DTMonth . . . 55
3.17 DTQwordDateTimeToDateTimeString . . . 56
3.18 DTQwordDateTimeToJulianMillisec . . . 58
3.19 DTQwordDateTimeToUnixTime . . . 59
3.20 DTQwordDateToJulianDate . . . 60
3.21 DTQwordTimeToMillisec . . . 62
3.22 DTUnixTimeToDateTimeString . . . 63
3.23 DTUnixTimeToQwordDateTime . . . 64
3.24 DTYear . . . 65

**4 DateTime Formats** **69**

**5 Indices and tables** **71**

CHAPTER

ONE

# INTRODUCTION

## 1.1 Overview

DateTime is a library which provides simple date and time conversion, manipulation and comparison functions for use in any of your projects.

### 1.1.1 Download

The DateTime Library, both x86 and x64 versions, are available to download from the github repository at github.com/mrfearless/DateTime-Library

### 1.1.2 Features

- Conversion of date time strings to dword or Julian Date
- Comparison of date time strings to determine if one is greater than the other
- Difference calculation of date time strings and return as days difference - or +
- Leap year calculation
- Day of the week calculation

### 1.1.3 Installation

The DateTime Library are available as both x86 and x64 versions.

See the *Installation & Setup* section for more details.

### 1.1.4 Contribute

If you wish to contribute, please follow the *Contributing* guide for details on how to add or edit, the DateTime Library source or documentation.

### 1.1.5 Frequently Asked Questions

Please visit the *Frequently Asked Questions* section for details.

## 1.2 Installation & Setup

### 1.2.1 MASM for 32bit assembler

- Download and install the MASM32 package.
- Download the 32bit version of the DateTime Library: DateTime-x86.zip
- Copy the `DateTime.inc` file to `X:\MASM32\Include` folder overwriting any previous versions.
- Copy the `DateTime.lib` file to `X:\MASM32\Lib` folder overwriting any previous versions.

---

**Note:** `X` is the drive letter where the MASM32 package has been installed to.

---

#### Adding DateTime Library to your MASM project

You are now ready to begin using the DateTime library in your Masm projects. Simply add the following lines to your project:

```
include DateTime.inc
includelib DateTime.lib
```

### 1.2.2 UASM for 64bit assembler

- Download and install the UASM assembler. Ideally you will have a setup that mimics the MASM32 setup, where you create manually folders for `bin`, `include` and `lib`
- Download the 64bit version of the DateTime Library: DateTime-x64.zip
- Copy the `DateTime.inc` file to `X:\UASM\Include` folder overwriting any previous versions.
- Copy the `DateTime.lib` file to `X:\UASM\Lib\x64` folder overwriting any previous versions.

---

**Note:** `X` is the drive letter where the UASM package has been installed to.

---

#### Adding DateTime Library to your UASM project

You are now ready to begin using the DateTime library in your Uasm projects. Simply add the following lines to your project:

```
include DateTime.inc
includelib DateTime.lib
```

---

**Note:** See the following for more details on setting up UASM to work with RadASM and other details that may be useful in creating a development environment that mimics the MASM32 SDK: UASM-with-RadASM, UASM-SDK

---

**Tip:** UASM can be used as a x86 32 bit assembler as well.

## 1.3 Building

The DateTime Libraries (both x86 and x64 versions) come with RadASMs project to help you build the sources. However if you wish to build the libraries manually, here are the command line options you should use.

### 1.3.1 Building the DateTime Library x86

The DateTime Library x86 source consists of the following files:

- `DateTime.inc`
- `DTDateFormat.asm`
- `DTDatesStringCompare.asm`
- `DTDateStringsDifference.asm`
- `DTDateTimeStringsDifference.asm`
- `DTDateTimeStringToDwordDateTime.asm`
- `DTDateTimeStringToJulianMillisec.asm`
- `DTDateTimeStringToUnixTime.asm`
- `DTDay.asm`
- `DTDayOfWeek.asm`
- `DTDwordDateTimeToDateTimeString.asm`
- `DTDwordDateTimeToJulianMillisec.asm`
- `DTDwordDateTimeToUnixTime.asm`
- `DTDwordDateToJulianDate.asm`
- `DTDwordTimeToMillisec.asm`
- `DTGetDateTime.asm`
- `DTIsLeapYear.asm`
- `DTJulianDateToDwordDate.asm`
- `DTJulianMillisecToDateTimeString.asm`
- `DTJulianMillisecToDwordDateTime.asm`
- `DTMillisecToDwordTime.asm`
- `DTMonth.asm`
- `DTUnixTimeToDateTimeString.asm`
- `DTUnixTimeToDwordDateTime.asm`
- `DTYear.asm`

**Building with Microsoft MASM** (`ML.EXE`):

```
ML.EXE /c /coff /Cp /nologo /I"X:\MASM32\Include" *.asm
```

**Note:** `X` is the drive letter where the MASM32 package has been installed to.

**Linking with Microsoft Library Manager** (`LIB.EXE`):

```
LIB *.obj /out:DateTime.lib
```

### 1.3.2 Building the DateTime Library x86 Debug Build

To build the DateTime Library x86 with debug information, supply the additional flag options `/Zi /Zd` on the command line for MASM (`ML.EXE`) like so:

```
ML.EXE /c /coff /Cp /Zi /Zd /nologo /I"X:\MASM32\Include" *.asm
```

**Note:** `X` is the drive letter where the MASM32 package has been installed to.

### 1.3.3 Building the DateTime Library x64

The DateTime Library x64 source consists of the following files:

- `DateTime.inc`
- `DTDateFormat.asm`
- `DTDatesStringCompare.asm`
- `DTDateStringsDifference.asm`
- `DTDateTimeStringsDifference.asm`
- `DTDateTimeStringToQwordDateTime.asm`
- `DTDateTimeStringToJulianMillisec.asm`
- `DTDateTimeStringToUnixTime.asm`
- `DTDay.asm`
- `DTDayOfWeek.asm`
- `DTGetDateTime.asm`
- `DTIsLeapYear.asm`
- `DTJulianDateToQwordDate.asm`
- `DTJulianMillisecToDateTimeString.asm`
- `DTJulianMillisecToQwordDateTime.asm`
- `DTMillisecToQwordTime.asm`
- `DTMonth.asm`
- `DTQwordDateTimeToDateTimeString.asm`
- `DTQwordDateTimeToJulianMillisec.asm`

- DTQwordDateTimeToUnixTime.asm
- DTQwordDateToJulianDate.asm
- DTQwordTimeToMillisec.asm
- DTUnixTimeToDateTimeString.asm
- DTUnixTimeToQwordDateTime.asm
- DTYear.asm

**Building with UASM** (UASM64.EXE):

```
UASM64.EXE /c -win64 -Zp8 /win64 /D_WIN64 /Cp /nologo /W2 /I"X:\UASM\Include" *.asm
```

**Note:** X is the drive letter where the UASM assembler has been installed to.

**Linking with Microsoft Library Manager** (LIB.EXE):

```
LIB *.obj /out:DateTime.lib
```

### 1.3.4 Building the DateTime Library x64 Debug Build

To build the DateTime Library x64 with debug information, supply the additional flag options /Zi /Zd on the command line for UASM (UASM64.EXE) like so:

```
UASM64.EXE /c -win64 -Zp8 /Zi /Zd /win64 /D_WIN64 /Cp /nologo /W2 /I"X:\UASM\Include" *.
↪asm
```

**Note:** X is the drive letter where the MASM32 package has been installed to... note:: X is the drive letter where the UASM assembler has been installed to.

## 1.4 Contributing

To contribute to the DateTime Library source code or documentation you will need to have a Github account. Sign up at www.Github.com if you don't have a Github account already.

### 1.4.1 Requirements

DateTime Library uses RadASM IDE for the projects hosted in the **DateTime-x86** folder and the **DateTime-x64** folder.

For x86 assembler projects:

- Install RadASM
- Install MASM32 SDK

For x64 assembler projects:

- Install UASM
- Install UASM-for-RadASM

- Install UASM-SDK

DateTime Library uses readthedocs for hosting the documentation. This requires the installation of python, and some python extensions.

- Install Python
- Open a command prompt/terminal and type :
  - `pip install sphinx`
  - `pip install sphinx_rtd_theme`
  - `pip install recommonmark`

### 1.4.2 DateTime Library Github Repository

To work with the DateTime Library source and/or documentation files you will need to first clone or fork the DateTime Library repository using a git GUI client or using git commands in a command prompt/terminal. The DateTime Library repository is located at: https://github.com/mrfearless/DateTime-Library

The workflow for users is:

For users who are already authorized contributors:

- Clone the DateTime Library repository locally
- Make changes
- Commit changes back to the DateTime Library repository

For users who are not authorized contributors, but who wish to contribute:

- Fork the DateTime Library repository to your own github account.
- Clone that repository locally
- Make changes
- Commit changes back to your version hosted on your account
- Submit a pull request - which will generate a pending commit on the original DateTime Library repository. This pending commit can be reviewed by the Author and/or other authorized persons and accepted or rejected.
- At some later stage, if changes have occurred on the DateTime Library repository, and if you are to keep up to date with these changes, then instead of deleting your repo version and reforking the DateTime Library, you can rebase your version to sync with the latest changes. Some GUI git clients may offer this as an upstream, which will allow you to sync with the main repository.

### 1.4.3 Editing the DateTime Library source files

Once the requirements above have been met and the repository has been cloned/forked, you are ready to edit and make changes to the DateTime Library source files:

- The DateTime Library source and RadASM project for MASM x86 assembler are stored in **DateTime-x86** folder.
- The DateTime Library source and RadASM project for UASM x64 assembler are stored in **DateTime-x64** folder.

Make your edits to the source files, and then once finished you are ready to submit changes

### 1.4.4 Editing the DateTime Library documentation files

The DateTime Library documentation is stored in the **docs** folder. All the files in that folder and sub-folders are **.rst** files (reStructuredText) and are similar to markdown files. See the reStructuredText Primer for details of usage.

Once any changes you have made are saved, you can preview the changes locally by generating the html files:

- Open a command prompt/terminal and change to the DateTime Library **docs** directory (for example: `cd "c:\Github\DateTime-Library\docs"`)
- Generate the html files by typing `make html`

The sphinx builder will create a **build** folder inside the DateTime Library **docs** folder automatically (if it doesn't exist already) and will create an **html** folder under that. Inside this **html** folder is the html files including the **index.html** which you can open to locally preview your changes.

---

**Note:** The **build** folder won't be included in any pull requests or commits, as its automatically ignored. You can safely ignore this folder, or delete it if you wish, as it will be built automatically each time you run `make html`

---

### 1.4.5 Submitting Changes

Once your happy with your changes, you can then commit to your locally cloned repository and submit a pull request on the DateTime Library on the Github website on your Github account. For authorized contributors you can just commit directly to the DateTime Library repository.

## 1.5 Frequently Asked Questions


- *What does DateTime Library do?*
- *What do I need to do to install DateTime Library?*
- *How do I add to the DateTime Library documentation?*
- *How do I report a bug?*
- *I need more information on date and time algorithms*


### 1.5.1 What does DateTime Library do?

DateTime is a library which provides simple date and time conversion, manipulation and comparison functions for use in any of your assembler projects.

### 1.5.2 What do I need to do to install DateTime Library?

Please follow the *Installation & Setup* guide for details on how to install the DateTime Library for your particular assembler versions that you are using.

### 1.5.3 How do I add to the DateTime Library documentation?

Please follow the *Contributing* guide for details on how to add to, or edit, the DateTime Library documentation.

### 1.5.4 How do I report a bug?

If you should encounter any bugs, please open a new issue on the DateTime Library Github repository.

### 1.5.5 I need more information on date and time algorithms

Here are some useful links for gregorian, julian, millisec, unixtime topics & algorithms:

| |
|---|
| Perpetual Calendar Weekday computation in your head by lutz |
| Julian Day Numbers by William H. Jefferys |
| Julian Day Numbers by Peter Meyer |
| Julian Day Numbers Using Calgo #199 by Paul Myers |
| Julian Day Numbers for dates on the Gregorian and Julian Calendars by Kelley L. Ross |
| How to calculate a Julian Date |
| Time format conversions SYSTEMTIME to FILETIME to SQL time string to Unix time_t by John Stewien |
| Convert milliseconds to time |
| Convert to Unix Time |
| Calendar Converter by John Walker |
| Gregorian calendar |
| Unix Time Stamp - Epoch Converter |

CHAPTER

TWO

# DATETIME FUNCTIONS X86

## 2.1 DTDateFormat

Converts a SYSTEMTIME structure, pointed to by the `lpSystermtimeStruct` parameter, to a date & time string, into the buffer pointed to by the `lpszDateTimeString` parameter and returns it formatted as specified by the `DateFormat` parameter.

```
DTDateFormat PROTO lpSystermtimeStruct:DWORD, lpszDateTimeString:DWORD, DateFormat:DWORD
```

**Parameters**

- `lpSystermtimeStruct` - Pointer to a SYSTEMTIME type that contains date & time information to convert.
- `lpszDateTimeString` - Pointer to a buffer to store the date & time string. The format of the date & time string is determined by the `DateFormat` parameter.
- `DateFormat` - Value indicating the date & time format to return in the buffer pointed to by `lpszDateTimeString` parameter. The parameter can contain one of the following constants as listed in the *DateTime Formats* page and as defined in the `DateTime.inc` include file:

```
; Reverse Date Formats
CCYYMMDDHHMMSSMS                    EQU 1  ; Example 1974/03/27 14:53:01:00
CCYYMMDDHHMMSS                      EQU 2  ; Example 1974/03/27 14:53:01
CCYYMMDDHHMM                        EQU 3  ; Example 1974/03/27 14:53
CCYYMMDDHH                          EQU 4  ; Example 1974/03/27 14:53
CCYYMMDD                            EQU 5  ; Example 1974/03/27
CCYYMM                              EQU 6  ; Example 1974/03
YEAR                                EQU 7  ; Example 1974

; British Date Formats
DDMMCCYYHHMMSSMS                    EQU 8  ; Example 27/03/1974 14:53:01:00
DDMMCCYYHHMMSS                      EQU 9  ; Example 27/03/1974 14:53:01
DDMMCCYYHHMM                        EQU 10 ; Example 27/03/1974 14:53
DDMMCCYY                            EQU 11 ; Example 27/03/1974
DDMM                                EQU 12 ; Example 27/03
DAY                                 EQU 13 ; Example 27

; American Date Formats
MMDDCCYYHHMMSSMS                    EQU 14 ; Example 03/27/1974 14:53:01:00
MMDDCCYYHHMMSS                      EQU 15 ; Example 03/27/1974 14:53:01
MMDDCCYYHHMM                        EQU 16 ; Example 03/27/1974 14:53
MMDDCCYY                            EQU 17 ; Example 03/27/1974
```

(continues on next page)

(continued from previous page)

```
MMDD                                EQU 18 ; Example 03/27
MONTH                               EQU 19 ; Example 03

; Reverse Without Century Date Formats
YYMMDDHHMMSSMS                      EQU 20 ; Example 74/03/27 14:53:01:00
YYMMDDHHMMSS                        EQU 21 ; Example 74/03/27 14:53:01
YYMMDDHHMM                          EQU 22 ; Example 74/03/27 14:53
YYMMDD                              EQU 23 ; Example 74/03/27
YYMM                                EQU 24 ; Example 74/03
YY                                  EQU 25 ; Example 74

; Other Date Formats
MMDDYY                              EQU 26 ; Example 03/27/74
DDMMYY                              EQU 27 ; Example 27/03/74
DAYOFWEEK                           EQU 28 ; Example Monday

; Time Formats
HHMMSSMS                            EQU 29 ; Example 14:53:01
HHMMSS                              EQU 30 ; Example 14:53:01
HHMM                                EQU 31 ; Example 14:53
HH                                  EQU 32 ; Example 14

; Useful Named Constants
TODAY                               EQU DDMMCCYYHHMMSS
NOW                                 EQU DDMMCCYYHHMMSS
TIME                                EQU HHMM

; Named Date Constants
AMERICAN                            EQU MMDDYY
BRITISH                             EQU DDMMYY
FRENCH                              EQU DDMMYY
JAPAN                               EQU YYMMDD
TAIWAN                              EQU YYMMDD
MDY                                 EQU MMDDYY
DMY                                 EQU DDMMYY
YMD                                 EQU YYMMDD
```

**Note:** Constants: CC=Century, YY=Year, MM=Month, DD=Day, HH=Hours, MM=Minutes, SS=Seconds, MS=Milliseconds, DOW=Day Of Week

**Returns**

Returns in `EAX` `TRUE` if succesful, or `FALSE` otherwise.

**Notes**

The user is responsible for ensuring the buffer pointed to by the `lpszDateTimeString` parameter is large enought to hold the date & time string value. Recommeneded minimum size for the buffer is `24` bytes long as defined by the `DATETIME_STRING` constant.

**Example**

```
.data
LocalDateTime SYSTEMTIME <>
szDataAndTimeString DB DATETIME_STRING DUP (0)

.code
Invoke GetLocalTime, Addr LocalDateTime
Invoke DTDateFormat, Addr LocalDateTime, Addr szDataAndTimeString, DDMMYY
```

**See Also**

*DTGetDateTime*, *DateTime Formats*

## 2.2 DTDateStringsCompare

Compare two date & time strings to determine if they are: equal in value, one is less than another, or one is greater than another.

```
DTDateStringsCompare PROTO lpszDateTimeString1:DWORD, lpszDateTimeString2:DWORD,␣
↪DateFormat:DWORD
```

**Parameters**

- `lpszDateTimeString1` - Pointer to a buffer containing the first date & time string to compare. The format of the date & time string is determined by the `DateFormat` parameter.
- `lpszDateTimeString2` - Pointer to a buffer containing the second date & time string to compare. The format of the date & time string is determined by the `DateFormat` parameter.
- `DateFormat` - Value indicating the date & time format used in the buffer pointed to by both the `lpszDateTimeString1` parameter and the `lpszDateTimeString2` parameter. The parameter can contain one of the following constants as listed in the *DateTime Formats* page and as defined in the `DateTime.inc` include file.

**Returns**

Return values in `EAX` indicate the following:

- `-1` First date is less than second date.
- `0` First date is equal to second date.
- `+1` First date is greater than second date.

**Notes**

The date & time string pointed to by the `lpszDateTimeString1` parameter is compared against the date & time string pointed to by the `lpszDateTimeString2` parameter.

Both dates are converted to `DWORD` values internally before the comparison to see which is greater or less than or equal.

If a date & time string contains time information this will be ignored.

**Example**

```
Invoke DTDateStringsCompare, Addr szDateTime1, Addr szDateTime2, CCYYMMDD
```

**See Also**

*DTDateStringsDifference*, *DTDateTimeStringsDifference*, *DateTime Formats*

## 2.3 DTDateStringsDifference

Returns the difference in days of the first date & time string to the second date & time string.

```
DTDateStringsDifference PROTO lpszDateTimeString1:DWORD, lpszDateTimeString2:DWORD,␣
↪DateFormat:DWORD
```

**Parameters**

- `lpszDateTimeString1` - Pointer to a buffer containing the first date & time string. The format of the date & time string is determined by the `DateFormat` parameter.
- `lpszDateTimeString2` - Pointer to a buffer containing the second date & time string. The format of the date & time string is determined by the `DateFormat` parameter.
- `DateFormat` - Value indicating the date & time format used in the buffer pointed to by both the `lpszDateTimeString1` parameter and the `lpszDateTimeString2` parameter. The parameter can contain one of the following constants as listed in the *DateTime Formats* page and as defined in the `DateTime.inc` include file.

**Returns**

On return `EAX` contains the difference between the two dates in days as an integer, either positive or negative.

**Notes**

The date & time string pointed to by the `lpszDateTimeString1` parameter is compared against the date & time string pointed to by the `lpszDateTimeString2` parameter to determine the difference of the dates.

Both dates are converted to Julian date format internally before comparing the days difference between dates.

If a date & time string contains time information this will be ignored.

**Example**

```
.data
CurrentDateTime db DATETIME_STRING dup (0)
SomeOtherDateTime db "2008/03/21",0

.data?
DayDifference dd ?

.code
Invoke DTGetDateTime, Addr CurrentDateTime, CCYYMMDD
Invoke DTDateStringsDifference, Addr CurrentDateTime, Addr SomeOtherDateTime, CCYYMMDD
mov DayDifference, eax ; save day difference to data variable

.IF eax == 0
     ; No difference in dates, do something...
.ELSE
     ; Day difference is - or +, do something else...
.ENDIF
```

**See Also**

*DTDateStringsCompare*, *DTDateTimeStringsDifference*, *DateTime Formats*

## 2.4 DTDateTimeStringToDwordDateTime

Converts a formatted date & time string to DWORD values. On return EAX contains the **date** information and EDX the **time** information.

```
DTDateTimeStringToDwordDateTime PROTO lpszDateTimeString:DWORD, DateFormat:DWORD
```

**Parameters**

- lpszDateTimeString - Pointer to a buffer containing the date & time string to convert to DWORD values. The format of the date & time string is determined by the DateFormat parameter.
- DateFormat - Value indicating the date & time format used in the buffer pointed to by lpszDateTimeString parameter. The parameter can contain one of the following constants as listed in the *DateTime Formats* page and as defined in the DateTime.inc include file.

**Returns**

On return EAX will contain the **date** information in the following format:

EAX Register Bits:

| WORD | BYTE | BYTE |
|---|---|---|
| Bits 31-16 | Bits 15-8 | Bits 7-0 |
| Century Year | Month | Day |
| CCCCYY | MM | DD |

On return EDX will contain the **time** information in the following format:

EDX Register Bits:

| BYTE | BYTE | BYTE | BYTE |
|---|---|---|---|
| Bits 31-23 | Bits 23-16 | Bits 15-8 | Bits 7-0 |
| Hour | Minute | Second | Millisec |
| HH | MM | SS | MS |

**Notes**

The registers always return the information in the format shown in the tables above, and do not change regardless of the DateFormat parameter.

The DateFormat parameter is used to indicate the format of the date & time string being passed to DTDateTimeStringToDwordDateTime.

Some information may be unavailable to convert if the passed DateTime string does not contain enough information. For example if only year information is in the DateTime string, then the DWORD **date** and DWORD **time** will be based only on that.

To prevent this, always pass a full date time string of CCYYMMDDHHMMSSMS format or similar (DDMMCCYYHHMMSSMS or MMDDCCYYHHMMSSMS)

To retrieve the various values of the date & time DWORD values you can use the following as an example:

**Example**

```
.data
DateTimeStringValue db "2008/03/21 16:21:01:00",0

.data?
DateValue dd ?
TimeValue dd ?
CentYear dd ?
Year dd ?
Month dd ?
Day dd ?

.code
Invoke DTDateTimeStringToDwordDateTime, Addr DateTimeStringValue, CCYYMMDDHHMMSSMS
mov DateValue, eax ; Save date info in eax to a data variable
mov TimeValue, edx ; Save time info in edx to a data variable

; save day, month, year and century info to data variables
mov edx, DateValue ; use edx as our work place

xor eax, eax ; clear register
mov al, dl ; dl contains day info
mov Day, eax ; save day info DD in dl to a data variable

xor eax, eax ; clear register
mov al, dh ; dh contains month info
mov Month, eax ; save month info MM in dh to a data variable

shr edx, 16 ; century and year are now in dx part of the register
mov CentYear, edx ; save century and year info CCYY to a data variable

xor eax, eax ; clear register
mov al, dl ; dl contains YY part of year
mov Year, eax ; save year info in dl to a data variable
```

**See Also**

*DTDwordDateTimeToDateTimeString*, *DateTime Formats*, *DTGetDateTime*

## 2.5 DTDateTimeStringToJulianMillisec

Converts a formatted date & time string to a Julian date integer in EAX and milliseconds in EDX

```
DTDateTimeStringToJulianMillisec PROTO lpszDateTimeString:DWORD, DateFormat:DWORD
```

**Parameters**

- lpszDateTimeString - Pointer to a buffer containing the date & time string to convert to Julian date integer and millisec values. The format of the date & time string is determined by the DateFormat parameter.
- DateFormat - Value indicating the date & time format used in the buffer pointed to by lpszDateTimeString parameter. The parameter can contain one of the following constants as listed in the *DateTime Formats* page and as defined in the DateTime.inc include file.

**Returns**

On return EAX contains Julian date integer value representing the date specified and EDX will contain the total time in milliseconds.

**Notes**

Some information may be unavailable to convert if the passed DateTime string does not contain enough information. For example if only year information is in the DateTime string, then the Julian date and millisec time will be based only on that.

To prevent this, always pass a full date time string of CCYYMMDDHHMMSSMS format or similar (DDMMCCYYHHMMSSMS or MMDDCCYYHHMMSSMS)

The Julian Day Count is a uniform count of days from a remote epoch in the past (-4712 January 1, 12 hours Greenwich Mean Time (Julian proleptic Calendar) = 4713 BCE January 1, 12 hours GMT (Julian proleptic Calendar) At this instant, the Julian Day Number is 0.

The Julian Day Count has nothing to do with the Julian Calendar introduced by Julius Caesar. It is named for Julius Scaliger, the father of Josephus Justus Scaliger, who invented the concept. It can also be thought of as a logical follow-on to the old Egyptian civil calendar, which also used years of constant lengths.

Scaliger chose the particular date in the remote past because it was before recorded history and because in that year, three important cycles coincided with their first year of the cycle: The 19-year Metonic Cycle, the 15-year Indiction Cycle (a Roman Taxation Cycle) and the 28-year Solar Cycle (the length of time for the old Julian Calendar to repeat exactly).

**Example**

```
.data
DateTimeStringValue db "2008/03/21 16:21:01:00",0

.data?
JulianDateValue dd ?
TotalMilliseconds dd ?

.code
Invoke DTDateTimeStringToJulianMillisec, Addr DateTimeStringValue
mov JulianDateValue, eax ; save julian date value to data variable
mov TotalMilliseconds, edx ; save total milliseconds to data variable
```

**Example**

```
.data?
dwDate1 dd ?
dwDate2 dd ?

.code
;
↪==========================================================================================
; Example to calculate difference in 2 date values over month boundary
Invoke DTDateTimeStringToJulianMillisec, CTEXT("27/02/2009"), DDMMCCYY
mov dwDate1, eax ; save dword date portion to variable, ignore edx which contains time␣
↪info

Invoke DTDateTimeStringToJulianMillisec, CTEXT("03/03/2009"), DDMMCCYY
mov dwDate2, eax ; save dword date portion to variable, ignore edx which contains time␣
↪info

mov ebx, dwDate1 ; Place date1 in ebx
sub eax, ebx ; subtract date2 from date1, eax = 4 days

;
↪==========================================================================================
; Another example to calculate difference in weeks between 2 date values
Invoke DTDateTimeStringToJulianMillisec, CTEXT("03/01/2009"), DDMMCCYY
mov dwDate1, eax ; save dword date portion to variable, ignore edx which contains time␣
↪info

Invoke DTDateTimeStringToJulianMillisec, CTEXT("10/01/2009"), DDMMCCYY
mov dwDate2, eax ; save dword date portion to variable, ignore edx which contains time␣
↪info

mov ebx, dwDate1 ; Place date1 in ebx
sub eax, ebx ; subtract date2 from date1, eax = 7 days
xor edx, edx ; clear edx for division
mov ecx, 7d ; Divisor is 7 in ecx
div ecx ; divide eax by 7, eax = 1
```

**See Also**

*DTJulianMillisecToDateTimeString*

## 2.6 DTDateTimeStringToUnixTime

Converts a formatted date & time string to a `DWORD` value containing a unix time integer value.

```
DTDateTimeStringToUnixTime PROTO lpszDateTimeString:DWORD, DateFormat:DWORD
```

**Parameters**

- `lpszDateTimeString` - Pointer to a buffer containing the date & time string to convert to a `DWORD` value containing a unix time integer value. The format of the date & time string is determined by the `DateFormat` parameter.
- `DateFormat` - Value indicating the date & time format used in the buffer pointed to by `lpszDateTimeString` parameter. The parameter can contain one of the following constants as listed in the *DateTime Formats* page and

as defined in the DateTime.inc include file.

**Returns**

On return EAX contains the unix time as a DWORD value.

**Notes**

Some information may be unavailable to convert if the passed DateTime string does not contain enough information. For example if only year information is in the DateTime string, then the UnixTime will be based only on that.

To prevent this, always pass a full date time string of CCYYMMDDHHMMSSMS format or similar (DDMMCCYYHHMMSSMS or MMDDCCYYHHMMSSMS)

Unix time is defined as the number of seconds elapsed since 00:00 Universal time on January 1, 1970 in the Gregorian calendar (Julian day 2440587.5)

The UNIXTIMESTAMP format is a **string** representation of the unix time in integer format if used as the DateFormat value.

**Example**

```
.data
DateTimeStringValue db "2008/03/21 16:21:01:00",0

.data?
UnixTime dd ?

.code
Invoke DTDateTimeStringToUnixTime, Addr DateTimeStringValue, CCYYMMDDHHMMSSMS
mov UnixTime, eax ; save unix time value to data variable which should contain 1206116461
```

**Example**

```
.data
UnixTimeAsAStringValue db "1656241202",0 ; Sun Jun 26 2022 11:00

.data?
UnixTime dd ?

.code
Invoke DTDateTimeStringToUnixTime, Addr UnixTimeAsAStringValue, UNIXTIMESTAMP
mov UnixTime, eax ; save unix time value to data variable which should contain 1656241202
```

**See Also**

*DTUnixTimeToDateTimeString*, *DTUnixTimeToDwordDateTime*, *DateTime Formats*

## 2.7 DTDateTimeStringsDifference

Returns the difference in days or milliseconds of the first date & time string to the second date & time string.

```
DTDateTimeStringsDifference PROTO lpszDateTimeString1:DWORD, lpszDateTimeString2:DWORD,␣
↪DateFormat:DWORD
```

**Parameters**

- `lpszDateTimeString1` - Pointer to a buffer containing the first date & time string. The format of the date & time string is determined by the `DateFormat` parameter.
- `lpszDateTimeString2` - Pointer to a buffer containing the second date & time string. The format of the date & time string is determined by the `DateFormat` parameter.
- `DateFormat` - Value indicating the date & time format used in the buffer pointed to by both the `lpszDateTimeString1` parameter and the `lpszDateTimeString2` parameter. The parameter can contain one of the following constants as listed in the *DateTime Formats* page and as defined in the `DateTime.inc` include file.

**Returns**

On return `EAX` contains the difference between the two dates in days as an integer, either positive or negative.

On return `EDX` contains the difference in time or `0`, see notes below for details.

**Notes**

If both dates are the same date, then `EAX` will be `0`. If both date & time strings contain time values and the `DateFormat` parameter specified also provides a time format, then `EDX` will contain the difference in time in milliseconds.

**Example**

```
.data
CurrentDateTime db DATETIME_STRING dup (0)
SomeOtherDateTime db "2008/03/21",0

.data?
DayDifference dd ?
MillisecDifference dd ?

.code
Invoke DTGetDateTime, Addr CurrentDateTime, CCYYMMDDHHMMSSMS
Invoke DTDateTimeStringsDifference, Addr CurrentDateTime, Addr SomeOtherDateTime,␣
↪CCYYMMDDHHMMSSMS
mov DayDifference, eax ; save day difference to data variable
mov MillisecDifference, edx ; save millisec diff if days the same

.IF eax == 0
     ; No difference in dates
     ; So MillisecDifference can be used
.ELSE
     ; Day difference is - or +, do something else...
     ; MillisecDifference is 0 value
.ENDIF
```

**See Also**

*DTDateStringsCompare*, *DTDateStringsDifference*, *DateTime Formats*

## 2.8 DTDay

Get the day part of a date & time string.

```
DTDay PROTO lpszDateTimeString:DWORD, DateFormat:DWORD
```

**Parameters**

- `lpszDateTimeString` - Pointer to a buffer containing the date & time string to retrieve the **day** portion from. The format of the date & time string is determined by the `DateFormat` parameter.
- `DateFormat` - Value indicating the date & time format used in the buffer pointed to by `lpszDateTimeString` parameter. The parameter can contain one of the following constants as listed in the *DateTime Formats* page and as defined in the `DateTime.inc` include file.

**Returns**

Returns the day value in `EAX`

**Example**

```
.data
DateTimeStringValue db "2008/03/21 16:21:01:00",0

.data?
DayValue dd ?

.code
Invoke DTDay, Addr DateTimeStringValue, CCYYMMDDHHMMSSMS
mov DayValue, eax ; save day part of date value to data variable
; eax should contain 21
```

**See Also**

*DTMonth*, *DTYear*, *DTIsLeapYear*, *DTDayOfWeek*, *DateTime Formats*

## 2.9 DTDayOfWeek

Returns an integer indicating the day of the week from a date & time string.

```
DTDayOfWeek PROTO lpszDateTimeString:DWORD, DateFormat:DWORD
```

**Parameters**

- `lpszDateTimeString` - Pointer to a buffer containing the date & time string to check the day of the week for. The format of the date & time string is determined by the `DateFormat` parameter.
- `DateFormat` - Value indicating the date & time format used in the buffer pointed to by `lpszDateTimeString` parameter. The parameter can contain one of the following constants as listed in the *DateTime Formats* page and as defined in the `DateTime.inc` include file.

**Returns**

On return `EAX` will contain a value to indicate the following day:

- `0` = Monday
- 1 = Tuesday

- 2 = Wednesday
- 3 = Thursday
- 4 = Friday
- 5 = Saturday
- 6 = Sunday

**Example**

```
; The following example calculates the first day of the week from a date using the␣
↪DTDayOfWeek function
.data
szDateTime db DATETIME_STRING dup (0)
.data?
dwDate1 dd ?
dwDayWeek dd ?
dwStartWeek dd ?

.code
Invoke DTDateTimeStringToJulianMillisec, CTEXT("04/06/2009"), DDMMCCYY ; Convert date to␣
↪julian date
mov dwDate1, eax ; only save date info returned in eax, ignore time info returned in edx

Invoke DTDayOfWeek, CTEXT("04/06/2009"), DDMMCCYY ; Get day of week for date
mov dwDayWeek, eax ; Save the returned integer representing day 0-6

mov ebx, eax
mov eax, dwDate1
sub eax, ebx ; subtract julian date value from day of week integer
mov dwStartWeek, eax ; we have new julian date which is start of the week

Invoke DTJulianToDwordDate, dwStartWeek ; convert new julian date start of week number␣
↪to a date in dword format
Invoke DTDwordDateTimeToDate, eax, NULL, Addr szDateTime, DDMMCCYY ; convert dword date␣
↪format to a string value
; szDateTime now contains '01/06/2009'
```

**See Also**

*DTDay*, *DTMonth*, *DTYear*, *DTIsLeapYear*, *DateTime Formats*

## 2.10 DTDwordDateTimeToDateTimeString

Converts DWORD values containing date & time information to a formatted date & time string as specified by the DateFormat parameter.

```
DTDwordDateTimeToDateTimeString PROTO dwDate:DWORD, dwTime:DWORD,␣
↪lpszDateTimeString:DWORD, DateFormat:DWORD
```

**Parameters**

- dwDate - DWORD value containing **Date** information to convert to the date & time string

The format for the value containing the **date** information is as follows:

**date** DWORD Register Bits:

| WORD | BYTE | BYTE |
|---|---|---|
| Bits 31-16 | Bits 15-8 | Bits 7-0 |
| Century Year | Month | Day |
| CCCCYY | MM | DD |

- dwTime - DWORD value containing **Time** information to convert to the date & time string

  The format for the value containing the **time** information is as follows:

  **time** DWORD Register Bits:

| BYTE | BYTE | BYTE | BYTE |
|---|---|---|---|
| Bits 31-23 | Bits 23-16 | Bits 15-8 | Bits 7-0 |
| Hour | Minute | Second | Millisec |
| HH | MM | SS | MS |

- lpszDateTimeString - Pointer to a buffer to store the date & time string. The format of the date & time string is determined by the DateFormat parameter.
- DateFormat - Value indicating the date & time format to return in the buffer pointed to by lpszDateTimeString parameter. The parameter can contain one of the following constants as listed in the *DateTime Formats* page and as defined in the DateTime.inc include file.

**Returns**

No return value

**Notes**

**Example**

```
.data
DateTimeStringValue db DATETIME_STRING dup (0) ; buffer to store date and time as string
DateValue dd 0
TimeValue dd 0

.code
xor eax, eax
mov eax, 1974 ; save year
shl eax, 16 ; move it into upper word of eax
mov ah, 03 ; save month
mov al, 15 ; save day
mov DateValue, eax
xor eax, eax
```

(continues on next page)

(continued from previous page)

```
mov ah, 12 ; save hours
mov al, 34 ; save minutes
shl eax, 16 ; move it into upper word of eax
mov ah, 01 ; save seconds
mov al, 0 ; save milliseconds
mov TimeValue, eax

Invoke DTDwordDateTimeToDateTimeString, DateValue, TimeValue, Addr DateTimeStringValue,␣
↪CCYYMMDDHHMMSS
; DateTimeStringValue will now contain "1974/03/15 12:36:01"
```

**See Also**

*DTDateTimeStringToDwordDateTime*, *DateTime Formats*, *DTGetDateTime*

## 2.11 DTDwordDateTimeToJulianMillisec

Converts DWORD values containing date & time information to Julian date and total milliseconds.

```
DTDwordDateTimeToJulianMillisec PROTO dwDate:DWORD, dwTime:DWORD
```

**Parameters**

- dwDate - DWORD value containing **date** information to convert to Julian date.

  The format for the value containing the **date** information is as follows:

  **date** DWORD Register Bits:

| WORD | BYTE | BYTE |
|---|---|---|
| Bits 31-16 | Bits 15-8 | Bits 7-0 |
| Century Year | Month | Day |
| CCCCYY | MM | DD |

- dwTime - DWORD value containing **time** information to convert to the total milliseconds.

  The format for the value containing the **time** information is as follows:

  **time** DWORD Register Bits:

| BYTE | BYTE | BYTE | BYTE |
|---|---|---|---|
| Bits 31-23 | Bits 23-16 | Bits 15-8 | Bits 7-0 |
| Hour | Minute | Second | Millisec |
| HH | MM | SS | MS |

**Returns**

On return EAX will contain the Julian date integer, and EDX the total millliseconds.

**Notes**

**Example**

```
Invoke DTDwordDateTimeToJulianMillisec, dwDate, dwTime
```

**See Also**

*DTJulianMillisecToDwordDateTime*, *DTJulianDateToDwordDate*, *DTMillisecToDwordTime*

## 2.12 DTDwordDateTimeToUnixTime

Converts DWORD values containing date & time information to a DWORD value containing a unix time integer value. On return EAX contains the unix time integer value.

```
DTDwordDateTimeToUnixTime PROTO dwDate:DWORD, dwTime:DWORD
```

**Parameters**

- dwDate - DWORD value containing **date** information to convert to a unix time integer value.

  The format for the value containing the **date** information is as follows:

  **date** DWORD Register Bits:

| WORD | BYTE | BYTE |
|---|---|---|
| Bits 31-16 | Bits 15-8 | Bits 7-0 |
| Century Year | Month | Day |
| CCCCYY | MM | DD |

- dwTime - DWORD value containing **time** information to convert to a unix time integer value.

  The format for the value containing the **time** information is as follows:

  **time** DWORD Register Bits:

| BYTE | BYTE | BYTE | BYTE |
|---|---|---|---|
| Bits 31-23 | Bits 23-16 | Bits 15-8 | Bits 7-0 |
| Hour | Minute | Second | Millisec |
| HH | MM | SS | MS |

**Returns**

On return EAX will contain the unix time integer value.

**Notes**

Unix time is defined as the number of seconds elapsed since 00:00 Universal time on January 1, 1970 in the Gregorian calendar (Julian day 2440587.5)

**Example**

```
Invoke DTDwordDateTimeToUnixTime, dwDate, dwTime
```

**See Also**

*DTUnixTimeToDwordDateTime*, *DTDateTimeStringToUnixTime*, *DTUnixTimeToDateTimeString*

## 2.13 DTDwordDateToJulianDate

Converts a DWORD value containing date information to a Julian date integer. On return EAX contains the date information in Julian Date format.

```
DTDwordDateToJulianDate PROTO dwDate:DWORD
```

**Parameters**

- dwDate - DWORD value containing **Date** information to convert to a Julian date integer.

  The format for the value containing the **date** information is as follows:

  **date** DWORD Register Bits:

| WORD | BYTE | BYTE |
|---|---|---|
| Bits 31-16 | Bits 15-8 | Bits 7-0 |
| Century Year | Month | Day |
| CCCCYY | MM | DD |

**Returns**

On return EAX contains Julian date integer value representing the date specified.

**Notes**

The Julian Day Count is a uniform count of days from a remote epoch in the past (-4712 January 1, 12 hours Greenwich Mean Time (Julian proleptic Calendar) = 4713 BCE January 1, 12 hours GMT (Julian proleptic Calendar) At this instant, the Julian Day Number is 0.

The Julian Day Count has nothing to do with the Julian Calendar introduced by Julius Caesar. It is named for Julius Scaliger, the father of Josephus Justus Scaliger, who invented the concept. It can also be thought of as a logical follow-on to the old Egyptian civil calendar, which also used years of constant lengths.

Scaliger chose the particular date in the remote past because it was before recorded history and because in that year, three important cycles coincided with their first year of the cycle: The 19-year Metonic Cycle, the 15-year Indiction Cycle (a Roman Taxation Cycle) and the 28-year Solar Cycle (the length of time for the old Julian Calendar to repeat exactly).

**Example**

```
.data
Day db 15
Month db 10
CentYear dw 1582
DateValue dd 0

.code
xor eax, eax
mov ax, CentYear
shl eax, 16 ; move year into upper word of eax
mov ah, Month ; move month into ah
mov al, Day ; move day into al
mov DateValue, eax

Invoke DTDwordDateToJulianDate, DateValue
; EAX now contains 2299161 which is the Julian Date for 1582 October 15
```

**See Also**

*DTJulianDateToDwordDate*, *DTMillisecToDwordTime*, *DTDwordDateTimeToJulianMillisec*, *DTJulianMillisecToDwordDateTime*

## 2.14 DTDwordTimeToMillisec

Converts a DWORD value containing time information to total amount of milliseconds.

```
DTDwordTimeToMillisec PROTO dwTime:DWORD
```

**Parameters**

- dwTime - DWORD value containing **time** information to convert to total amount of milliseconds.

  The format for the value containing the **time** information is as follows:

  **time** DWORD Register Bits:

```
+------------+------------+-----------+-----------+
| BYTE       | BYTE       | BYTE      | BYTE      |
+------------+------------+-----------+-----------+
| Bits 31-23 | Bits 23-16 | Bits 15-8 | Bits 7-0  |
+------------+------------+-----------+-----------+
| Hour       | Minute     | Second    | Millisec  |
+------------+------------+-----------+-----------+
| HH         | MM         | SS        | MS        |
+------------+------------+-----------+-----------+
```

**Returns**

On return EAX will contain the total milliseconds.

**Example**

```
Invoke DTDwordTimeToMillisec, dwTime
```

**See Also**

*DTMillisecToDwordTime*, *DTJulianDateToDwordDate*, *DTDwordDateTimeToJulianMillisec*, *DTJulianMillisecToDwordDateTime*

## 2.15 DTGetDateTime

Get the current date & time and return it as a formatted string as specified by the DateFormat parameter.

```
DTGetDateTime PROTO lpszDateTimeString:DWORD, DateFormat:DWORD
```

**Parameters**

- lpszDateTimeString - Pointer to a buffer to store the date & time string. The format of the date & time string is determined by the DateFormat parameter.
- DateFormat - Value indicating the date & time format to return in the buffer pointed to by lpszDateTimeString parameter. The parameter can contain one of the following constants as listed in the *DateTime Formats* page and as defined in the DateTime.inc include file.

**Returns**

Returns in eax TRUE if succesful, or FALSE otherwise.

**Notes**

The user is responsible for ensuring the buffer pointed to by the lpszDateTimeString parameter is large enought to hold the date & time string value. Recommeneded minimum size for the buffer is 24 bytes long as defined by the DATETIME_STRING constant.

**Example**

```
.data
DateTimeStringValue db DATETIME_STRING dup (0) ; buffer to store date and time as string

.code
Invoke DTGetDateTime, Addr DateTimeStringValue, CCYYMMDDHHMMSS
```

**See Also**

*DTDateFormat*, *DateTime Formats*

## 2.16 DTIsLeapYear

Calculate if a specified year is a leap year.

```
DTIsLeapYear PROTO dwYear:DWORD
```

**Parameters**

- dwYear - a DWORD value that has the year value to check for leap year.

**Returns**

Returns TRUE if year is a leap year, otherwise FALSE.

**Notes**

dwYear must include the century. For example 2008 which in a DWORD value is stored as 0x7D8

A year will be a leap year if it is divisible by 4 but not by 100. If a year is divisible by 4 and by 100, it is not a leap year unless it is also divisible by 400.

**Example**

```
.data
CurrentDateTimeValue db 32 dup (0) ; buffer to store date & time as string
Year dd 0

.code
Invoke DTGetDateTime, Addr CurrentDateTimeValue, CCYYMMDDHHMMSS
Invoke DTYear, Addr CurrentDateTimeValue, CCYYMMDDHHMMSS
mov Year, eax ; save year

Invoke DTIsLeapYear, eax ; check leap year
.IF eax == TRUE
     ; Leap year, do something...
.ELSE
     ; No leap year, do something other...
.ENDIF
```

**See Also**

*DTDay*, *DTMonth*, *DTYear*, *DTDayOfWeek*

## 2.17 DTJulianDateToDwordDate

Converts a Julian Date number to a DWORD value containing date information. Uses CCYYMMDD *DateTime Format*

```
DTJulianDateToDwordDate PROTO JulianDate:DWORD
```

**Parameters**

- JulianDate - A DWORD value containing the Julian date integer to convert to a DWORD format in EAX

**Returns**

On return EAX will contain the **date** information in the following format:

EAX Register Bits:

| WORD | BYTE | BYTE |
|---|---|---|
| Bits 31-16 | Bits 15-8 | Bits 7-0 |
| Century Year | Month | Day |
| CCCCYY | MM | DD |

**Notes**

The Julian Day Count is a uniform count of days from a remote epoch in the past (-4712 January 1, 12 hours Greenwich Mean Time (Julian proleptic Calendar) = 4713 BCE January 1, 12 hours GMT (Julian proleptic Calendar) At this instant, the Julian Day Number is 0.

The Julian Day Count has nothing to do with the Julian Calendar introduced by Julius Caesar. It is named for Julius Scaliger, the father of Josephus Justus Scaliger, who invented the concept. It can also be thought of as a logical follow-on to the old Egyptian civil calendar, which also used years of constant lengths.

Scaliger chose the particular date in the remote past because it was before recorded history and because in that year, three important cycles coincided with their first year of the cycle: The 19-year Metonic Cycle, the 15-year Indiction Cycle (a Roman Taxation Cycle) and the 28-year Solar Cycle (the length of time for the old Julian Calendar to repeat exactly).

**Example**

```
.data
Day db 0
Month db 0
CentYear dw 0

.code
Invoke DTJulianDateToDwordDate, 2299161 ; Gregorian date of this Julian Date value is␣
↪1582 October 15

mov edx, eax
mov Month, ah
mov Day, al
shr eax, 16 ; move year info into lower word of eax
mov CentYear, ax

; 1582 October 15 - CentYear now contains 1582, Month contains 10, Day contains 15
```

**See Also**

*DTDwordDateToJulianDate*

## 2.18 DTJulianMillisecToDateTimeString

Converts a Julian date integer and millisec time to a formatted date & time string as specified by the `DateFormat` parameter.

```
DTJulianMillisecToDateTimeString PROTO JulianDate:DWORD, Milliseconds:DWORD,␣
↪lpszDateTimeString:DWORD, DateFormat:DWORD
```

**Parameters**

- `JulianDate` - A `DWORD` value containing the Julian date integer to convert to a gregorian date in the date & time string.
- `Milliseconds` - A `DWORD` value containting the total milliseconds to convert to hours, minutes, seconds and milliseconds in the date & time string.
- `lpszDateTimeString` - Pointer to a buffer to store the date & time string. The format of the date & time string is determined by the `DateFormat` parameter.
- `DateFormat` - Value indicating the date & time format to return in the buffer pointed to by `lpszDateTimeString` parameter. The parameter can contain one of the following constants as listed in the *DateTime Formats* page and as defined in the `DateTime.inc` include file.

**Returns**

There is no return value, the date & time string will contain the date & time as specified by the DateFormat parameter.

**Notes**

No time information is converted with the Julian date value, so if formatting outputs a time, it will be empty as in 00:00:00:00

**Example**

```
.data
DateTimeStringValue db DATETIME_STRING dup (0) ; buffer to store date and time as string

.code
Invoke DTJulianMillisecToDateTimeString, 2299161, 0, Addr DateTimeStringValue,␣
↪CCYYMMDDHHMMSS
; DateTimeStringValue now contains: 1582/10/15 00:00:00
```

**See Also**

*DTDateTimeStringToJulianMillisec*, *DateTime Formats*

## 2.19 DTJulianMillisecToDwordDateTime

Converts Julian date integer value and total milliseconds to DWORD values containing date and time information. On return EAX contains the **date** information and EDX the **time** information.

```
DTJulianMillisecToDwordDateTime PROTO JulianDate:DWORD, Milliseconds:DWORD
```

**Parameters**

- JulianDate - A DWORD value containing the Julian date integer to convert to a DWORD date in EAX.
- Milliseconds - A DWORD value containting the total milliseconds to convert to hours, minutes, seconds and milliseconds in DWORD time in EDX.

**Returns**

On return EAX will contain the **date** information in the following format:

EAX Register Bits:

```
+------------------------+------------+-----------+
| WORD                   | BYTE       | BYTE      |
+------------------------+------------+-----------+
| Bits 31-16             | Bits 15-8  | Bits 7-0  |
+------------------------+------------+-----------+
| Century Year           | Month      | Day       |
+------------------------+------------+-----------+
| CCCCYY                 | MM         | DD        |
+------------------------+------------+-----------+
```

On return EDX will contain the **time** information in the following format:

EDX Register Bits:

```
+------------+------------+-----------+-----------+
| BYTE       | BYTE       | BYTE      | BYTE      |
```

(continues on next page)

(continued from previous page)

| Bits 31-23 | Bits 23-16 | Bits 15-8 | Bits 7-0 |
|---|---|---|---|
| Hour | Minute | Second | Millisec |
| HH | MM | SS | MS |

**Notes**

The registers always return the information in the format shown in the tables above.

**Example**

```
.data?
dwDate dd ?
dwTime dd ?
Year dw ?
Month db ?
Day db ?
Hours db ?
Minutes db ?
Seconds db ?
Millisec db ?

.code
Invoke DTJulianMillisecToDwordDateTime, 2299161, 0
mov dwDate, eax ; eax contains the date information
mov dwTime, edx ; edx contains the time information

mov eax, dwDate
shr eax, 16 ; move year into upper word of eax
mov Year,  ax ; year
mov eax, dwDate
mov Month, ah ; month
mov Day, al ; day

mov eax, dwTime
shr eax, 16 ; move hour and minute into upper word of eax
mov Hours, ah ; hours
mov Minutes, al ; minutes
mov eax, dwTime
mov Seconds, ah ; seconds
mov Millisec, al ; millseconds
```

**See Also**

*DTDwordDateTimeToJulianMillisec*, *DTDwordDateToJulianDate*, *DTDwordTimeToMillisec*

## 2.20 DTMillisecToDwordTime

Converts total milliseconds to a DWORD value containing time information in hours, minutes, seconds and milliseconds.

```
DTMillisecToDwordTime PROTO Milliseconds:DWORD
```

**Parameters**

- Milliseconds - A DWORD value containting the total milliseconds to convert to hours, minutes, seconds and milliseconds in DWORD time in EAX.

**Returns**

On return EAX will contain the **time** information in the following format:

EAX Register Bits:

```
+------------+------------+-----------+-----------+
| BYTE       | BYTE       | BYTE      | BYTE      |
+------------+------------+-----------+-----------+
| Bits 31-23 | Bits 23-16 | Bits 15-8 | Bits 7-0  |
+------------+------------+-----------+-----------+
| Hour       | Minute     | Second    | Millisec  |
+------------+------------+-----------+-----------+
| HH         | MM         | SS        | MS        |
+------------+------------+-----------+-----------+
```

**Notes**

**Example**

```
Invoke DTMillisecToDwordTime, dwMilliseconds
```

**See Also**

*DTDwordTimeToMillisec*, *DTDwordDateToJulianDate*, *DTDwordDateTimeToJulianMillisec*, *DTJulianMillisecToDwordDateTime*

## 2.21 DTMonth

Get the month part of a date & time string.

```
DTMonth PROTO lpszDateTimeString:DWORD, DateFormat:DWORD
```

**Parameters**

- lpszDateTimeString - Pointer to a buffer containing the date & time string to retrieve the **month** portion from. The format of the date & time string is determined by the DateFormat parameter.
- DateFormat - Value indicating the date & time format used in the buffer pointed to by lpszDateTimeString parameter. The parameter can contain one of the following constants as listed in the *DateTime Formats* page and as defined in the DateTime.inc include file.

**Returns**

Returns the month value in EAX

**Example**

```
.data
DateTimeStringValue db "2008/03/21 16:21:01:00",0

.data?
MonthValue dd ?

.code
Invoke DTMonth, Addr DateTimeStringValue, CCYYMMDDHHMMSSMS
mov MonthValue, eax ; save month part of date value to data variable
; eax should contain 3
```

**Example**

```
.data?
dwMonth1 dd ?
dwMonth2 dd ?

.code
Invoke DTMonth, CTEXT("27/02/2009"), DDMMCCYY
mov dwMonth1, eax ; save month to variable

Invoke DTMonth, CTEXT("03/03/2009"), DDMMCCYY
mov dwMonth2, eax ; save month to variable

mov ebx, dwMonth1 ; place month1 variable in ebx
sub eax, ebx ; subtract month2 from month1, eax = 1 in this example
```

**See Also**

*DTDay*, *DTYear*, *DTIsLeapYear*, *DTDayOfWeek*, *DateTime Formats*

## 2.22 DTUnixTimeToDateTimeString

Converts a DWORD containing a unix time integer value to a formatted date & time string as specified by the DateFormat parameter.

```
DTUnixTimeToDateTimeString PROTO UnixTime:DWORD, lpszDateTimeString:DWORD,␣
↪DateFormat:DWORD
```

**Parameters**

- UnixTime - DWORD containing a unix time integer value to convert to a date & time string.
- lpszDateTimeString - Pointer to a buffer to store the date & time string. The format of the date & time string is determined by the DateFormat parameter.
- DateFormat - Value indicating the date & time format to return in the buffer pointed to by lpszDateTimeString parameter. The parameter can contain one of the following constants as listed in the *DateTime Formats* page and as defined in the DateTime.inc include file.

**Returns**

There is no return value, the date & time string will contain the date & time as specified by the DateFormat specified.

**Notes**

Unix time is defined as the number of seconds elapsed since 00:00 Universal time on January 1, 1970 in the Gregorian calendar (Julian day 2440587.5)

The `UNIXTIMESTAMP` format is a **string** representation of the unix time in integer format if used as the `DateFormat` value.

**Example**

```
.data
DateTimeStringValue db DATETIME_STRING dup (0)

.code
Invoke DTUnixTimeToDateTimeString, 1276278420, Addr szDateTimeString, DDMMCCYYHHMMSSMS
; DateTimeString should now contain the string "11/06/2010 17:47:00:00"
```

**Example**

```
.data
DateTimeStringValue db DATETIME_STRING dup (0)

.code
Invoke DTUnixTimeToDateTimeString, 1656241202, Addr szDateTimeString, UNIXTIMESTAMP
; DateTimeString should now contain the string "1656241202"
```

**See Also**

*DTDateTimeStringToUnixTime*, *DTDwordDateTimeToUnixTime*, *DateTime Formats*

## 2.23 DTUnixTimeToDwordDateTime

Converts a `DWORD` containing a unix time integer value to `DWORD` values containing date and time information.

```
DTUnixTimeToDwordDateTime PROTO UnixTime:DWORD
```

**Parameters**

- `UnixTime` - `DWORD` containing a unix time integer value to convert to `DWORD` date & time values.

**Returns**

On return `EAX` will contain the **date** information in the following format:

`EAX` Register Bits:

| WORD | BYTE | BYTE |
|---|---|---|
| Bits 31-16 | Bits 15-8 | Bits 7-0 |
| Century Year | Month | Day |
| CCCCYY | MM | DD |

On return `EDX` will contain the **time** information in the following format:

`EDX` Register Bits:

| BYTE | BYTE | BYTE | BYTE |
|---|---|---|---|
| Bits 31-23 | Bits 23-16 | Bits 15-8 | Bits 7-0 |
| Hour | Minute | Second | Millisec |
| HH | MM | SS | MS |

**Notes**

Unix time is defined as the number of seconds elapsed since 00:00 Universal time on January 1, 1970 in the Gregorian calendar (Julian day 2440587.5)

**Example**

```
.data?
dwDate dd ?
dwTime dd ?
Year dw ?
Month db ?
Day db ?
Hours db ?
Minutes db ?
Seconds db ?
Millisec db ?

.code
Invoke DTUnixTimeToDwordDateTime, 1276278420
mov dwDate, eax ; eax contains the date information
mov dwTime, edx ; edx contains the time information

mov eax, dwDate
shr eax, 16 ; move year into upper word of eax
mov Year,  ax ; year
mov eax, dwDate
mov Month, ah ; month
mov Day, al ; day

mov eax, dwTime
shr eax, 16 ; move hour and minute into upper word of eax
mov Hours, ah ; hours
mov Minutes, al ; minutes
mov eax, dwTime
mov Seconds, ah ; seconds
mov Millisec, al ; millseconds
```

**See Also**

*DTDwordDateTimeToUnixTime*, *DTDateTimeStringToUnixTime*, *DTUnixTimeToDateTimeString*

## 2.24 DTYear

Get the year part of a date & time string.

```
DTYear PROTO lpszDateTimeString:DWORD, DateFormat:DWORD
```

**Parameters**

- `lpszDateTimeString` - Pointer to a buffer containing the date & time string to retrieve the **year** portion from. The format of the date & time string is determined by the `DateFormat` parameter.
- `DateFormat` - Value indicating the date & time format used in the buffer pointed to by `lpszDateTimeString` parameter. The parameter can contain one of the following constants as listed in the *DateTime Formats* page and as defined in the `DateTime.inc` include file.

**Returns**

Returns the year value in `EAX`

**Example**

```
.data
DateTimeStringValue db "2008/03/21 16:21:01:00",0

.data?
YearValue dd ?

.code
Invoke DTYear, Addr DateTimeStringValue
mov YearValue Value, eax ; save year part of date value to data variable
; eax should contain 2008
```

**See Also**

*DTDay*, *DTMonth*, *DTIsLeapYear*, *DTDayOfWeek*, *DateTime Formats*

| Function | Description |
|---|---|
| *DTGetDateTime* | Get the current date & time and return it as a formatted string |
| *DTDateFormat* | Converts a `SYSTEMTIME` structure to a formatted date & time string |
| | |
| *DTDwordDateTimeToDateTimeString* | Converts `DWORD` values containing date & time information to a formatted date & time string |
| *DTDateTimeStringToDwordDateTime* | Converts a formatted date & time string to `DWORD` values containing date and time informatio |
| | |
| *DTDwordDateToJulianDate* | Converts a `DWORD` value containing date information to a Julian date integer |
| *DTJulianDateToDwordDate* | Converts a Julian Date number to a `DWORD` value containing date information |
| | |
| *DTDateTimeStringToJulianMillisec* | Converts a formatted date & time string to a Julian date integer in `EAX` and milliseconds in `E` |
| *DTJulianMillisecToDateTimeString* | Converts a Julian date integer and millisec time to a formatted date & time string |
| | |
| *DTDwordDateTimeToJulianMillisec* | Converts `DWORD` values containing date & time information to Julian date and total millisecon |
| *DTJulianMillisecToDwordDateTime* | Converts Julian date integer value and total milliseconds to `DWORD` values containing date and |
| | |
| *DTDateTimeStringToUnixTime* | Converts a formatted date & time string to a `DWORD` value containing a unix time integer valu |
| *DTUnixTimeToDateTimeString* | Converts a `DWORD` containing a unix time integer value to a formatted date & time string |
| | |

Table 1 – continued from previous page

| | |
|---|---|
| *DTDwordDateTimeToUnixTime* | Converts DWORD values containing date & time information to a DWORD value containing a un |
| *DTUnixTimeToDwordDateTime* | Converts a DWORD containing a unix time integer value to DWORD values containing date and t |
| | |
| *DTDwordTimeToMillisec* | Converts a DWORD value containing time information to total amount of milliseconds |
| *DTMillisecToDwordTime* | Converts total milliseconds to a DWORD value containing time information in hours, minutes, |
| | |
| *DTDayOfWeek* | Returns an integer indicating the day of the week from a date & time string |
| *DTIsLeapYear* | Calculate if a specified year is a leap year |
| *DTDay* | Get the day part of a date & time string |
| *DTMonth* | Get the month part of a date & time string |
| *DTYear* | Get the year part of a date & time string |
| | |
| *DTDateStringsCompare* | Compare two date & time strings to determine if they are: equal in value, one is less than and |
| *DTDateStringsDifference* | Returns the difference in days of the first date & time string to the second date & time string |
| *DTDateTimeStringsDifference* | Returns the difference in days or milliseconds of the first date & time string to the second dat |

CHAPTER

THREE

# DATETIME FUNCTIONS X64

## 3.1 DTDateFormat

Converts a SYSTEMTIME structure, pointed to by the `lpSystermtimeStruct` parameter, to a date & time string, into the buffer pointed to by the `lpszDateTimeString` parameter and returns it formatted as specified by the `DateFormat` parameter.

```
DTDateFormat PROTO lpSystermtimeStruct:QWORD, lpszDateTimeString:QWORD, DateFormat:QWORD
```

**Parameters**

- `lpSystermtimeStruct` - Pointer to a SYSTEMTIME type that contains date & time information to convert.
- `lpszDateTimeString` - Pointer to a buffer to store the date & time string. The format of the date & time string is determined by the `DateFormat` parameter.
- `DateFormat` - Value indicating the date & time format to return in the buffer pointed to by `lpszDateTimeString` parameter. The parameter can contain one of the following constants as listed in the *DateTime Formats* page and as defined in the `DateTime.inc` include file:

```
; Reverse Date Formats
CCYYMMDDHHMMSSMS                    EQU 1  ; Example 1974/03/27 14:53:01:00
CCYYMMDDHHMMSS                      EQU 2  ; Example 1974/03/27 14:53:01
CCYYMMDDHHMM                        EQU 3  ; Example 1974/03/27 14:53
CCYYMMDDHH                          EQU 4  ; Example 1974/03/27 14:53
CCYYMMDD                            EQU 5  ; Example 1974/03/27
CCYYMM                              EQU 6  ; Example 1974/03
YEAR                                EQU 7  ; Example 1974

; British Date Formats
DDMMCCYYHHMMSSMS                    EQU 8  ; Example 27/03/1974 14:53:01:00
DDMMCCYYHHMMSS                      EQU 9  ; Example 27/03/1974 14:53:01
DDMMCCYYHHMM                        EQU 10 ; Example 27/03/1974 14:53
DDMMCCYY                            EQU 11 ; Example 27/03/1974
DDMM                                EQU 12 ; Example 27/03
DAY                                 EQU 13 ; Example 27

; American Date Formats
MMDDCCYYHHMMSSMS                    EQU 14 ; Example 03/27/1974 14:53:01:00
MMDDCCYYHHMMSS                      EQU 15 ; Example 03/27/1974 14:53:01
MMDDCCYYHHMM                        EQU 16 ; Example 03/27/1974 14:53
MMDDCCYY                            EQU 17 ; Example 03/27/1974
```

(continues on next page)

(continued from previous page)

```
MMDD                                EQU 18 ; Example 03/27
MONTH                               EQU 19 ; Example 03

; Reverse Without Century Date Formats
YYMMDDHHMMSSMS                      EQU 20 ; Example 74/03/27 14:53:01:00
YYMMDDHHMMSS                        EQU 21 ; Example 74/03/27 14:53:01
YYMMDDHHMM                          EQU 22 ; Example 74/03/27 14:53
YYMMDD                              EQU 23 ; Example 74/03/27
YYMM                                EQU 24 ; Example 74/03
YY                                  EQU 25 ; Example 74

; Other Date Formats
MMDDYY                              EQU 26 ; Example 03/27/74
DDMMYY                              EQU 27 ; Example 27/03/74
DAYOFWEEK                           EQU 28 ; Example Monday

; Time Formats
HHMMSSMS                            EQU 29 ; Example 14:53:01
HHMMSS                              EQU 30 ; Example 14:53:01
HHMM                                EQU 31 ; Example 14:53
HH                                  EQU 32 ; Example 14

; Useful Named Constants
TODAY                               EQU DDMMCCYYHHMMSS
NOW                                 EQU DDMMCCYYHHMMSS
TIME                                EQU HHMM

; Named Date Constants
AMERICAN                            EQU MMDDYY
BRITISH                             EQU DDMMYY
FRENCH                              EQU DDMMYY
JAPAN                               EQU YYMMDD
TAIWAN                              EQU YYMMDD
MDY                                 EQU MMDDYY
DMY                                 EQU DDMMYY
YMD                                 EQU YYMMDD
```

**Note:** Constants: CC=Century, YY=Year, MM=Month, DD=Day, HH=Hours, MM=Minutes, SS=Seconds, MS=Milliseconds, DOW=Day Of Week

**Returns**

Returns in RAX TRUE if succesful, or FALSE otherwise.

**Notes**

The user is responsible for ensuring the buffer pointed to by the lpszDateTimeString parameter is large enought to hold the date & time string value. Recommeneded minimum size for the buffer is 24 bytes long as defined by the DATETIME_STRING constant.

**Example**

```
.data
LocalDateTime SYSTEMTIME <>
szDataAndTimeString DB DATETIME_STRING DUP (0)

.code
Invoke GetLocalTime, Addr LocalDateTime
Invoke DTDateFormat, Addr LocalDateTime, Addr szDataAndTimeString, DDMMYY
```

**See Also**

*DTGetDateTime*, *DateTime Formats*

## 3.2 DTDateStringsCompare

Compare two date & time strings to determine if they are: equal in value, one is less than another, or one is greater than another.

```
DTDateStringsCompare PROTO lpszDateTimeString1:QWORD, lpszDateTimeString2:QWORD,␣
↪DateFormat:QWORD
```

**Parameters**

- `lpszDateTimeString1` - Pointer to a buffer containing the first date & time string to compare. The format of the date & time string is determined by the `DateFormat` parameter.
- `lpszDateTimeString2` - Pointer to a buffer containing the second date & time string to compare. The format of the date & time string is determined by the `DateFormat` parameter.
- `DateFormat` - Value indicating the date & time format used in the buffer pointed to by both the `lpszDateTimeString1` parameter and the `lpszDateTimeString2` parameter. The parameter can contain one of the following constants as listed in the *DateTime Formats* page and as defined in the `DateTime.inc` include file.

**Returns**

Return values in `RAX` indicate the following:

- `-1` First date is less than second date.
- `0` First date is equal to second date.
- `+1` First date is greater than second date.

**Notes**

The date & time string pointed to by the `lpszDateTimeString1` parameter is compared against the date & time string pointed to by the `lpszDateTimeString2` parameter.

Both dates are converted to `QWORD` values internally before the comparison to see which is greater or less than or equal.

If a date & time string contains time information this will be ignored.

**Example**

```
Invoke DTDateStringsCompare, Addr szDateTime1, Addr szDateTime2, CCYYMMDD
```

**See Also**

*DTDateStringsDifference*, *DTDateTimeStringsDifference*, *DateTime Formats*

## 3.3 DTDateStringsDifference

Returns the difference in days of the first date & time string to the second date & time string.

```
DTDateStringsDifference PROTO lpszDateTimeString1:QWORD, lpszDateTimeString2:QWORD,␣
↪DateFormat:QWORD
```

**Parameters**

- `lpszDateTimeString1` - Pointer to a buffer containing the first date & time string. The format of the date & time string is determined by the `DateFormat` parameter.
- `lpszDateTimeString2` - Pointer to a buffer containing the second date & time string. The format of the date & time string is determined by the `DateFormat` parameter.
- `DateFormat` - Value indicating the date & time format used in the buffer pointed to by both the `lpszDateTimeString1` parameter and the `lpszDateTimeString2` parameter. The parameter can contain one of the following constants as listed in the *DateTime Formats* page and as defined in the `DateTime.inc` include file.

**Returns**

On return `RAX` contains the difference between the two dates in days as an integer, either positive or negative.

**Notes**

The date & time string pointed to by the `lpszDateTimeString1` parameter is compared against the date & time string pointed to by the `lpszDateTimeString2` parameter to determine the difference of the dates.

Both dates are converted to Julian date format internally before comparing the days difference between dates.

If a date & time string contains time information this will be ignored.

**Example**

```
.data
CurrentDateTime db DATETIME_STRING dup (0)
SomeOtherDateTime db "2008/03/21",0

.data?
DayDifference dq ?

.code
Invoke DTGetDateTime, Addr CurrentDateTime, CCYYMMDD
Invoke DTDateStringsDifference, Addr CurrentDateTime, Addr SomeOtherDateTime, CCYYMMDD
mov DayDifference, rax ; save day difference to data variable

.IF rax == 0
     ; No difference in dates, do something...
.ELSE
     ; Day difference is - or +, do something else...
.ENDIF
```

**See Also**

*DTDateStringsCompare*, *DTDateTimeStringsDifference*, *DateTime Formats*

## 3.4 DTDateTimeStringToJulianMillisec

Converts a formatted date & time string to a Julian date integer in RAX and milliseconds in RDX

```
DTDateTimeStringToJulianMillisec PROTO lpszDateTimeString:QWORD, DateFormat:QWORD
```

**Parameters**

- lpszDateTimeString - Pointer to a buffer containing the date & time string to convert to Julian date integer and millisec values. The format of the date & time string is determined by the DateFormat parameter.
- DateFormat - Value indicating the date & time format used in the buffer pointed to by lpszDateTimeString parameter. The parameter can contain one of the following constants as listed in the *DateTime Formats* page and as defined in the DateTime.inc include file.

**Returns**

On return RAX contains Julian date integer value representing the date specified and RDX will contain the total time in milliseconds.

**Notes**

Some information may be unavailable to convert if the passed DateTime string does not contain enough information. For example if only year information is in the DateTime string, then the Julian date and millisec time will be based only on that.

To prevent this, always pass a full date time string of CCYYMMDDHHMMSSMS format or similar (DDMMCCYYHHMMSSMS or MMDDCCYYHHMMSSMS)

The Julian Day Count is a uniform count of days from a remote epoch in the past (-4712 January 1, 12 hours Greenwich Mean Time (Julian proleptic Calendar) = 4713 BCE January 1, 12 hours GMT (Julian proleptic Calendar) At this instant, the Julian Day Number is 0.

The Julian Day Count has nothing to do with the Julian Calendar introduced by Julius Caesar. It is named for Julius Scaliger, the father of Josephus Justus Scaliger, who invented the concept. It can also be thought of as a logical follow-on to the old Egyptian civil calendar, which also used years of constant lengths.

Scaliger chose the particular date in the remote past because it was before recorded history and because in that year, three important cycles coincided with their first year of the cycle: The 19-year Metonic Cycle, the 15-year Indiction Cycle (a Roman Taxation Cycle) and the 28-year Solar Cycle (the length of time for the old Julian Calendar to repeat exactly).

**Example**

```
.data
DateTimeStringValue db "2008/03/21 16:21:01:00",0

.data?
JulianDateValue dq ?
TotalMilliseconds dq ?

.code
Invoke DTDateTimeStringToJulianMillisec, Addr DateTimeStringValue
mov JulianDateValue, rax ; save julian date value to data variable
mov TotalMilliseconds, rdx ; save total milliseconds to data variable
```

**Example**

```
.data?
qwDate1 dq ?
qwDate2 dq ?

.code
;
↪=============================================================================================
; Example to calculate difference in 2 date values over month boundary
Invoke DTDateTimeStringToJulianMillisec, CTEXT("27/02/2009"), DDMMCCYY
mov qwDate1, rax ; save dword date portion to variable, ignore edx which contains time␣
↪info

Invoke DTDateTimeStringToJulianMillisec, CTEXT("03/03/2009"), DDMMCCYY
mov qwDate2, rax ; save dword date portion to variable, ignore edx which contains time␣
↪info

mov rbx, qwDate1 ; Place date1 in rbx
sub rax, rbx ; subtract date2 from date1, rax = 4 days

;
↪=============================================================================================
; Another example to calculate difference in weeks between 2 date values
Invoke DTDateTimeStringToJulianMillisec, CTEXT("03/01/2009"), DDMMCCYY
mov qwDate1, rax ; save qword date portion to variable, ignore rdx which contains time␣
↪info

Invoke DTDateTimeStringToJulianMillisec, CTEXT("10/01/2009"), DDMMCCYY
mov qwDate2, rax ; save dword date portion to variable, ignore rdx which contains time␣
↪info

mov rbx, qwDate1 ; Place date1 in rbx
sub rax, rbx ; subtract date2 from date1, rax = 7 days
xor rdx, rdx ; clear rdx for division
mov rcx, 7d ; Divisor is 7 in rcx
div rcx ; divide eax by 7, rax = 1
```

**See Also**

*DTJulianMillisecToDateTimeString*

## 3.5 DTDateTimeStringToQwordDateTime

Converts a formatted date & time string to `QWORD` values. On return `RAX` contains the **date** information and `RDX` the **time** information.

```
DTDateTimeStringToQwordDateTime PROTO lpszDateTimeString:QWORD, DateFormat:QWORD
```

**Parameters**

- `lpszDateTimeString` - Pointer to a buffer containing the date & time string to convert to `QWORD` values. The format of the date & time string is determined by the `DateFormat` parameter.
- `DateFormat` - Value indicating the date & time format used in the buffer pointed to by `lpszDateTimeString`

parameter. The parameter can contain one of the following constants as listed in the *DateTime Formats* page and as defined in the `DateTime.inc` include file.

**Returns**

On return `RAX` will contain the **date** information in the following format:

`RAX` Register Bits:

| DWORD | | | | WORD | | | |
|---|---|---|---|---|---|---|---|
| Bits 63-32 | | | | Bits 31-16 | | | |
| Not used - Not applicable | | | | Century Year | | | |
| N/A | | | | CCCCYY | | | |

| BYTE | | | | BYTE | | | |
|---|---|---|---|---|---|---|---|
| Bits 15-8 | | | | Bits 7-0 | | | |
| Month | | | | Day | | | |
| MM | | | | DD | | | |

On return `RDX` will contain the **time** information in the following format:

`RDX` Register Bits:

| DWORD | | | | BYTE | | | |
|---|---|---|---|---|---|---|---|
| Bits 63-32 | | | | Bits 31-23 | | | |
| Not used - Not applicable | | | | Hour | | | |
| N/A | | | | HH | | | |

| BYTE | | | | BYTE | | | |
|---|---|---|---|---|---|---|---|
| Bits 15-8 | | | | Bits 7-0 | | | |
| Second | | | | Millisec | | | |
| SS | | | | MS | | | |

**Notes**

The registers always return the information in the format shown in the tables above, and do not change regardless of the `DateFormat` parameter.

The `DateFormat` parameter is used to indicate the format of the date & time string being passed to

DTDateTimeStringToQwordDateTime.

Some information may be unavailable to convert if the passed DateTime string does not contain enough information. For example if only year information is in the DateTime string, then the DWORD **date** and DWORD **time** will be based only on that.

To prevent this, always pass a full date time string of CCYYMMDDHHMMSSMS format or similar (DDMMCCYYHHMMSSMS or MMDDCCYYHHMMSSMS)

To retrieve the various values of the date & time QWORD values you can use the following as an example:

**Example**

```
.data
DateTimeStringValue db "2008/03/21 16:21:01:00",0

.data?
DateValue dq ?
TimeValue dq ?
CentYear dq ?
Year dq ?
Month dq ?
Day dq ?

.code
Invoke DTDateTimeStringToQwordDateTime, Addr DateTimeStringValue, CCYYMMDDHHMMSSMS
mov DateValue, rax ; Save date info in rax to a data variable
mov TimeValue, rdx ; Save time info in rdx to a data variable

; save day, month, year and century info to data variables
mov rdx, DateValue ; use rdx as our work place

xor rax, rax ; clear register
mov al, dl ; dl contains day info
mov Day, rax ; save day info DD in dl to a data variable

xor rax, rax ; clear register
mov al, dh ; dh contains month info
mov Month, rax ; save month info MM in dh to a data variable

shr edx, 16 ; century and year are now in dx part of the register
mov CentYear, rdx ; save century and year info CCYY to a data variable

xor rax, rax ; clear register
mov al, dl ; dl contains YY part of year
mov Year, rax ; save year info in dl to a data variable
```

**See Also**

*DTQwordDateTimeToDateTimeString*, *DateTime Formats*, *DTGetDateTime*

## 3.6 DTDateTimeStringToUnixTime

Converts a formatted date & time string to a QWORD value containing a unix time integer value.

```
DTDateTimeStringToUnixTime PROTO lpszDateTimeString:QWORD, DateFormat:QWORD
```

**Parameters**

- lpszDateTimeString - Pointer to a buffer containing the date & time string to convert to a QWORD value containing a unix time integer value. The format of the date & time string is determined by the DateFormat parameter.
- DateFormat - Value indicating the date & time format used in the buffer pointed to by lpszDateTimeString parameter. The parameter can contain one of the following constants as listed in the *DateTime Formats* page and as defined in the DateTime.inc include file.

**Returns**

On return RAX contains the unix time as a QWORD value.

**Notes**

Some information may be unavailable to convert if the passed DateTime string does not contain enough information. For example if only year information is in the DateTime string, then the UnixTime will be based only on that.

To prevent this, always pass a full date time string of CCYYMMDDHHMMSSMS format or similar (DDMMCCYYHHMMSSMS or MMDDCCYYHHMMSSMS)

Unix time is defined as the number of seconds elapsed since 00:00 Universal time on January 1, 1970 in the Gregorian calendar (Julian day 2440587.5)

The UNIXTIMESTAMP format is a **string** representation of the unix time in integer format if used as the DateFormat value.

**Example**

```
.data
DateTimeStringValue db "2008/03/21 16:21:01:00",0

.data?
UnixTime dq ?

.code
Invoke DTDateTimeStringToUnixTime, Addr DateTimeStringValue, CCYYMMDDHHMMSSMS
mov UnixTime, rax ; save unix time value to data variable which should contain 1206116461
```

**Example**

```
.data
UnixTimeAsAStringValue db "1656241202",0 ; Sun Jun 26 2022 11:00

.data?
UnixTime dq ?

.code
Invoke DTDateTimeStringToUnixTime, Addr UnixTimeAsAStringValue, UNIXTIMESTAMP
mov UnixTime, rax ; save unix time value to data variable which should contain 1656241202
```

**See Also**

*DTUnixTimeToDateTimeString*, *DTUnixTimeToQwordDateTime*, *DateTime Formats*

## 3.7 DTDateTimeStringsDifference

Returns the difference in days or milliseconds of the first date & time string to the second date & time string.

```
DTDateTimeStringsDifference PROTO lpszDateTimeString1:QWORD, lpszDateTimeString2:QWORD,␣
↪DateFormat:QWORD
```

**Parameters**

- `lpszDateTimeString1` - Pointer to a buffer containing the first date & time string. The format of the date & time string is determined by the `DateFormat` parameter.
- `lpszDateTimeString2` - Pointer to a buffer containing the second date & time string. The format of the date & time string is determined by the `DateFormat` parameter.
- `DateFormat` - Value indicating the date & time format used in the buffer pointed to by both the `lpszDateTimeString1` parameter and the `lpszDateTimeString2` parameter. The parameter can contain one of the following constants as listed in the *DateTime Formats* page and as defined in the `DateTime.inc` include file.

**Returns**

On return `RAX` contains the difference between the two dates in days as an integer, either positive or negative.

On return `RDX` contains the difference in time or `0`, see notes below for details.

**Notes**

If both dates are the same date, then `RAX` will be `0`. If both date & time strings contain time values and the `DateFormat` parameter specified also provides a time format, then `RDX` will contain the difference in time in milliseconds.

**Example**

```
.data
CurrentDateTime db DATETIME_STRING dup (0)
SomeOtherDateTime db "2008/03/21",0

.data?
DayDifference dq ?
MillisecDifference dq ?

.code
Invoke DTGetDateTime, Addr CurrentDateTime, CCYYMMDDHHMMSSMS
Invoke DTDateTimeStringsDifference, Addr CurrentDateTime, Addr SomeOtherDateTime,␣
↪CCYYMMDDHHMMSSMS
mov DayDifference, rax ; save day difference to data variable
mov MillisecDifference, rdx ; save millisec diff if days the same

.IF rax == 0
     ; No difference in dates
     ; So MillisecDifference can be used
.ELSE
     ; Day difference is - or +, do something else...
     ; MillisecDifference is 0 value
.ENDIF
```

**See Also**

*DTDateStringsCompare*, *DTDateStringsDifference*, *DateTime Formats*

## 3.8 DTDay

Get the day part of a date & time string.

```
DTDay PROTO lpszDateTimeString:QWORD, DateFormat:QWORD
```

**Parameters**

- `lpszDateTimeString` - Pointer to a buffer containing the date & time string to retrieve the **day** portion from. The format of the date & time string is determined by the `DateFormat` parameter.
- `DateFormat` - Value indicating the date & time format used in the buffer pointed to by `lpszDateTimeString` parameter. The parameter can contain one of the following constants as listed in the *DateTime Formats* page and as defined in the `DateTime.inc` include file.

**Returns**

Returns the day value in `RAX`

**Example**

```
.data
DateTimeStringValue db "2008/03/21 16:21:01:00",0

.data?
DayValue dq ?

.code
Invoke DTDay, Addr DateTimeStringValue, CCYYMMDDHHMMSSMS
mov DayValue, rax ; save day part of date value to data variable
; rax should contain 21
```

**See Also**

*DTMonth*, *DTYear*, *DTIsLeapYear*, *DTDayOfWeek*, *DateTime Formats*

## 3.9 DTDayOfWeek

Returns an integer indicating the day of the week from a date & time string.

```
DTDayOfWeek PROTO lpszDateTimeString:QWORD, DateFormat:QWORD
```

**Parameters**

- `lpszDateTimeString` - Pointer to a buffer containing the date & time string to check the day of the week for. The format of the date & time string is determined by the `DateFormat` parameter.
- `DateFormat` - Value indicating the date & time format used in the buffer pointed to by `lpszDateTimeString` parameter. The parameter can contain one of the following constants as listed in the *DateTime Formats* page and as defined in the `DateTime.inc` include file.

**Returns**

On return `RAX` will contain a value to indicate the following day:

- `0` = Monday
- 1 = Tuesday

- 2 = Wednesday
- 3 = Thursday
- 4 = Friday
- 5 = Saturday
- 6 = Sunday

**Example**

```
; The following example calculates the first day of the week from a date using the␣
↪DTDayOfWeek function
.data
szDateTime db DATETIME_STRING dup (0)

.data?
qwDate1 dq ?
qwDayWeek dq ?
qwStartWeek dq ?

.code
Invoke DTDateTimeStringToJulianMillisec, CTEXT("04/06/2009"), DDMMCCYY ; Convert date to␣
↪julian date
mov qwDate1, rax ; only save date info returned in eax, ignore time info returned in edx

Invoke DTDayOfWeek, CTEXT("04/06/2009"), DDMMCCYY ; Get day of week for date
mov qwDayWeek, rax ; Save the returned integer representing day 0-6

mov rbx, rax
mov rax, qwDate1
sub rax, rbx ; subtract julian date value from day of week integer
mov qwStartWeek, rax ; we have new julian date which is start of the week

Invoke DTJulianToDwordDate, qwStartWeek ; convert new julian date start of week number␣
↪to a date in dword format
Invoke DTDwordDateTimeToDate, rax, NULL, Addr szDateTime, DDMMCCYY ; convert dword date␣
↪format to a string value
; szDateTime now contains '01/06/2009'
```

**See Also**

*DTDay*, *DTMonth*, *DTYear*, *DTIsLeapYear*, *DateTime Formats*

## 3.10 DTGetDateTime

Get the current date & time and return it as a formatted string as specified by the `DateFormat` parameter.

```
DTGetDateTime PROTO lpszDateTimeString:QWORD, DateFormat:QWORD
```

**Parameters**

- `lpszDateTimeString` - Pointer to a buffer to store the date & time string. The format of the date & time string is determined by the `DateFormat` parameter.

- DateFormat - Value indicating the date & time format to return in the buffer pointed to by lpszDateTimeString parameter. The parameter can contain one of the following constants as listed in the *DateTime Formats* page and as defined in the DateTime.inc include file.

**Returns**

Returns in RAX TRUE if succesful, or FALSE otherwise.

**Notes**

The user is responsible for ensuring the buffer pointed to by the lpszDateTimeString parameter is large enought to hold the date & time string value. Recommeneded minimum size for the buffer is 24 bytes long as defined by the DATETIME_STRING constant.

**Example**

```
.data
DateTimeStringValue db DATETIME_STRING dup (0) ; buffer to store date and time as string

.code
Invoke DTGetDateTime, Addr DateTimeStringValue, CCYYMMDDHHMMSS
```

**See Also**

*DTDateFormat*, *DateTime Formats*

## 3.11 DTIsLeapYear

Calculate if a specified year is a leap year.

```
DTIsLeapYear PROTO qwYear:QWORD
```

**Parameters**

- qwYear - a QWORD value that has the year value to check for leap year.

**Returns**

Returns TRUE if year is a leap year, otherwise FALSE.

**Notes**

qwYear must include the century. For example 2008 which in a QWORD value is stored as 0x7D8

A year will be a leap year if it is divisible by 4 but not by 100. If a year is divisible by 4 and by 100, it is not a leap year unless it is also divisible by 400.

**Example**

```
.data
CurrentDateTimeValue db 32 dup (0) ; buffer to store date & time as string
Year dq 0

.code
Invoke DTGetDateTime, Addr CurrentDateTimeValue, CCYYMMDDHHMMSS
Invoke DTYear, Addr CurrentDateTimeValue, CCYYMMDDHHMMSS
mov Year, rax ; save year

Invoke DTIsLeapYear, rax ; check leap year
```

(continues on next page)

(continued from previous page)

```
.IF rax == TRUE
     ; Leap year, do something...
.ELSE
     ; No leap year, do something other...
.ENDIF
```

**See Also**

*DTDay*, *DTMonth*, *DTYear*, *DTDayOfWeek*

## 3.12 DTJulianDateToQwordDate

Converts a Julian Date number to a QWORD value containing date information. Uses CCYYMMDD *DateTime Format*

```
DTJulianDateToQwordDate PROTO JulianDate:QWORD
```

**Parameters**

- JulianDate - A QWORD value containing the Julian date integer to convert to a QWORD format in RAX

**Returns**

On return RAX will contain the **date** information in the following format:

RAX Register Bits:

```
+-------------------------------------------------+-----------------------+--
↪----------+-----------+
| DWORD                                           | WORD                  |␣
↪BYTE       | BYTE      |
+-------------------------------------------------+-----------------------+--
↪----------+-----------+
| Bits 63-32                                      | Bits 31-16            |␣
↪Bits 15-8  | Bits 7-0  |
+-------------------------------------------------+-----------------------+--
↪----------+-----------+
| Not used - Not applicable                       | Century Year          |␣
↪Month      | Day       |
+-------------------------------------------------+-----------------------+--
↪----------+-----------+
| N/A                                             | CCCCYY                |␣
↪MM         | DD        |
+-------------------------------------------------+-----------------------+--
↪----------+-----------+
```

**Notes**

The Julian Day Count is a uniform count of days from a remote epoch in the past (-4712 January 1, 12 hours Greenwich Mean Time (Julian proleptic Calendar) = 4713 BCE January 1, 12 hours GMT (Julian proleptic Calendar) At this instant, the Julian Day Number is 0.

The Julian Day Count has nothing to do with the Julian Calendar introduced by Julius Caesar. It is named for Julius Scaliger, the father of Josephus Justus Scaliger, who invented the concept. It can also be thought of as a logical follow-on to the old Egyptian civil calendar, which also used years of constant lengths.

Scaliger chose the particular date in the remote past because it was before recorded history and because in that year, three important cycles coincided with their first year of the cycle: The 19-year Metonic Cycle, the 15-year Indiction Cycle (a Roman Taxation Cycle) and the 28-year Solar Cycle (the length of time for the old Julian Calendar to repeat exactly).

**Example**

```
.data
Day db 0
Month db 0
CentYear dw 0

.code
Invoke DTJulianDateToQwordDate, 2299161 ; Gregorian date of this Julian Date value is␣
↪1582 October 15

mov rdx, rax
mov Month, ah
mov Day, al
shr rax, 16 ; move year info into lower word of eax
mov CentYear, ax

; 1582 October 15 - CentYear now contains 1582, Month contains 10, Day contains 15
```

**See Also**

*DTQwordDateToJulianDate*

## 3.13 DTJulianMillisecToDateTimeString

Converts a Julian date integer and millisec time to a formatted date & time string as specified by the `DateFormat` parameter.

```
DTJulianMillisecToDateTimeString PROTO JulianDate:QWORD, Milliseconds:QWORD,␣
↪lpszDateTimeString:QWORD, DateFormat:QWORD
```

**Parameters**

- `JulianDate` - A `QWORD` value containing the Julian date integer to convert to a gregorian date in the date & time string.
- `Milliseconds` - A `QWORD` value containting the total milliseconds to convert to hours, minutes, seconds and milliseconds in the date & time string.
- `lpszDateTimeString` - Pointer to a buffer to store the date & time string. The format of the date & time string is determined by the `DateFormat` parameter.
- `DateFormat` - Value indicating the date & time format to return in the buffer pointed to by `lpszDateTimeString` parameter. The parameter can contain one of the following constants as listed in the *DateTime Formats* page and as defined in the `DateTime.inc` include file.

**Returns**

There is no return value, the date & time string will contain the date & time as specified by the `DateFormat` parameter.

**Notes**

No time information is converted with the Julian date value, so if formatting outputs a time, it will be empty as in `00:00:00:00`

**Example**

```
.data
DateTimeStringValue db DATETIME_STRING dup (0) ; buffer to store date and time as string

.code
Invoke DTJulianMillisecToDateTimeString, 2299161, 0, Addr DateTimeStringValue,␣
↪CCYYMMDDHHMMSS
; DateTimeStringValue now contains: 1582/10/15 00:00:00
```

**See Also**

*DTDateTimeStringToJulianMillisec*, *DateTime Formats*

## 3.14 DTJulianMillisecToQwordDateTime

Converts Julian date integer value and total milliseconds to QWORD values containing date and time information. On return RAX contains the **date** information and RDX the **time** information.

```
DTJulianMillisecToQwordDateTime PROTO JulianDate:QWORD, Milliseconds:QWORD
```

**Parameters**

- `JulianDate` - A QWORD value containing the Julian date integer to convert to a QWORD date in RAX.
- `Milliseconds` - A QWORD value containting the total milliseconds to convert to hours, minutes, seconds and milliseconds in QWORD time in RDX.

**Returns**

On return RAX will contain the **date** information in the following format:

RAX Register Bits:

```
+-----------------------------------------------+-----------------------+--
↪----------+----------+
| DWORD                                         | WORD                  |␣
↪BYTE       | BYTE      |
+-----------------------------------------------+-----------------------+--
↪----------+----------+
| Bits 63-32                                    | Bits 31-16            |␣
↪Bits 15-8  | Bits 7-0  |
+-----------------------------------------------+-----------------------+--
↪----------+----------+
| Not used - Not applicable                     | Century Year          |␣
↪Month      | Day       |
+-----------------------------------------------+-----------------------+--
↪----------+----------+
| N/A                                           | CCCCYY                |␣
↪MM         | DD        |
+-----------------------------------------------+-----------------------+--
↪----------+----------+
```

On return RDX will contain the **time** information in the following format:

RDX Register Bits:

| DWORD | | | | BYTE | BYTE | BYTE | BYTE |
|---|---|---|---|---|---|---|---|
| Bits 63-32 | | | | Bits 31-23 | Bits 23-16 | Bits 15-8 | Bits 7-0 |
| Not used - Not applicable | | | | Hour | Minute | Second | Millisec |
| N/A | | | | HH | MM | SS | MS |

**Notes**

The registers always return the information in the format shown in the tables above.

**Example**

```
.data?
qwDate dq ?
qwTime dq ?
Year dw ?
Month db ?
Day db ?
Hours db ?
Minutes db ?
Seconds db ?
Millisec db ?

.code
Invoke DTJulianMillisecToDwordDateTime, 2299161, 0
mov qwDate, rax ; rax contains the date information
mov qwTime, rdx ; rdx contains the time information

mov rax, qwDate
shr rax, 16 ; move year into upper word of eax
mov Year,  ax ; year
mov rax, qwDate
mov Month, ah ; month
mov Day, al ; day

mov rax, qwTime
shr rax, 16 ; move hour and minute into upper word of eax
mov Hours, ah ; hours
```

(continues on next page)

(continued from previous page)

```
mov Minutes, al ; minutes
mov rax, qwTime
mov Seconds, ah ; seconds
mov Millisec, al ; millseconds
```

**See Also**

*DTQwordDateTimeToJulianMillisec*, *DTQwordDateToJulianDate*, *DTQwordTimeToMillisec*

## 3.15 DTMillisecToQwordTime

Converts total milliseconds to a QWORD value containing time information in hours, minutes, seconds and milliseconds.

```
DTMillisecToQwordTime PROTO Milliseconds:QWORD
```

**Parameters**

- Milliseconds - A QWORD value containting the total milliseconds to convert to hours, minutes, seconds and milliseconds in QWORD time in RAX.

**Returns**

On return RAX will contain the **time** information in the following format:

RAX Register Bits:

```
+-------------------------------------------------+-----------+-----------+-
↪----------+----------+
| DWORD                                           | BYTE      | BYTE      |␣
↪BYTE      | BYTE     |
+-------------------------------------------------+-----------+-----------+-
↪----------+----------+
| Bits 63-32                                      | Bits 31-23 | Bits 23-16 |␣
↪Bits 15-8 | Bits 7-0  |
+-------------------------------------------------+-----------+-----------+-
↪----------+----------+
| Not used - Not applicable                       | Hour      | Minute    |␣
↪Second    | Millisec  |
+-------------------------------------------------+-----------+-----------+-
↪----------+----------+
| N/A                                             | HH        | MM        |␣
↪SS        | MS        |
+-------------------------------------------------+-----------+-----------+-
↪----------+----------+
```

**Notes**

**Example**

```
Invoke DTMillisecToQwordTime, qwMilliseconds
```

**See Also**

*DTQwordTimeToMillisec*, *DTQwordDateToJulianDate*, *DTQwordDateTimeToJulianMillisec*, *DTJulianMillisecToQwordDateTime*

## 3.16 DTMonth

Get the month part of a date & time string.

```
DTMonth PROTO lpszDateTimeString:QWORD, DateFormat:QWORD
```

**Parameters**

- `lpszDateTimeString` - Pointer to a buffer containing the date & time string to retrieve the **month** portion from. The format of the date & time string is determined by the `DateFormat` parameter.
- `DateFormat` - Value indicating the date & time format used in the buffer pointed to by `lpszDateTimeString` parameter. The parameter can contain one of the following constants as listed in the *DateTime Formats* page and as defined in the `DateTime.inc` include file.

**Returns**

Returns the month value in `RAX`

**Example**

```
.data
DateTimeStringValue db "2008/03/21 16:21:01:00",0

.data?
MonthValue dq ?

.code
Invoke DTMonth, Addr DateTimeStringValue, CCYYMMDDHHMMSSMS
mov MonthValue, rax ; save month part of date value to data variable
; rax should contain 3
```

**Example**

```
.data?
qwMonth1 dq ?
qwMonth2 dq ?

.code
Invoke DTMonth, CTEXT("27/02/2009"), DDMMCCYY
mov qwMonth1, rax ; save month to variable

Invoke DTMonth, CTEXT("03/03/2009"), DDMMCCYY
mov qwMonth2, rax ; save month to variable

mov rbx, qwMonth1 ; place month1 variable in rbx
sub rax, rbx ; subtract month2 from month1, rax = 1 in this example
```

**See Also**

*DTDay*, *DTYear*, *DTIsLeapYear*, *DTDayOfWeek*, *DateTime Formats*

## 3.17 DTQwordDateTimeToDateTimeString

Converts QWORD values containing date & time information to a formatted date & time string as specified by the DateFormat parameter.

```
DTQwordDateTimeToDateTimeString PROTO qwDate:QWORD, qwTime:QWORD,␣
↪lpszDateTimeString:QWORD, DateFormat:QWORD
```

**Parameters**

- qwDate - QWORD value containing **Date** information to convert to the date & time string

  The format for the value containing the **date** information is as follows:

  **date** QWORD Register Bits:

```
+-------------------------------------------------+----------------------+--
↪---------+----------+
| DWORD                                           | WORD                 |␣
↪BYTE       | BYTE      |
+-------------------------------------------------+----------------------+--
↪---------+----------+
| Bits 63-32                                      | Bits 31-16           |␣
↪Bits 15-8  | Bits 7-0  |
+-------------------------------------------------+----------------------+--
↪---------+----------+
| Not used - Not applicable                       | Century Year         |␣
↪Month      | Day       |
+-------------------------------------------------+----------------------+--
↪---------+----------+
| N/A                                             | CCCCYY               |␣
↪MM         | DD        |
+-------------------------------------------------+----------------------+--
↪---------+----------+
```

- qwTime - QWORD value containing **Time** information to convert to the date & time string

  The format for the value containing the **time** information is as follows:

  **time** QWORD Register Bits:

```
+-------------------------------------------------+----------+----------+-
↪---------+----------+
| DWORD                                           | BYTE     | BYTE     |␣
↪BYTE      | BYTE      |
+-------------------------------------------------+----------+----------+-
↪---------+----------+
| Bits 63-32                                      | Bits 31-23 | Bits 23-16 |␣
↪Bits 15-8 | Bits 7-0  |
+-------------------------------------------------+----------+----------+-
↪---------+----------+
| Not used - Not applicable                       | Hour     | Minute   |␣
↪Second    | Millisec  |
+-------------------------------------------------+----------+----------+-
↪---------+----------+
| N/A                                             | HH       | MM       |␣
↪SS        | MS        |
```

(continues on next page)

(continued from previous page)

```
+-------------------------------------------------+-----------+-----------+-
↪----------+-----------+
```

- `lpszDateTimeString` - Pointer to a buffer to store the date & time string. The format of the date & time string is determined by the `DateFormat` parameter.
- `DateFormat` - Value indicating the date & time format to return in the buffer pointed to by `lpszDateTimeString` parameter. The parameter can contain one of the following constants as listed in the *DateTime Formats* page and as defined in the `DateTime.inc` include file.

**Returns**

No return value

**Notes**

**Example**

```
.data
DateTimeStringValue db DATETIME_STRING dup (0) ; buffer to store date and time as string
DateValue dq 0
TimeValue dq 0

.code
xor rax, rax
mov rax, 1974 ; save year
shl rax, 16 ; move it into upper word of rax
mov ah, 03 ; save month
mov al, 15 ; save day
mov DateValue, rax
xor rax, rax
mov ah, 12 ; save hours
mov al, 34 ; save minutes
shl rax, 16 ; move it into upper word of rax
mov ah, 01 ; save seconds
mov al, 0 ; save milliseconds
mov TimeValue, rax

Invoke DTQwordDateTimeToDateTimeString, DateValue, TimeValue, Addr DateTimeStringValue,␣
↪CCYYMMDDHHMMSS
; DateTimeStringValue will now contain "1974/03/15 12:36:01"
```

**See Also**

*DTDateTimeStringToQwordDateTime*, *DateTime Formats*, *DTGetDateTime*

## 3.18 DTQwordDateTimeToJulianMillisec

Converts QWORD values containing date & time information to Julian date and total milliseconds.

```
DTQwordDateTimeToJulianMillisec PROTO qwDate:QWORD, qwTime:QWORD
```

**Parameters**

- qwDate - QWORD value containing **date** information to convert to Julian date.

  The format for the value containing the **date** information is as follows:

  **date** QWORD Register Bits:

```
+-----------------------------------------------+-----------------------+--
↪----------+-----------+
| DWORD                                         | WORD                  |␣
↪BYTE       | BYTE      |
+-----------------------------------------------+-----------------------+--
↪----------+-----------+
| Bits 63-32                                    | Bits 31-16            |␣
↪Bits 15-8  | Bits 7-0  |
+-----------------------------------------------+-----------------------+--
↪----------+-----------+
| Not used - Not applicable                     | Century Year          |␣
↪Month      | Day       |
+-----------------------------------------------+-----------------------+--
↪----------+-----------+
| N/A                                           | CCCCYY                |␣
↪MM         | DD        |
+-----------------------------------------------+-----------------------+--
↪----------+-----------+
```

- qwTime - QWORD value containing **time** information to convert to the total milliseconds.

  The format for the value containing the **time** information is as follows:

  **time** QWORD Register Bits:

```
+-----------------------------------------------+-----------+-----------+-
↪----------+-----------+
| DWORD                                         | BYTE      | BYTE      |␣
↪BYTE      | BYTE      |
+-----------------------------------------------+-----------+-----------+-
↪----------+-----------+
| Bits 63-32                                    | Bits 31-23 | Bits 23-16 |␣
↪Bits 15-8 | Bits 7-0  |
+-----------------------------------------------+-----------+-----------+-
↪----------+-----------+
| Not used - Not applicable                     | Hour      | Minute    |␣
↪Second    | Millisec  |
+-----------------------------------------------+-----------+-----------+-
↪----------+-----------+
| N/A                                           | HH        | MM        |␣
↪SS        | MS        |
+-----------------------------------------------+-----------+-----------+-
↪----------+-----------+
```

(continues on next page)

(continued from previous page)

**Returns**

On return RAX will contain the Julian date integer, and RDX the total millliseconds.

**Notes**

**Example**

```
Invoke DTQwordDateTimeToJulianMillisec, qwDate, qwTime
```

**See Also**

*DTJulianMillisecToQwordDateTime*, *DTJulianDateToQwordDate*, *DTMillisecToQwordTime*

## 3.19 DTQwordDateTimeToUnixTime

Converts QWORD values containing date & time information to a QWORD value containing a unix time integer value. On return RAX contains the unix time integer value.

```
DTQwordDateTimeToUnixTime PROTO qwDate:QWORD, qwTime:QWORD
```

**Parameters**

- qwDate - QWORD value containing **date** information to convert to a unix time integer value.

  The format for the value containing the **date** information is as follows:

  **date** QWORD Register Bits:

```
+-----------------------------------------------+-----------------------+--
↪----------+-----------+
| DWORD                                         | WORD                  |␣
↪BYTE       | BYTE      |
+-----------------------------------------------+-----------------------+--
↪----------+-----------+
| Bits 63-32                                    | Bits 31-16            |␣
↪Bits 15-8  | Bits 7-0  |
+-----------------------------------------------+-----------------------+--
↪----------+-----------+
| Not used - Not applicable                     | Century Year          |␣
↪Month      | Day       |
+-----------------------------------------------+-----------------------+--
↪----------+-----------+
| N/A                                           | CCCCYY                |␣
↪MM         | DD        |
+-----------------------------------------------+-----------------------+--
↪----------+-----------+
```

- qwTime - QWORD value containing **time** information to convert to a unix time integer value.

  The format for the value containing the **time** information is as follows:

  **time** QWORD Register Bits:

```
+-------------------------------------------------+----------+----------+-
↪---------+----------+
| DWORD                                           | BYTE     | BYTE     |␣
↪BYTE     | BYTE     |
+-------------------------------------------------+----------+----------+-
↪---------+----------+
| Bits 63-32                                      | Bits 31-23 | Bits 23-16 |␣
↪Bits 15-8 | Bits 7-0  |
+-------------------------------------------------+----------+----------+-
↪---------+----------+
| Not used - Not applicable                       | Hour     | Minute   |␣
↪Second   | Millisec  |
+-------------------------------------------------+----------+----------+-
↪---------+----------+
| N/A                                             | HH       | MM       |␣
↪SS       | MS        |
+-------------------------------------------------+----------+----------+-
↪---------+----------+
```

**Returns**

On return RAX will contain the unix time integer value.

**Notes**

Unix time is defined as the number of seconds elapsed since 00:00 Universal time on January 1, 1970 in the Gregorian calendar (Julian day 2440587.5)

**Example**

```
Invoke DTQwordDateTimeToUnixTime, qwDate, qwTime
```

**See Also**

*DTUnixTimeToQwordDateTime*, *DTDateTimeStringToUnixTime*, *DTUnixTimeToDateTimeString*

## 3.20 DTQwordDateToJulianDate

Converts a QWORD value containing date information to a Julian date integer. On return RAX contains the date information in Julian Date format.

```
DTQwordDateToJulianDate PROTO qwDate:QWORD
```

**Parameters**

- qwDate - QWORD value containing **Date** information to convert to a Julian date integer.

  The format for the value containing the **date** information is as follows:

  **date** QWORD Register Bits:

```
+-------------------------------------------------+----------------------+--
↪---------+----------+
| DWORD                                           | WORD                 |␣
↪BYTE     | BYTE     |
+-------------------------------------------------+----------------------+--
↪---------+----------+
```

(continues on next page)

(continued from previous page)

```
| Bits 63-32                                          | Bits 31-16              |␣
↪Bits 15-8  | Bits 7-0  |
+---------------------------------------------------+-----------------------+--
↪----------+-----------+
| Not used - Not applicable                          | Century Year            |␣
↪Month      | Day       |
+---------------------------------------------------+-----------------------+--
↪----------+-----------+
| N/A                                                | CCCCYY                  |␣
↪MM         | DD        |
+---------------------------------------------------+-----------------------+--
↪----------+-----------+
```

**Returns**

On return RAX contains Julian date integer value representing the date specified.

**Notes**

The Julian Day Count is a uniform count of days from a remote epoch in the past (-4712 January 1, 12 hours Greenwich Mean Time (Julian proleptic Calendar) = 4713 BCE January 1, 12 hours GMT (Julian proleptic Calendar) At this instant, the Julian Day Number is 0.

The Julian Day Count has nothing to do with the Julian Calendar introduced by Julius Caesar. It is named for Julius Scaliger, the father of Josephus Justus Scaliger, who invented the concept. It can also be thought of as a logical follow-on to the old Egyptian civil calendar, which also used years of constant lengths.

Scaliger chose the particular date in the remote past because it was before recorded history and because in that year, three important cycles coincided with their first year of the cycle: The 19-year Metonic Cycle, the 15-year Indiction Cycle (a Roman Taxation Cycle) and the 28-year Solar Cycle (the length of time for the old Julian Calendar to repeat exactly).

**Example**

```
.data
Day db 15
Month db 10
CentYear dw 1582
DateValue dq 0

.code
xor rax, rax
mov ax, CentYear
shl rax, 16 ; move year into upper word of rax
mov ah, Month ; move month into ah
mov al, Day ; move day into al
mov DateValue, rax

Invoke DTQwordDateToJulianDate, DateValue
; EAX now contains 2299161 which is the Julian Date for 1582 October 15
```

**See Also**

*DTJulianDateToQwordDate*, *DTMillisecToQwordTime*, *DTQwordDateTimeToJulianMillisec*, *DTJulianMillisecToQwordDateTime*

## 3.21 DTQwordTimeToMillisec

Converts a QWORD value containing time information to total amount of milliseconds.

```
DTQwordTimeToMillisec PROTO dwTime:QWORD
```

**Parameters**

- qwTime - QWORD value containing **time** information to convert to total amount of milliseconds.

  The format for the value containing the **time** information is as follows:

  **time** QWORD Register Bits:

```
+-------------------------------------------------+-----------+-----------+-
↪----------+-----------+
| DWORD                                           | BYTE      | BYTE      |␣
↪BYTE      | BYTE      |
+-------------------------------------------------+-----------+-----------+-
↪----------+-----------+
| Bits 63-32                                      | Bits 31-23 | Bits 23-16 |␣
↪Bits 15-8 | Bits 7-0  |
+-------------------------------------------------+-----------+-----------+-
↪----------+-----------+
| Not used - Not applicable                       | Hour      | Minute    |␣
↪Second    | Millisec  |
+-------------------------------------------------+-----------+-----------+-
↪----------+-----------+
| N/A                                             | HH        | MM        |␣
↪SS        | MS        |
+-------------------------------------------------+-----------+-----------+-
↪----------+-----------+
```

**Returns**

On return RAX will contain the total milliseconds.

**Notes**

**Example**

```
Invoke DTQwordTimeToMillisec, qwTime
```

**See Also**

*DTMillisecToQwordTime*, *DTJulianDateToQwordDate*, *DTQwordDateTimeToJulianMillisec*, *DTJulianMillisecToQwordDateTime*

## 3.22 DTUnixTimeToDateTimeString

Converts a QWORD containing a unix time integer value to a formatted date & time string as specified by the DateFormat parameter.

```
DTUnixTimeToDateTimeString PROTO UnixTime:QWORD, lpszDateTimeString:QWORD,␣
↪DateFormat:QWORD
```

**Parameters**

- UnixTime - QWORD containing a unix time integer value to convert to a date & time string.
- lpszDateTimeString - Pointer to a buffer to store the date & time string. The format of the date & time string is determined by the DateFormat parameter.
- DateFormat - Value indicating the date & time format to return in the buffer pointed to by lpszDateTimeString parameter. The parameter can contain one of the following constants as listed in the *DateTime Formats* page and as defined in the DateTime.inc include file.

**Returns**

There is no return value, the date & time string will contain the date & time as specified by the DateFormat specified.

**Notes**

Unix time is defined as the number of seconds elapsed since 00:00 Universal time on January 1, 1970 in the Gregorian calendar (Julian day 2440587.5)

The UNIXTIMESTAMP format is a **string** representation of the unix time in integer format if used as the DateFormat value.

**Example**

```
.data
DateTimeStringValue db DATETIME_STRING dup (0)

.code
Invoke DTUnixTimeToDateTimeString, 1276278420, Addr szDateTimeString, DDMMCCYYHHMMSSMS
; DateTimeString should now contain the string "11/06/2010 17:47:00:00"
```

**Example**

```
.data
DateTimeStringValue db DATETIME_STRING dup (0)

.code
Invoke DTUnixTimeToDateTimeString, 1656241202, Addr szDateTimeString, UNIXTIMESTAMP
; DateTimeString should now contain the string "1656241202"
```

**See Also**

*DTDateTimeStringToUnixTime*, *DTQwordDateTimeToUnixTime*, *DateTime Formats*

## 3.23 DTUnixTimeToQwordDateTime

Converts a QWORD containing a unix time integer value to QWORD values containing date and time information.

```
DTUnixTimeToQwordDateTime PROTO UnixTime:QWORD
```

**Parameters**

- UnixTime - QWORD containing a unix time integer value to convert to QWORD date & time values.

**Returns**

On return RAX will contain the **date** information in the following format:

RAX Register Bits:

```
+-----------------------------------------------+-----------------------+--
↪---------+----------+
| DWORD                                         | WORD                  |␣
↪BYTE       | BYTE      |
+-----------------------------------------------+-----------------------+--
↪---------+----------+
| Bits 63-32                                    | Bits 31-16            |␣
↪Bits 15-8  | Bits 7-0  |
+-----------------------------------------------+-----------------------+--
↪---------+----------+
| Not used - Not applicable                     | Century Year          |␣
↪Month      | Day       |
+-----------------------------------------------+-----------------------+--
↪---------+----------+
| N/A                                           | CCCCYY                |␣
↪MM         | DD        |
+-----------------------------------------------+-----------------------+--
↪---------+----------+
```

On return RDX will contain the **time** information in the following format:

RDX Register Bits:

```
+-----------------------------------------------+----------+----------+-
↪---------+----------+
| DWORD                                         | BYTE     | BYTE     |␣
↪BYTE      | BYTE      |
+-----------------------------------------------+----------+----------+-
↪---------+----------+
| Bits 63-32                                    | Bits 31-23 | Bits 23-16 |␣
↪Bits 15-8 | Bits 7-0  |
+-----------------------------------------------+----------+----------+-
↪---------+----------+
| Not used - Not applicable                     | Hour     | Minute   |␣
↪Second    | Millisec  |
+-----------------------------------------------+----------+----------+-
↪---------+----------+
| N/A                                           | HH       | MM       |␣
↪SS        | MS        |
+-----------------------------------------------+----------+----------+-
↪---------+----------+
```

(continues on next page)

(continued from previous page)

**Notes**

Unix time is defined as the number of seconds elapsed since 00:00 Universal time on January 1, 1970 in the Gregorian calendar (Julian day 2440587.5)

**Example**

```
.data?
qwDate dq ?
qwTime dq ?
Year dw ?
Month db ?
Day db ?
Hours db ?
Minutes db ?
Seconds db ?
Millisec db ?

.code
Invoke DTUnixTimeToQwordDateTime, 1276278420
mov qwDate, rax ; rax contains the date information
mov qwTime, rdx ; rdx contains the time information

mov rax, qwDate
shr rax, 16 ; move year into upper word of rax
mov Year,  ax ; year
mov rax, qwDate
mov Month, ah ; month
mov Day, al ; day

mov rax, qwTime
shr eax, 16 ; move hour and minute into upper word of eax
mov Hours, ah ; hours
mov Minutes, al ; minutes
mov rax, qwTime
mov Seconds, ah ; seconds
mov Millisec, al ; millseconds
```

**See Also**

*DTQwordDateTimeToUnixTime*, *DTDateTimeStringToUnixTime*, *DTUnixTimeToDateTimeString*

## 3.24 DTYear

Get the year part of a date & time string.

```
DTYear PROTO lpszDateTimeString:QWORD, DateFormat:QWORD
```

**Parameters**

- `lpszDateTimeString` - Pointer to a buffer containing the date & time string to retrieve the **year** portion from. The format of the date & time string is determined by the `DateFormat` parameter.

- DateFormat - Value indicating the date & time format used in the buffer pointed to by lpszDateTimeString parameter. The paramater can contain one of the following constants as listed in the *DateTime Formats* page and as defined in the DateTime.inc include file.

**Returns**

Returns the year value in RAX

**Example**

```
.data
DateTimeStringValue db "2008/03/21 16:21:01:00",0

.data?
YearValue dq ?

.code
Invoke DTYear, Addr DateTimeStringValue
mov YearValue Value, rax ; save year part of date value to data variable
; eax should contain 2008
```

**See Also**

*DTDay*, *DTMonth*, *DTIsLeapYear*, *DTDayOfWeek*, *DateTime Formats*

| Function | Description |
|---|---|
| *DTGetDateTime* | Get the current date & time and return it as a formatted string |
| *DTDateFormat* | Converts a SYSTEMTIME structure to a formatted date & time string |
| | |
| *DTQwordDateTimeToDateTimeString* | Converts QWORD values containing date & time information to a formatted date & time string |
| *DTDateTimeStringToQwordDateTime* | Converts a formatted date & time string to QWORD values containing date and time informatio |
| | |
| *DTQwordDateToJulianDate* | Converts a QWORD value containing date information to a Julian date integer |
| *DTJulianDateToQwordDate* | Converts a Julian Date number to a QWORD value containing date information |
| | |
| *DTDateTimeStringToJulianMillisec* | Converts a formatted date & time string to a Julian date integer in RAX and milliseconds in RI |
| *DTJulianMillisecToDateTimeString* | Converts a Julian date integer and millisec time to a formatted date & time string |
| | |
| *DTQwordDateTimeToJulianMillisec* | Converts QWORD values containing date & time information to Julian date and total millisecor |
| *DTJulianMillisecToQwordDateTime* | Converts Julian date integer value and total milliseconds to QWORD values containing date and |
| | |
| *DTDateTimeStringToUnixTime* | Converts a formatted date & time string to a QWORD value containing a unix time integer valu |
| *DTUnixTimeToDateTimeString* | Converts a QWORD containing a unix time integer value to a formatted date & time string |
| | |
| *DTQwordDateTimeToUnixTime* | Converts QWORD values containing date & time information to a QWORD value containing a un |
| *DTUnixTimeToQwordDateTime* | Converts a QWORD containing a unix time integer value to QWORD values containing date and t |
| | |
| *DTQwordTimeToMillisec* | Converts a QWORD value containing time information to total amount of milliseconds |
| *DTMillisecToQwordTime* | Converts total milliseconds to a QWORD value containing time information in hours, minutes, |
| | |
| *DTDayOfWeek* | Returns an integer indicating the day of the week from a date & time string |
| *DTIsLeapYear* | Calculate if a specified year is a leap year |
| *DTDay* | Get the day part of a date & time string |
| *DTMonth* | Get the month part of a date & time string |

Table 1 – continued from previous page

| | |
|---|---|
| *DTYear* | Get the year part of a date & time string |
| | |
| *DTDateStringsCompare* | Compare two date & time strings to determine if they are: equal in value, one is less than and |
| *DTDateStringsDifference* | Returns the difference in days of the first date & time string to the second date & time string |
| *DTDateTimeStringsDifference* | Returns the difference in days or milliseconds of the first date & time string to the second dat |

CHAPTER

FOUR

# DATETIME FORMATS

Some of the functions used in the library allow for a specified date & time format as either required by a function or in some cases as desired as an output.

The following are constants defined in the `DateTime.inc` include file.

| DateTime Constant | Example of DateTime String |
|---|---|
| CCYYMMDDHHMMSSMS | 1974/03/27 14:53:01:00 |
| CCYYMMDDHHMMSS | 1974/03/27 14:53:01 |
| CCYYMMDDHHMM | 1974/03/27 14:53 |
| CCYYMMDD | 1974/03/27 |
| CCYYMM | 1974/03 |
| YEAR | 1974 |
| | |
| DDMMCCYYHHMMSSMS | 27/03/1974 14:53:01:00 |
| DDMMCCYYHHMMSS | 27/03/1974 14:53:01 |
| DDMMCCYYHHMM | 27/03/1974 14:53 |
| DDMMCCYY | 27/03/1974 |
| DDMM | 27/03 |
| DAY | 27 |
| | |
| MMDDCCYYHHMMSSMS | 03/27/1974 14:53:01:00 |
| MMDDCCYYHHMMSS | 03/27/1974 14:53:01 |
| MMDDCCYYHHMM | 03/27/1974 14:53 |
| MMDDCCYY | 03/27/1974 |
| MMDD | 03/27 |
| MONTH | 03 |
| | |
| YYMMDDHHMMSSMS | 74/03/27 14:53:01:00 |
| YYMMDDHHMMSS | 74/03/27 14:53:01 |
| YYMMDDHHMM | 74/03/27 14:53 |
| YYMMDD | 74/03/27 |
| YYMM | 74/03 |
| YY | 74 |
| | |
| MMDDYY | 03/27/74 |
| DDMMYY | 27/03/74 |
| DAYOFWEEK | Monday |
| | |
| HHMMSSMS | 14:53:01 |

continues on next page

Table 1 – continued from previous page

| | |
|---|---|
| HHMMSS | 14:53:01 |
| HHMM | 14:53 |
| HH | 14 |
| | |
| UNIXTIMESTAMP | 1276278420 |
| | |
| TODAY | same as DDMMCCYYHHMMSS |
| NOW | same as DDMMCCYYHHMMSS |
| TIME | same as HHMM |
| | |
| AMERICAN | same as MMDDYY |
| BRITISH | same as DDMMYY |
| FRENCH | same as DDMMYY |
| JAPAN | same as YYMMDD |
| TAIWAN | same as YYMMDD |
| MDY | same as MMDDYY |
| DMY | same as DDMMYY |
| YMD | same as YYMMDD |

**Note:** Constants: CC=Century, YY=Year, MM=Month, DD=Day, HH=Hours, MM=Minutes, SS=Seconds, MS=Milliseconds, DOW=Day Of Week

**Note:** The **UNIXTIMESTAMP** format is a **string** representation of the unix time in integer format and used in the *DTDateTimeStringToUnixTime* and *DTUnixTimeToDateTimeString* functions.

CHAPTER

FIVE

# INDICES AND TABLES

- genindex
- modindex
- search