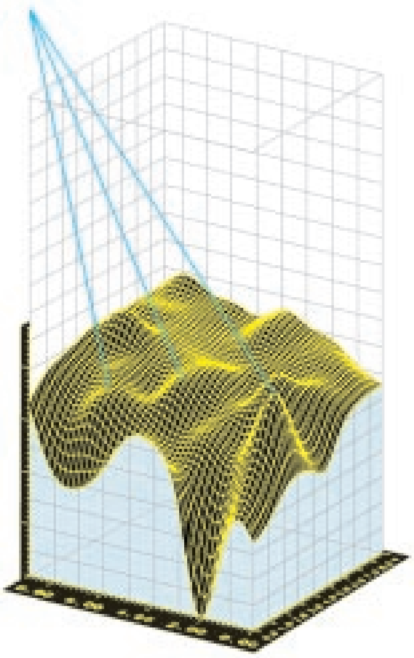

The GPS End-of-Week Rollover

Richard B. Langley

University of New Brunswick

At a few seconds after midnight, Universal Time, on August 22, 1999, the GPS week counter will roll over from 1023 to zero. Although perhaps a little less momentous than the so-called Y2K problem, it has the potential to cause difficulties for some GPS users. In this month's column, we'll examine this event, why it will occur, and the anticipated consequences.

"Innovation" is a regular column featuring discussions about recent advances in GPS technology and its applications as well as the fundamentals of GPS positioning. The column is coordinated by Richard Langley of the Department of Geodesy and Geomatics Engineering at the University of New Brunswick, who appreciates receiving your comments as well as topic suggestions for future columns. To contact him, see the "Columnists" section on page 4 of this issue.

There has been much talk recently in the technical and popular press about the so-called millennium or Y2K bug. At the stroke of midnight initiating the year 2000 (though technically not the start of the next millennium, which actually begins on January 1, 2001), many older computers and software programs may cease to function properly because they will think that the date is January 1, 1900, instead of January 1, 2000. The problem's root cause lies in the fact that some computer manufacturers and programmers had used only two digits to specify the year in dates. So, the computer correctly takes 98 to represent 1998, but 00 means 1900. Furthermore, some computers will consider this year invalid and mistakenly reset their clock to January 4 1980 or some other date.

Governments and the private sector are spending large amounts of money to fix this date error in critical software used for everything from maintaining bank records to running telephone exchanges to controlling aircraft.

While GPS, like all our other systems, is potentially affected by the Y2K problem, including misinterpreting 00 as 1900 and forgetting that the year 2000 is a leap year (the first centurial leap year since 1600), another time-related "bug" occurring next year is one that is peculiar to the Global Positioning System: the end-of-week (EOW) rollover. To understand the nature of this problem, we first need to review how GPS keeps time.

GPS TIME

The signals transmitted by GPS satellites are referenced to GPS (System) Time. Until June 1990, this was the time kept by a single atomic clock at one of the system monitor stations. However, the practice now is to obtain GPS Time from a composite or "paper" clock derived from all operational monitoring station and satellite clocks.

GPS Time is steered over the long run to keep within one microsecond of the global time standard, Coordinated Universal Time (UTC). The definitive UTC time scale is derived *ex post facto* by the Bureau International des Poids et Mesures (BIPM) in Sèvres, just outside of Paris, France. BIPM calculates UTC from a combination of data from about 230 atomic clocks (cesium standards and hydrogen masers) kept by 65 laboratories spread around the globe and reported by some 50 timing centers that maintain a local version of UTC, UTC(k). BIPM carries out clock comparisons chiefly using data from GPS techniques and achieves an ultimate precision of a few nanoseconds. Most timing centers maintain their UTC(k) to within a few microseconds of UTC.

Leap seconds are inserted into the UTC time scale every so often to keep UTC approximately synchronized with the earth's rotational period with respect to the sun, the *length of day*, which varies slightly from day to day and year to year. The yearly average length of day is currently about two milliseconds longer than it was back in 1900, the base year that was used to define the "standard" day containing exactly 86,400 seconds, primarily due to the earth's tidal breaking. A difference of two milliseconds each day, every day, adds up to about 0.7 seconds after one year. The last time a leap second was inserted into the UTC time scale was on June 30, 1997, and the next will be on December 31, 1998.

Time Differences. Unlike UTC, GPS Time has no leap-second jumps. At the integer second level, GPS Time matched UTC in 1980, but because of the leap seconds inserted into UTC, GPS Time is now ahead of UTC by 12 seconds plus a fraction of a microsecond that varies day to day. The difference between UTC as maintained by the U.S. Naval Observatory, UTC(USNO), and GPS Time is included on page 18 of subframe four in the satellite-transmitted navigation message. That page also includes the time of scheduled future leap-second updates. Currently, the GPS Master Control Station is actually maintaining GPS Time to within tens of nanoseconds of UTC(USNO), which, in turn, is kept to within 100 nanoseconds of UTC.

Nevertheless, a Standard Positioning Service (SPS) user can only determine GPS Time, and hence UTC, to the accuracy afforded by selective availability — the dominant error source for standalone (single-receiver) GPS positioning and timing. The present SPS timing accuracy is 340 nanoseconds (at the 95 percent probability level). The corresponding Precise Positioning Service timing accuracy is within 200 nanoseconds.

SPS users can achieve higher time-transfer accuracies by averaging results over an interval of time. By using a common-view mode of GPS time transfer, where two different users observe the same GPS satellite at nearly the same instant of time, considerably higher accuracies can be obtained. By taking differences of the observations made at each site, the users can obtain the difference between the clocks at the two sites even when separated by a large distance.

Z Count. The GPS satellites count and communicate GPS Time in a unique manner that is ultimately related to how they generate the pseudorandom noise (PRN) ranging codes. The P-code is generated by combining two shorter PRN codes, X1 and X2, that are clocked in phase at a chip rate equal to the satellite atomic clock's 10.23-MHz oscillator frequency. Each satellite is assigned a uniquely delayed version of X2 that gives it a corresponding unique P-code. (The product of X1 and an undelayed X2 generates an approximately 38-week-long code. By delaying X2 by a different number of chips, each satellite has a unique, nonoverlapping, one-week segment of the full P-code.) X1 and X2 are reset every Saturday/Sunday midnight, thus resetting the P-code.

X1 has a repetition interval, or period, of 1.5 seconds — a fundamental GPS timing unit. The start of each 1.5-second interval identifies an epoch. The number of X1 epochs since the beginning of the week is called the time of week (TOW) count, which runs from zero to 403,199 at the EOW. The TOW count returns to zero coincident with the resetting of the PRN codes (see Figure 1).

Time of Week. The TOW count can be represented as a 19-bit binary number, a truncated version (the 17 most significant bits) of which is part of the handover word (HOW) that a satellite transmits every six seconds. The HOW appears as the second word in each data subframe of the navigation message. These 17 bits correspond to the TOW count at the X1 epoch that occurs at the start of the immediately following subframe, and so effectively preannounces the arrival of a time marker, just like telephone "speaking clocks" and shortwave radio time and frequency stations. The HOW helps con-

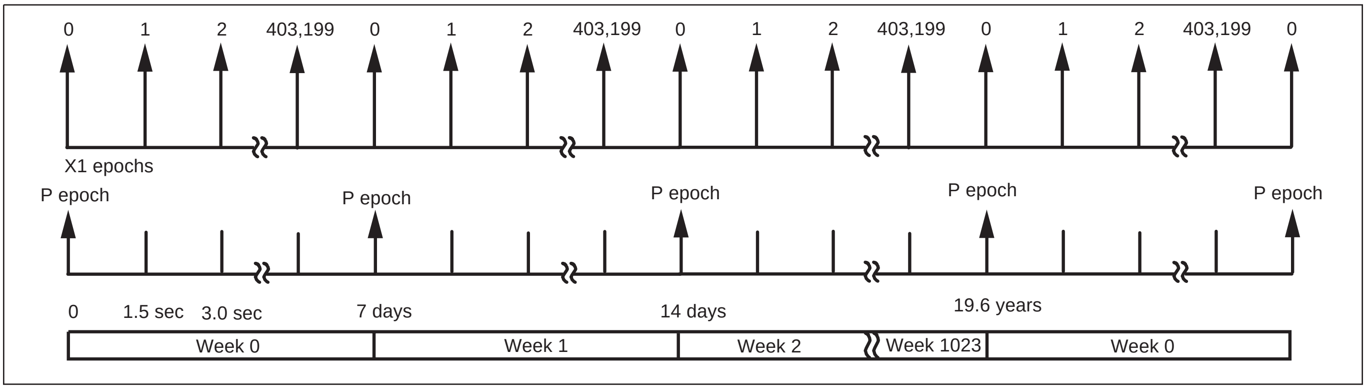

Figure 1. The inherent, fundamental GPS timing unit is the 1.5-second repetition period of the P-code's X1 subcode. The P-code is reset every week or 403,200 X1 epochs. The GPS week number count is reset every 1024 weeks or approximately 19.6 years.

ventional P(Y)-code receivers rapidly acquire the P-code after obtaining the C/A-code and helps resolve the inherent 1-millisecond ambiguity of C/A-code pseudorange measurements.

The TOW count by itself cannot be used to unambiguously establish the date of an event. It can only time an event at modulo 604,800 seconds because it is reset every week (see the “Modular Arithmetic” sidebar). This time ambiguity is reduced by noting the number of full weeks that have elapsed since the GPS Time zero point of midnight UTC beginning January 6, 1980. GPS weeks are numbered consecutively with week zero starting on January 6 and ending on January 12, 1980.

The TOW count and the GPS week number combine to form the 29-bit Z count. The 19 least-significant bits are the TOW count and the 10 most-significant bits are the GPS week number.

The current GPS week is included in subframe one of the navigation message, which — along with other subframes containing satellite clock, ephemeris data, and other user-required information — is transmitted every 30 seconds. Only 10 bits are used to represent the GPS week, and so the largest possible week number is 1023 (2^{10} – 1). In other words, the GPS week number is modulo 1024. At the end of week number 1023, the week number rolls over to zero! Herein lies the cause of a potential problem.

THE ROLLOVER

GPS week number 1023 will begin on August 15, 1999, and end on August 21. As previously described, the week number will then become zero. If a GPS receiver has not been programmed correctly, it could experience *déjà vu* and interpret the start of GPS week zero as being January 6, 1980. The initial release of the GPS Interface Control Document, ICD-GPS-200, identified the GPS week count’s modular nature, which should have been noted by all GPS manufacturers. The ICD’s most recent version specifically notes: “At the expiration of GPS week number 1023, the GPS week number will rollover to zero (0). *Users must account for the previous 1024 weeks*” (italics added). In other

FURTHER READING

For the official GPS signal description, see

- Interface Control Document, *Navstar GPS Space Segment / Navigation User Interfaces*, ICD-GPS-200, Revision C (IRN-200C-002), published on behalf of the Department of Defense by ARINC Research Corp., El Segundo, California, 1997. This document is available as a PDF file from the U.S. Coast Guard’s Web site: <http://www.navcen.uscg.mil/gps/geninfo/gpsdocuments/icd200/icd200c.pdf>.

For details about GPS signal structure and receiver operation, see

- *Global Positioning System: Theory and Applications*, edited by B.W. Parkinson and J.J. Spilker Jr., published by the American Institute of Aeronautics and Astronautics, Inc., Washington, D.C., as Vol. 163 and 164 of *Progress in Astronautics and Aeronautics*, 1996.
- *Understanding GPS: Principles and Applications*, edited by E.D. Kaplan, published by Artech House, Inc., Norwood, Massachusetts, 1996.

For details about the compliance efforts of the GPS Joint Program Office, see

- “GPS Rollover: Compliance Efforts Under Way,” by A. Johnson, in *GPS World*, Vol. 8, No. 9, September 1997, pp. 62–64.
- <http://gps.laafb.af.mil/y2000/index.html>.

Several Web sites contain information about the EOW rollover with links to manufacturer Web sites. For example, see

- <http://www.sustainableworld.com/y2kgps/gpsbug.html>
- <http://www.navcen.uscg.mil/gps/geninfo/y2k/>.

Modular Arithmetic

GPS Time, like all time systems, is based on modular arithmetic. This arithmetic is a little different from conventional arithmetic in that numbers, typically restricted to integers, have a finite maximum value. Adding one to that number doesn’t get you a larger number — it gets you a smaller one, a much smaller one: zero.

Modular arithmetic is known to us all as clock arithmetic. Take the 24-hour time system as an example. If it’s currently 1800, then 8 hours later we say it’s 0200, not 2600. Similarly, if it’s currently 0400, then 6 hours earlier it was not –0200 but 2200. The idea here is that if two numbers differ by 24 or a multiple of 24, then they are “equal.” We could simply write 26 = 2 but this could be confusing. So we write 26 ≡ 2 (mod 24), and –2 ≡ 22 (mod 24), or in words, 26 is congruent (or somehow “equal”) to 2 (modulo 24) and –2 is congruent to 22 (modulo 24). In arithmetic modulo 24, any number larger than 24 is congruent to some number less than 24 because we can always subtract a multiple of 24 from the larger number to get the smaller one. Similarly, any negative number is congruent to some positive number less than 24, and 24 is congruent to 0. This means that in arithmetic modulo 24, we need deal only with integers from 0 to 23.

We can choose any positive integer for the modulus and carry out arithmetic operations accordingly. Using a modulus of 4, for example, we would have 2 + 2 = 0 in our loose notation — a disturbing result if interpreted as conventional arithmetic. But when written 2 + 2 ≡ 0 (mod 4), the meaning is clear.

As another example of modular arithmetic, consider this question: If today is Monday, what day of the week is it 185 days from now? The modulus here of course is 7, the number of days in the week. So, mathematically stated: 1 + 185 ≡ ? (mod 7). The answer: 4 or Thursday. The answer is obtained by dividing the sum on the left side of the congruency by 7, using “long division,” and noting the remainder. Or, alternatively, the sum is divided by 7, and the decimal part of the result is then multiplied by 7.

An interesting quirk of modular arithmetic is that a number and the sum of its digits are congruent, modulo 9. This property is the basis for a formerly well-known procedure (before the days of calculators and computers) for checking the correctness of hand multiplication — the rule for casting out nines, which states that the product of two numbers and the product of the sums of their digits must have the same remainder on division by 9.

Many computer languages have a built-in modular arithmetic function or operator. Typically called MOD, it returns the remainder from an integer division operation. In BASIC, for example, if we enter 5 MOD 2, we get 1 because 5 divided by 2 is 2 with a remainder of 1. The same computation is coded 5 2 MOD in FORTH and MOD (5,2) in FORTRAN.

The following line of FORTRAN code by Henry Fliegel of The Aerospace Corporation inherently uses modular arithmetic by way of integer division to determine the Julian day (JD) number from the year, month, and day of an AD Gregorian calendar date, incorporating all leap year rules:

```
JD=367*Y-7*(Y+(M+9)/12)/4-3*((Y+(M-9)/7)/100+1)/4+
                275*M/9+D+1721029
```

And just how can the GPS EOW rollover be described using modular arithmetic? Very simply: 1023 + 1 ≡ 0 (mod 1024).

words, a GPS receiver must know the current GPS week window or cycle without being told by the navigation message as it contains no year or week cycle information. We are currently in cycle one. Cycle two will last from August 22, 1999, until April 6, 2019. Cycle three will last from April 7, 2019, until November 20, 2038. And so on (see Table 1).

The change next August, however, will not be the first time GPS has experienced a rollover. As we have seen, a rollover occurs in the TOW count every week. Also, the UTC reference week used by a GPS receiver to relate GPS Time to UTC is represented by an eight-bit number and, so consequently, rolls over every 256 weeks. In fact, the Department of Defense issued a Notice Advisory to Navstar Users (NANU) on September 23, 1994, alerting users and manufacturers that some GPS receivers may not be accounting for this particular type of rollover.

Table 1. Start and end dates of the first three GPS week cycles

GPS week cycle	Start of week 0	End of week 1023
1	January 6, 1980. (44244)	August 21, 1999 (51411)
2	August 22, 1999 (51412)	April 6, 2019 (58579)
3	April 7, 2019 (58580)	November 20, 2038 (65747)

Note: The numbers in parentheses are the corresponding modified Julian dates, a running count of elapsed days since midnight beginning November 17, 1858 (Julian date 2400000.5).

RECEIVER EFFECTS

Some GPS receiver manufacturers were well aware of the EOW rollover and programmed their receivers accordingly long before the recent ringing of alarm bells. Others, however, awoke to those bells and only thereafter scurried to revise the firmware in their receivers.

So how does, or how should, a GPS receiver handle the EOW rollover so that it will continue to operate normally through and after the event? A GPS unit needs to know the current GPS cycle to figure out the current date. It can determine this if, for example, it has a correctly set real-time clock (that is hopefully Y2K compliant) backed up by a battery.

If the receiver has no real-time clock, has not been set, or has a depleted battery, the receiver could use a reference year stored in nonvolatile memory, such as an EEPROM (electrically erasable programmable read-only memory), which is updated whenever the receiver is operated. If the reference year is no further back in time than one GPS week cycle, then the receiver could again determine the current date by noting that the year derived from the current week number cannot be earlier than the reference year.

If the receiver lacks a battery backup or EEPROM, it could still use the release date coded in the receiver's firmware. However, in this case, the receiver cannot determine the correct week cycle if the difference between the reference date and the current date is more than one week cycle or about 19.6 years.

Pinning Down the Problem. How a particular receiver will react to the EOW rollover will vary and depend on how the manufacturer programmed its firmware. Other than a careful examination of the firmware code, the only way to check if a particular receiver will work normally through the rollover is by testing it with a radio-frequency signal simulator. The simulator can be programmed to transmit synthetic GPS signals, including the navigation message, for a specified time interval before and after the scheduled rollover epoch. One should also conduct tests using various dates in the next GPS week cycle.

It is probably safe to say that all receivers with the latest firmware versions currently on the market will not experience problems during or after the rollover. Older receivers may or may not have trouble, and the severity of the problem could vary from manufacturer to manufacturer and model to model. Some manufacturers are offering firmware upgrades for selected older receivers to cure rollover problems. To determine if a particular receiver will have difficulty and whether a firmware upgrade is available, one should contact the manufacturer. Some companies have posted the relevant information on their Internet sites. Consult the links listed in the "Further Reading" sidebar.

When the rollover occurs, some noncompliant receivers might simply report an incorrect date, or they may incorrectly determine the position of satellites from the ephemeris transmitted in the navigation message, therefore computing and displaying incorrect receiver positions. Alternatively, they might sense that an error has occurred and simply refuse to calculate a position. Such problems may or may not go away if the receiver has its power cycled off and on.

If a noncompliant receiver is switched on after the rollover occurs, it could experience longer startup times or fail to lock onto satellites. Yet, it might be prompted to start working normally by performing an *auto locate* or *search the sky* operation in order to acquire satellites, collect a fresh almanac, and perform navigation functions.

Some units may avoid trouble during the first year or years of the new GPS week cycle but encounter it later when the difference between the programmed reference year and the current year exceeds the length of one GPS week cycle.

CONCLUSION

The popular press has characterized the GPS EOW rollover as a bug in the system. But GPS is no more "buggy" than our calendar, which resets itself every 365 days (or 366 days in a leap year) to the first of January. In fact, the GPS Joint Program Office is examining the operations of the satellites and control segment components to ensure that they are EOW rollover and Y2K compliant. They have already declared the satellites and satellite support systems to be compliant and are currently modifying the control segment to achieve compliance by January 1999.

That will leave only some old non-compliant receivers at risk come August 22, 1999. Or, to paraphrase Cassius in Shakespeare's *Julius Caesar*:

> *The fault, dear user, is not in our satellites,*
> *But in our receivers, that they are underlings.* ■