



Represent The Date Value In 2 Bytes

Mohammed Thakir Shaamood

English Department, Al-iraqia University, Baghdad, Iraq

m4alani9@gmail.com

Abstract

Dates are encoded and stored in various ways in different libraries and operating systems by different programming languages; The operating system may use different ways to encode and store a date depending on the use-case. including the calendar system. So most systems need a certain amount of date and then stop. In this research, we will mention a history of the date and how it is represented in languages and applications such as SQL, Python, and SAS programs. We merge the bits of the two so we have 16 bits of which we book 5 bits to represent the day, then four to represent the month, and the rest to represent the year. But that way we have 128 years. There is a way to increase this number if we set a starting date and add the number of years required. We will explain The way and give examples of representing the highest date that we can get.

Keywords: Date, Time, Byte.

تمثيل قيمة التاريخ في 2 بايت

محمد ذاكر شعموط

قسم اللغة الانجليزية، الجامعة العراقية، بغداد، العراق

m4alani9@gmail.com

الملخص

يتم تشفير التواريخ وتخزينها بطرق مختلفة في مختلف المكتبات ونظم التشغيل حسب لغات البرمجة المختلفة ؛ قد يستخدم نظام التشغيل طرقاً مختلفة للتشفير وتخزين التاريخ اعتماداً على حالة الاستخدام. بما في ذلك النظام التقويمي. لذلك تحتاج معظم الأنظمة إلى قدر معين من التاريخ ثم تتوقف. في هذا البحث، سنذكر التاريخ وكيف يتم تمثيله في لغات وتطبيقات مثل SQL ، Python ، SAS . نقوم بدمج أجزاء البايتهين بحيث يكون لدينا 16 بت نحجز 5 بت منها لتمثيل اليوم، ثم أربعة لتمثيل الشهر، والباقي لتمثيل العام. لكن بهذه الطريقة لدينا 128 عامًا. هناك طريقة لزيادة هذا الرقم إذا حددنا تاريخ البدء وأضفنا عدد السنوات المطلوبة. سنشرح الطريق ونقدم أمثلة على تمثيل أعلى تاريخ يمكننا الحصول عليه.

الكلمات المفتاحية: التاريخ، الوقت، بايت.

Introduction

We seldom consider in detail since it's a really fundamental concept. Edge cases must be considered while developing software or evaluating data generated by software. In computer science and programming, a system date is a notion that defines a computer system's perception of the passage of time. Date also refers to



the passage of time as defined by the passage of days on the calendar in this sense. A machine clock computes the current date on the system, which is often implemented as a simple count of the number of ticks that have occurred from an arbitrary beginning date known as the epoch. Unix and POSIX-compliant systems, for example, encode system date ("Unix date") as the number of seconds elapsed from the start of the Unix epoch on 1 January 1970 00:00:00 UT, with leap seconds excluded [1]. System dates are stored in two forms in systems that implement the 32-bit and 64-bit versions of the Windows API, such as Windows 9x and Windows NT: SYSTEMDATE, which is a year/month/day value, and FILEDATE, which is a count of the number of 100 millisecond ticks. Calendar dates may be converted to device dates, which are more comprehensible to the majority of people. The Unix system time 1000000000 seconds from the beginning of the epoch, for example, corresponds to the calendar date 9 September 2001 01:46:40 UT [2].

The library subscribers that manage these changes can also handle time zone adjustments, daylight saving time (DST), snapshot seconds and the user's local preferences. To use library routines that convert calendar dates to device dates, many libraries also provide procedures that convert between the two. There are a variety of ways that date values may be stored, including text (e.g., "March 1, 2006") or numerically (e.g., the number of days since January 1, 1970). When a date is recorded as a number, some measurements, such as the number of days between two dates, are reduced to triviality. Without knowing the reference date, the stored value has no significance. While not having to know a reference date solves the issue of storing dates, this comes with many additional difficulties. When dates are stored as text, the format differs depending on the nation. In English-speaking nations, the second month of the year is known as February, while elsewhere it is referred to by another name. This date type should be handled with care, since it may cause problems if it appears in a form like this: 01/03/06 [4]. When used in certain countries, the third of January 2006 is the first of March 2006; while, in other nations, the first of March 2006 is the third of January 2006. There are many methods for resolving these problems, including the adoption of ISO 8601, an international standard for dates.

This is an official guideline for when it is acceptable to use a four-digit year, two-digit month, and two-digit day. There should be dashes between each number. For example, 2021/11/30 is written as the first day of March in 2006. The time periods used for dates, such as day, week, month, year, quarter, etc., are different from time periods, which describe a particular day as well as the hours, seconds, and fractions of a second on that day. Difficulties with dates arise because they change depending on location. For example, mid-day on April 12-2020 may occur at noon in some places, and in the afternoon in others. The issue is exacerbated by daylight



savings time. The ISO standard 8601 defines how time information, including time zone, should be applied for a date. Dates are often recorded as numbers, for example the number of seconds since the start of 1970, as for basic dates. The period of the month is from 1 to 12. The daily component is 1 - x, with x being 28, 29, 30, or 31, and depending on the month, It is between 0001 and 9999[5] each year. The internal representation of the day is a 4 byte string, each byte is made up of 2 decimal numbers, the first 2 bytes is represented by year, the third bytes is represented by month and the final byte is represented by day[6]. As stated by SQLDA, the length of a DATE column is 10 bytes, which is the proper time to represent a character string for the term or value and the date in each language or application is different. Most systems need some date and then cease. The date in various languages and applications, for example:

- Python

This table illustrates how Python maintains the dates of its different components (year, month, and day). When it comes to the range of seconds and the year, this is very similar to C's implementation. It is possible that leap seconds may have a duration of 60 seconds. C begins the year 1990.

The Name	Bytes(8 bits)	The Values
Year	2.0	1-9999
The Month	1.0	1-12
The Day	1.0	1-31

Table 1: Structure of Python Dates

- SAS program

Using SAS, dates are represented as numeric constants. Let's examine the way dates are both displayed within and outside the company. Because they alter the way values are presented in production, formats cannot be used to specify a certain date as a constant in a program. We may choose to utilize Informat, since they will always translate whatever is provided to them, but we have to use the INPUT() method, which transforms the value we give it to an Informat. After the final quotation, the text that follows is the most important element of a date constant. D may be written in uppercase or lowercase, which represents the date. We would be creating a character variable rather than a numeric variable if we placed a literal date in quotations without the letter at the end [7]. As we use the newly generated variable in an exercise that includes a computation, the change may not be immediately obvious. our "D", "T" Customization options pertaining to dates In SAS, you have a number of system settings. These choices govern how the SAS job or session operates. These four important date options are YEARCUTOFF,



DATESTYLE, DATE/NODATE, and DTRESET. year-end deadline On New Year's Eve in 1999, many were holding their breath. assumed to be 1900 or 2000 (19). To understate or obfuscate: A consequence of this was the Y2K problem, in which we could not account for how the years were stored (19). When two-digit years occur in the 1900s, SAS supports two-digit years. We are able to provide 100 year timeframes to two-digit years by using YEARCUTOFF. Two-digit years may relate to all dates that we supply SAS with. This comprises constants with a fixed date (the value may be anything between the specified dates), constant dates received from raw data (using the INPUT statement), and dates generated from character strings (using the INPUT() function). Once a date value is generated, YEARCUTOFF has no impact on it. The default system environment is 1920. For example, every two-digit year between 20 and 99 will be interpreted as the years from 1920 to 1999, whereas the other digits (00 to 19) would be seen as the years from 2000 to 2019. Any args you provide are sent as OPTIONS. The year of year cut-off is specified as 'yyyy' and 'yyyyyy'. We'll demonstrate this concept using a sequence of OPTIONS statements and date constants. For example, in Figure 1, we demonstrate how YEARCUTOFF, a two-digit year value, is calculated by utilizing date constants.

Unformatted Constants		
date	time	datetime
16287	26100	-1748226540

Formatted Constants		
date	time	datetime
August 4, 2004	7:15 AM	07AUG1904:21:31:00

Figure 1:Date, time, datetime constants of SAS program

- Each leap year day, including the year leap year day in the year 2000, must be recorded in the SAS data values.
- Once the calendar has been revised several days later, after September 1752, SAS date values will correctly inform us on which day of the week a certain day occurs. In the future SAS estimates of day-of-the-week and length-of-date until A.D. 19,900[10] will be exact.
- SAS data values may be handled by several SAS language elements: function, format and informat[11].

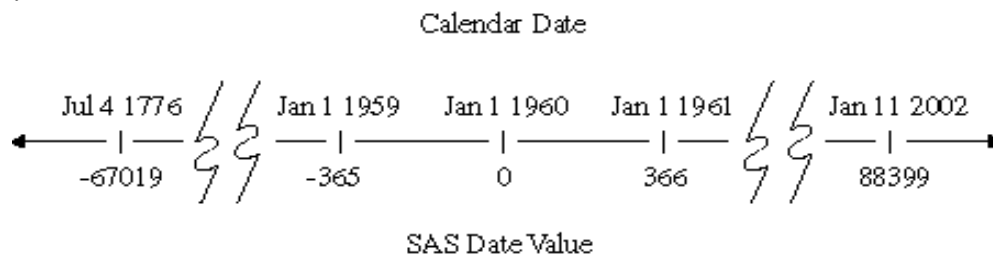
SAS date value

Is the amount of seconds between January 1, 1960 and some hour, minute or second.

Some dates appear in a calendar and in the following graphic as SAS-date values.



Calendar dates How SAS Converts Calendar Dates [12].



SQL

MySQL has a variety of forms, including YEAR, MONTH and DAY, for various kinds of date data. One second is MySQL's lowest date granularity. MariaDB has a temporal shape with a granularity of microseconds. The time may be calculated at the microsecond level. There is no doubt, which is the best option, most temporary kinds have no alternatives. What do we do when both dates are stored [13]? The only issue is what to do.

For this reason, DATE and TIMESTAMP are two extremely comparable MySQL data types. Both function with specific applications, although one works better than the other in certain circumstances. Have a look at this:

DATE

With a single-second precision, from the year 1001 to 9999, this form has a vast variety of values. It saves the date packed in YYYYMMDD format, regardless of the time zone, This utilizes a storage area of eight bytes, Mysql shows DATE values, as a standard method of representing dates, in an unequivocally sortable style, such as 2008-01-16[14].

TIMESTAMP

Like a Unix TIMESTAMP, the TIMESTAMP form keeps up to a minimum of seconds that have passed on January 1, 1970 since midnight (GMT). As just 4 bytes of data is collected by TIMESTAMP, its range is much smaller than DATE: 1970 to mid-2038. MySQL offers UNIXTIME() and UNIX_TIMESTAMP() methods to convert Unix TIMESTAMP into a date. The DATE values in MySQL 4.1 and newer are formatted similarly to the TIMESTAMP values but are displayed in MySQL 4.0 or older versions without any punctuation. In all MySQL versions the TIMESTAMP format for storage is the same; it is only a variation in display formatting. The time zone impacts the significance of a TIMESTAMP. On the MySQL Server, OS, and client connections, time zones are configured. Thus in Eastern Standard Time (EST) 5 hours ahead of GMT, the TIMESTAMP is 1969-13-31 19:00:00 with the value 0. It is important to note this distinction: If we store or access data in several time zones, the actions of TIMESTAMP and DATE would be very different. In relation to the time zone the former retains values and the latter conserves the textual representation of the date. TIMESTAMP has unique



features lacking in DATE. By default MySQL sets it to the current date when we insert a line without entering a value for the first column of TIMESTAMP.

If a value in the UPDATE statement is not specifically assigned, MySQL updates the value of the first column in the timeline when updating the row. Insert and update behaviors can be personalized in any TIMESTAMP column. In the end, TIMESTAMP columns aren't default, unlike any other data type. Aside from the odd behaviors, TIMESTAMP is more space economical than DATE and may thus be used everywhere we can. Unix TIMESTAMPS are typically stored as integers, although we rarely do it. We don't propose the usage of integer formats since working with them is usually uncomfortable. We may use the BIGINT data type and record the value in microseconds as a TIMESTAMP, or use a DOUBLE for the second fraction after decimal. To get the required data type, we may utilize our storage format: Therefore the BIGINT data type is used to store the value as a TIMESTAMP in microseconds, or to store the second portion after the DOUBLE. Both methods are going to succeed. Instead of MySQL, MariaDB may instead be utilized. The following store classes exist for each value saved in a SQLite (or modified by the database) database:

- **NULL.** This is a value NULL.
- **INTEGER.** It is a signed value [15]. The integer is stored in 1, 2, 3, 4, 6, or 8 bytes, according to the size of the value.
- **REAL:** The value is a floating IEEE point number of 8 bytes representing a floating point value.
- **TEXT:** This is a text string, encoded using UTF-8, UTF-16BE or UTF-16LE database encoding.
- **BLOB:** The value is a blob of information stored as it was inserted in the same format.

There is no dates-specific storage class in SQLite. On the other hand, integrated dates of SQLite may be saved as TEXT, True, or INTEGER:

- The "YYYYYY-MM-DD" text is as ISO8601.
- On November 24, 4714 B.C., according to the proleptic Gregorian calendar, the number of days after noon is actual.
- This INTEGER contains the number of seconds after 1970-01-01 00:00:00 UTC.

An application may utilize the built-in date methods to transform dates stored in different formats.

A BRIEF HISTORIC OVERVIEW

In Analytica, a date value is used for the representation of a date, or the number of days following the date of the date origin. The default date origin of Excel on the Macintosh is January 1, 1904. On most other Windows applications, go to



'Preferences' and set the date origin to January 1, 1900." Establish the start date by using Excel's date definition [16]. The option 'Use Excel date root' causes dates in Analytica and Excel for Windows to have the same numeric value regardless of when the date falls. Because of this, dates before February 29, 1900 will not have the same numeric index in Analytica as they do in Excel. The integer part of the date value is used to indicate the number of days following the date origin. Time as a fraction of a 24-hour day from midnight on begins at the fractional component of the fraction. When utilizing templates with dates or date functions from Analytica versions or before, do not use the Use Excel Date Origin checkbox. When linking data from Excel for Windows or other Windows programs to or from Analytica, we should first double-check this option. Analytica is capable of handling dates that extend from the 1st century AD to the 9999th century. For dates older than the date root, negative integers are utilized. The Gregorian calendar is used for dates; it is a lunar calendar. Except for those that are divisible by 100, every four-year period is a leap year, with the exception of those that are divisible by 400, which are not. Adding an integer n to the date one day forward will provide the date plus n days in the future. The C date functions are a set of functions included in the standard library of the C programming language that perform date manipulation techniques. They assist with time and date accumulation, as well as the ability to interpret date formats and return values to strings. The Y2038 issue was discovered on AOLserver in May of 2006. The software was created using a hack in order to handle the database request that "should never" time out. The project's initial design decided to use an arbitrary time limit beyond which everything was assumed to be OK. By default, the server sets the request to timeout after one billion seconds. Roughly 32 years (32 years, 8 months, and 15 days) have elapsed from the day at which the cutoff date of 2038 will have passed (i.e., approximately one billion seconds (1,000,000,000 seconds) at 12:27:28 UTC on May 13, 2006). The solution, which corrects the overflowing time out calculation, is that the results are reverted to the date before them, causing the application to crash. Once the issue was identified, AOL Server operators had to make changes to the configuration file, lowering the time out. As a result, if you attempt to bypass the waiting period by changing the date on your computer, you will not be able to change the time in 32-bit Unix format, as it is stored in a non-volatile storage device such as a hard drive. Embedded systems that utilize dates for calculation or diagnostic logging are most likely to be impacted by the 2038 problem. A significant number of embedded devices are used in transportation systems of all types, including aircraft, automobiles, and trains. Some of these technologies (e.g., ABS, ESC/ESP, traction control, and four-wheel drive) may be used on both land and air vehicles. As such, these systems are mostly unaffected by the Y2038 issue. Just those who have systems that monitor solely the difference between dates can



profit from systems that only use dates as a means of measuring time. Because the measurement is so simple, there wouldn't be any significant problems.

Government-mandated standards like CARB require this for vehicle diagnostics (California Air Resources Board). As more and more computing devices become built on the Linux OS, embedded systems are proving to be particularly useful in communication devices such as mobile phones and Internet appliances (e.g. routers, wireless access points, and so on).

Consider the example of the Y2038 issue, which causes certain 32-bit Android devices to crash and fail to resume on changing the clock to that date. Even though the new system technology of computers sometimes involves an 18–24 month generational upgrade, embedded systems are built to endure the lifetime of the machine on which they are installed [18].

While some of these technologies may still be functioning in 2038, it is a possibility. The 32-bit restriction of these systems can't be solved without replacing the software, requiring a complete overhaul if the systems are to be upgraded. MySQL database built-in operations such UNIX TIMESTAMP() will return 0 after 03:14:07 UTC on January 19, 2038. Prior to Mac OS X v10.0, the year 2038 problem affected early Mac OS X models. Since 2000, the internal clock of Deep Impact, which measures time in 100-millisecond intervals, has progressed 232 100-millisecond intervals (one-tenth of a second). Faith, economics, technological advances, and navigation have all played a role in defining time and dates throughout history. It was proposed in 46 BC with the Julian calendar. In leap years, every fourth year is a leap year. A Julian year is often used as the foundation for a light year. In order to tie the months to the seasons, the Julian Calendar linked the twelve months with the seasons.

At the International Meridian Conference, Sir Sandford Fleming proposed a global standard time to which 24 time zones of 15 degrees of longitude (360 degrees) are added as local offsets. This way, the local time at each location is kept approximately half an hour from the global standard time and makes the system simpler.

In other words, the conference recognized a different version of Universal Time, but refused to acknowledge his zones, since they were said to be a local matter instead of having anything to do with the conference's goals.

CET (sometimes called Central European Time) superseded Berlin time as a universal time in Germany in 1891. CET was chosen by the Imperial German government as the standard measurement for all state purposes in the year 1893.

In 1916, the German Empire introduced daylight saving time and date in order to save energy.

1924: A new method of synchronizing time and date is presented in the form of the Greenwich Time Signal.



A time scale including both the use of atomic clocks located at Caesium's Institute in Switzerland and phase comparisons to VLF radio waves, used as a standard reference, came into being in 1955.

USNO established the A.1 Scale in 1956. basing our atomic clocks on a Caesium-based standard.

1959: DCF77 was the first DCF station to begin regular operation. It was established in 1960 with the Universal Time Coordinated (UTC).

The second SI unit was established as the atomic weight of the Cesium atom. "TAI is computed according to this precise definition; the Temps Atomique International (TAI) is one unit of energy" (BIMP).

UT1 was established as a replacement to Greenwich Mean Time in 1968. (GMT). The calculation is based on observed features of the solar system, such as the mean solar time. At the same time, the definition of a second was redefined to refer to atomic clocks rather than astronomical observations.

The 1972 leap second was the first time this was done. Beginning in 1986, the IANA time zone database was developed.

The Method of the research

We merge the bits of the two so we have 16 bits of which we book 5 bits to represent the day, then four to represent the month and the rest to represent the year. But that way we have 128 years. There is a way to increase this number if we set a starting date and add the number of years required. We combine bits of months with bits of years and we get 11 bits that represent the highest number 2048 months. When divided by 12 we get 170 years long enough for the systems. Here we add the number of years to the start date of the program and we get the desired date.

The modus operandi for extracting months and years is:

X

0	1	2	3	4	5	6	7	8	9	10	11
0	1	2	3	4	5	6	7	8	9	10	11
12	13	14	15	16	17	18	19	20	21	22	23
24	25	26	27	28	29	30	31	32	33	34	35
36	37	...									
										...	2047

Table 2: Representation (11 bits) in decimal format

Most systems need a certain amount of date and then stop.

- ind = number stored in 11 bits within 2 bytes.



- X = the number of months represented by the columns.
- Y = number of years represented in rows.
- W = the number of months in a year, which is 12, which is considered constant and does not change Laws to extract the months and the year.
- $X = \text{ind} \bmod W$
- $Y = \text{ind} \setminus w$

The law of converting the month and year into a number for purposes of storage

- $\text{Ind} = Y * W + x$

When displaying the representation of the date in the languages that I mentioned (VB, .Net, Access, SQL) and noting the Range, the date in it, we notice written in it bearing a date from year to year such and such. At the beginning of the search we described the date in two languages or programs, for example python and the New Access, so then in our work we can store the date in 2 bytes. while the program is running The system lasts 170 years, which is sufficient for most systems if we adopt 80 years in the past and 90 years for the future The focus is only on Date and not on DateTime when running the class Better after a binary representation of history, we convert it into hexadecimal and then store it The 2 bytes used can be called Low and High, Low is used to store numbers of ones and tens, and High to store hundreds and thousands. The class contains several functions, including converting to binary, decoding the binary, converting to hexadecimal and returning it, and other functions that take bits of the day (the first five bits) and a function that takes the rest of the bits to extract the month and year.

- **The Steps of the Method:**

1. The declaration of two variables, the first has an integer and is the starting year value. The second is a date type variable.

Private _StartYear As Integer

Private _Value As Date

2. Do a constructor class to create the starting value of the class ("1900" the first year the system works on) and to infer the first date value of the variable value_.

_Value = New Date(_StartYear, 1, 1)

3. The property is called MaxAvaliable, and it is read-only. It returns the highest date value that the system can cover through 2byte.

Public ReadOnly Property MaxAvailable As Date

Get

Return New Date(_StartYear + 170, 8, 31)

End Get

End Property

4. The property is called MaxAvaliable, and it is read-only. Returns to us the lowest date value that the system can cover through 2byte.



Public ReadOnly Property MinAvailable As Date

Get

Return New Date(_StartYear, 1, 1)

End Get

End Property

5. A property called value is to calculate the current date value that was entered by the user. If we want to query the date only, we will enter the Get and take the value_ value which carries the date value. If we send a specific date with calling this property, it will check it through The condition that exists, and it cannot be less than the launch date, nor more than the date that the program can cover using 2byte. Then the date value is updated and the new date value is assigned to the variable value_.

Public Property Value As Date

Get

Return _Value

End Get

Set(value As Date)

If value > MaxAvailable And value < MinAvailable Then

Throw New NotImplementedException("Error Message")

Else

_Value = value

End If End Set

End Property

6. The (ToBinary) function receives the first variable Num, which is a decimal number, the value of the entire date, and the second variable Digits, which is the number of bits required to represent the number in it. The line found is to convert the received number (Num) into binary, and the way it works is: Convert.ToString (Num, 2) This function converts the decimal number into a binary String, and the number 2 is the base of the conversion, and here in this function is called Base.

Private Function ToBinary(Num As Integer, Digits As Integer) As String

Return Convert.ToString(Num, 2).PadLeft(Digits, "0"c)

End Function

7. The FromBinary function is to convert the binary to decimal: it receives a variable of type String and then runs a loop to read each bit separately and enter it into the existing equation and then add it with the previous value of the previous bit (this equation is very accurate to convert the binary to decimal).

Private Function FromBinary(Bin As String) As Integer

Dim Ret As Integer



```

For i As Integer = 1 To Len(Bin)
    Ret += Bin.Substring(Len(Bin) - i, 1) * (2 ^ (i - 1))
Next
Return Ret
End Function

```

8. The BinaryValue function This function calculates the months and converts them into a binary, where it is represented by 11 bits and converts the value of the day into a binary may also represented by 5 bits and combines the bits of the month 11 with the bits of the day 5 to form 2 bytes, which is the basis of the research idea. At the beginning will be an announcement of the months variable to calculate the number of months, which is the difference between the date given by the user and the launch date: the year minus the starting year multiplied by 12 to convert it into months.

Public Function BinaryValue() As String

```

1-Dim Months As Integer=(_Value.Year - _StartYear)* 12 + _Value.Month -1
2    Dim MthBin As String = ToBinary(Months, 11)
3    Dim DaysBin As String = ToBinary(_Value.Day, 5)
4    Dim Bin As String = MthBin & DaysBin
Return Bin
End Function

```

9. Data Property is the most important part of this class, AsUShort stands for 2Byte storage variants. Get returns the value of the date in its natural form, i.e. it takes a binary value and converts it into a date in the form of decimal numbers and returns it. Set declares a value variable of type as UShort, meaning that it holds a value not exceeding 2bytes. Bin: assigns the String the output value to 16 bits.

Public Property Data As UShort

```

Get
Return FromBinary(BinaryValue)
End Get
Set(value As UShort)
    Dim Bin As String = ToBinary(value, 16)

    Dim DaysBin As String = Bin.Substring(11)
    Dim Days As Integer = FromBinary(DaysBin)

    Dim MthsBin As String = Bin.Substring(0, 11)
    Dim Months As Integer = FromBinary(MthsBin)

    Dim RealMonths As Integer = 1 + Months Mod 12

```



Dim RealYears As Integer = _StartYear + Months \ 12

Dim Ret As Date = **New Date**(RealYears, RealMonths, Days)

If Ret > MaxAvailable And Ret < MinAvailable Then

Throw New NotImplementedException("Error Message")

Else

_Value = Ret

End If

End Set

End Property

End Class

10. When running the application, an object named CD is created from the CompactData class.

Public Class Form1

Dim CD As CompactDate

11. When the application is run directly, Click on the btnSet button to call CompactData for the purpose of building the Constructor and creating the starting value of the date which is (1900) and this value is based on the (TextBox) in the design and its name is numStartYear.

Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load

btnSet_Click(sender, e)

End Sub

12. The btnSet code when pressed to send the program's launch date value, which is in the TextBox, and this is called (numStartYear), which is 1900, and then assigned to the object CD. pkrValue is a TextBox in the program window 1- It has a value based on it 2- We make the MinDate property the same as the starting value. The lowest date can be written in this application 3- The highest value of TextBox is "pkrValue" It is counted from summoning the property "MaxAvailable" in class via the CD object.

Private Sub btnSet_Click(sender As Object, e As EventArgs) Handles btnSet.Click

CD = New CompactDate(numStartYear.Value)

pkrValue.Value = CD.Value

pkrValue.MinDate = CD.Value

pkrValue.MaxDate = CD.MaxAvailable

lblMax.Text = "Max value is:" & Format(CD.MaxAvailable, "yyyy/MM/dd")

End Sub

13. When clicking on btnStart, which appears in the form as "Show": pkrValue.Value: CD.Value when typing a specific date in the specified TextBox with its name (pkrValue) By the user and pressed btnStart The CD



object takes the written value and applies it to the class to calculate the date and extract the binary values ... etc. Then TextBox) txtBin) Display the binary values by taking the required text and displaying it in 8-8 bits.

Private Sub btnStart_Click(sender As Object, e As EventArgs) Handles btnStart.Click

```
CD.Value = pkrValue.Value
txtBin.Text = CD.BinaryValue.Substring(0, 8) & "-" &
CD.BinaryValue.Substring(8)
txtDec.Text = CD.Data
txtHex.Text = Hex(CD.Data)
```

End Sub

14. When this button is pressed, the condition is tested to make sure that the existing Text is a number, then it takes the decimal number of 16 bits and calls the Data property in this (CD.Data = txtDec.Text) in order to base it on the value of txtDec. lblDateRead.Text Display the date in the date format.

Private Sub btnRead_Click(sender As Object, e As EventArgs) Handles btnRead.Click

```
If IsNumeric(txtDec.Text) Then
    CD.Data = txtDec.Text
    lblDateRead.Text = "Readed Date is: " & vbCrLf &
    Format(CD.Value, "yyyy/MM/dd")
End If End Sub End Class
```

Results

Illustration by means of images of the result of dates and their representation in the Binary and Hexadecimal language ...

A representation of the starting date is Figure 2

Figure 2: Represent the date 1/1/1900

- The highest date that can be written in this representation Figure 3:



Form1

Start Year
1900 [Set]
Max value is: 2070/08/31

Date Value
2070/08/31 [Show]

Hexa Value
FFFF

Binary Value
11111111-11111111

Data As U-Short
65535

Readed Date is: 2070/08/31 [Read]

Figure 3: A representation of the highest date that can be written in this paper
8/31/2070

Other examples for representing date are Figures 4, 5, and 6

Form1

Start Year
1900 [Set]
Max value is: 2070/08/31

Date Value
1975/12/31 [Show]

Hexa Value
71FF

Binary Value
01110001-11111111

Data As U-Short
29183

Readed Date is: 1975/12/31 [Read]

Figure 4: Represent the date 31/12/1975

Form1

Start Year
1900 [Set]
Max value is: 2070/08/31

Date Value
2000/01/01 [Show]

Hexa Value
9601

Binary Value
10010110-00000001

Data As U-Short
38401

Readed Date is: 2000/01/01 [Read]

Figure 5: Represent the date 1/1/2000



- [1] "System date – Names and nicknames for System date – NicknameDB," Nicknamedb.com. [Online]. Available: <https://nicknamedb.com/system%20date/top-rated>. [Accessed: 01-Apr-2021].
- [2] Wikipedia contributors, "System time," Wikipedia, The Free Encyclopedia, 03-Mar-2021. [Online]. Available: https://en.wikipedia.org/w/index.php?title=System_time&oldid=1009961656. [Accessed: 01-Apr-2021].
- [3] Auckland.ac.nz.[Online].Available: <https://www.stat.auckland.ac.nz/~paul/ItDT/HTML/node39.html>. [Accessed: 01-Apr-2021].
- [4] Wu.ac.at. [Online]. Available: <http://statmath.wu.ac.at/courses/data-analysis/itdtHTML/node55.html>. [Accessed: 01-Apr-2021].
- [5] G. Blokdyk, IBM docs: Complete self-assessment guide. North Charleston, SC: Createspace Independent Publishing Platform, 2018.
- [6] "ColumnStore Data Types," Mariadb.com. [Online]. Available: <https://mariadb.com/kb/en/columnstore-data-types/>. [Accessed: 02-Apr-2021].
- [7] "Informats and Formats - SAS for Beginners," Webgarden.com. [Online]. Available: <https://sasforum.webgarden.com/sas/sas-programmes/base-sas/informats-and-formats>. [Accessed: 02-Apr-2021].
- [8] D. P. Morgan, The Essential Guide to SAS dates and times, Second Edition. SAS Institute, 2014.
- [9] Martinsinnovativeproductsoptin.com. [Online]. Available: <http://martinsinnovativeproductsoptin.com/littlevgl-esp32-tjsiq/binary-date-converter.html>. [Accessed: 03-Apr-2021].



- [10] Sas.com. [Online]. Available: <https://v8doc.sas.com/sashtml/lrcon/zenid-63.htm>. [Accessed: 03-Apr-2021].
- [11] "About SAS Date, Time, and Datetime Values," Sas.com, 31-Jan-2017. [Online]. Available: <https://support.sas.com/documentation/cdl/en/lrcon/65287/HTML/default/p1wj0wt2ebe2a0n1lv4lem9hdc0v.htm>. [Accessed: 03-Apr-2021].
- [12] "About SAS Date, Time, and Datetime Values," Sas.com, 11-Feb-2010. [Online]. Available: <https://support.sas.com/documentation/cdl/en/lrcon/62955/HTML/default/a002200738.htm>. [Accessed: 03-Apr-2021].
- [13] H. Schwartz, P. Zaitsev, V. Tkachenko, and B. Schwartz, High performance MySQL: Optimization, backups, and replication, 3rd ed. O'Reilly Media, 2012.
- [14] B. Schwartz, High performance MySQL: Optimization, backups, replication, and more, 3rd ed. Sebastopol, CA: O'Reilly Media, 2012.
- [15] "Search SQLite Documentation," Sqlite.org. [Online]. Available: <https://www.sqlite.org/search?q=stored>. [Accessed: 03-Apr-2021].
- [16] "Date Functions," Analytica.com. [Online]. Available: https://wiki.analytica.com/Date_Functions. [Accessed: 04-Apr-2021].
- [17] Wikipedia contributors, "C date and time functions," Wikipedia, The Free Encyclopedia, 07-Mar-2021. [Online]. Available: https://en.wikipedia.org/w/index.php?title=C_date_and_time_functions&oldid=1010858019. [Accessed: 04-Apr-2021].
- [18] Wikipedia contributors, "Year 2038 problem," Wikipedia, The Free Encyclopedia, 29-Mar-2021. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Year_2038_problem&oldid=1014799163. [Accessed: 05-Apr-2021].
- [19] M. T. E-Mail:, "What every developer should know about time," Zenodo.org. [Online]. Available: <https://zenodo.org/record/1443533/files/2018-10-06-what-developers-should-know-about-time.pdf?download=1>. [Accessed: 05-Apr-2021].
- [20] Wikipedia contributors, "Sandford Fleming," Wikipedia, The Free Encyclopedia, 12-Mar-2021. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Sandford_Fleming&oldid=1011669609. [Accessed: 05-Apr-2021].