GPS Creations

Application Notes

How does “Parity Checking” used by GPS Satellites Work?

February 25, 2004

Written by Douglas Baker
Software example provided by Dr. Cliff Kelley

How does “data parity checking” used by the GPS satellites work?

Not a lot of information is available on this subject. For those that want to know, the following article pertains to the error detection method used for determining when the data transmitted by the GPS satellites has been correctly decoded.

All of the 50 bit-per-second data transmitted by the GPS satellites is broken up into 30 bit words, 24 bits of data and 6 bits of parity. Each sub-frame has 10 words of 300 bits. And the parity checking test of each 30-bit word builds on the last two bits of parity of the previous word. See the section on “Parity Rules” for more information.

The following six parity encoding equations are listed in the ICD-GPS-200C, page 136.

$D_{25} = D_{29}^{*} \oplus d_1 \oplus d_2 \oplus d_3 \oplus d_5 \oplus d_6 \oplus d_{10} \oplus d_{11} \oplus d_{12} \oplus d_{13} \oplus d_{14} \oplus d_{17} \oplus d_{18} \oplus d_{20} \oplus d_{23}$
$D_{26} = D_{30}^{*} \oplus d_2 \oplus d_3 \oplus d_4 \oplus d_6 \oplus d_7 \oplus d_{11} \oplus d_{12} \oplus d_{13} \oplus d_{14} \oplus d_{15} \oplus d_{18} \oplus d_{19} \oplus d_{21} \oplus d_{24}$
$D_{27} = D_{29}^{*} \oplus d_1 \oplus d_3 \oplus d_4 \oplus d_5 \oplus d_7 \oplus d_8 \oplus d_{12} \oplus d_{13} \oplus d_{14} \oplus d_{15} \oplus d_{16} \oplus d_{19} \oplus d_{20} \oplus d_{22}$
$D_{28} = D_{30}^{*} \oplus d_2 \oplus d_4 \oplus d_5 \oplus d_6 \oplus d_8 \oplus d_9 \oplus d_{13} \oplus d_{14} \oplus d_{15} \oplus d_{16} \oplus d_{17} \oplus d_{20} \oplus d_{21} \oplus d_{23}$
$D_{29} = D_{30}^{*} \oplus d_1 \oplus d_3 \oplus d_5 \oplus d_6 \oplus d_7 \oplus d_9 \oplus d_{10} \oplus d_{14} \oplus d_{15} \oplus d_{16} \oplus d_{17} \oplus d_{18} \oplus d_{21} \oplus d_{22} \oplus d_{24}$
$D_{30} = D_{29}^{*} \oplus d_3 \oplus d_5 \oplus d_6 \oplus d_8 \oplus d_9 \oplus d_{10} \oplus d_{11} \oplus d_{13} \oplus d_{15} \oplus d_{19} \oplus d_{22} \oplus d_{23} \oplus d_{24}$

D_{25}, D_{26}, D_{27}, D_{28}, D_{29}, D_{30} are the six parity bits which are appended to the end of each 24 bits of data, making up the full 30 bit “Word”. And D_{29}^{*} & D_{30}^{*} are the last two bits of parity of the previous 30 bit word.

The following “ones and zeros” 30-bit table (Table 1) can be constructed from the six equations above by putting a “1” in place of d_1 through d_{24} where these values are present in the equation:

d1	d2	d3	d4	d5	d6	d7	d8	d9	d10	d11	d12	d13	d14	d15	d16	d17	d18	d19	d20	d21	d22	d23	d24	d25	d26	d27	d28	d29	d30	
1	1	1	0	1	1	0	0	0	1	1	1	1	1	0	0	1	1	0	1	0	0	1	0	0	0	0	0	0	0	PB1
0	1	1	1	0	1	1	0	0	0	1	1	1	1	1	0	0	1	1	0	1	0	0	1	0	0	0	0	0	0	PB2
1	0	1	1	1	0	1	1	0	0	0	1	1	1	1	1	0	0	1	1	0	1	0	0	0	0	0	0	0	0	PB3
0	1	0	1	1	1	0	1	1	0	0	0	1	1	1	1	1	0	0	1	1	0	1	0	0	0	0	0	0	0	PB4
1	0	1	0	1	1	1	0	1	1	0	0	0	1	1	1	1	1	0	0	1	1	0	1	0	0	0	0	0	0	PB5
0	0	1	0	1	1	0	1	1	1	1	0	1	0	1	0	0	0	1	0	0	1	1	1	0	0	0	0	0	0	PB6

PB1 = 0x3b1f3480 PB2 = 0x1d8f9a40 PB3 = 0x2ec7cd00

PB4 = 0x1763e680 PB5 = 0x2bb1f340 PB6 = 0x0b7a89c0

Table 1, Determination of PB1 through PB6

Here, numbers PB1 thru PB6 are the hexadecimal values obtained by substituting “ones” and “zeros” into the six parity equations from the ICD-GPS-200C. PB1 through PB6 can now be used in the software program (see appendix A) for doing parity tests on the data received from the GPS satellites.

Parity Check Rules:

1. Last two parity bits of Word # 2, sub-frame #1 are always zero. Bits 23 & 24 in this word have to be solved in order for this condition to be true.

2. Bits #29 and #30 of each parity block are used in determining the parity of the next 30-bit word.

3. Bit #30, if set to a "1", determines if the data bits in the following word must have a "one's complement" operation done on them before doing the parity check.

The next drawing (Figure 1) shows actual data bits as transmitted by a GPS satellite. The drawing shows three 30-bit words of sub-frame # 1. The parity bits can be seen at the end of each word.

The first word contains the preamble value used to detect the beginning of sub-frame #1.

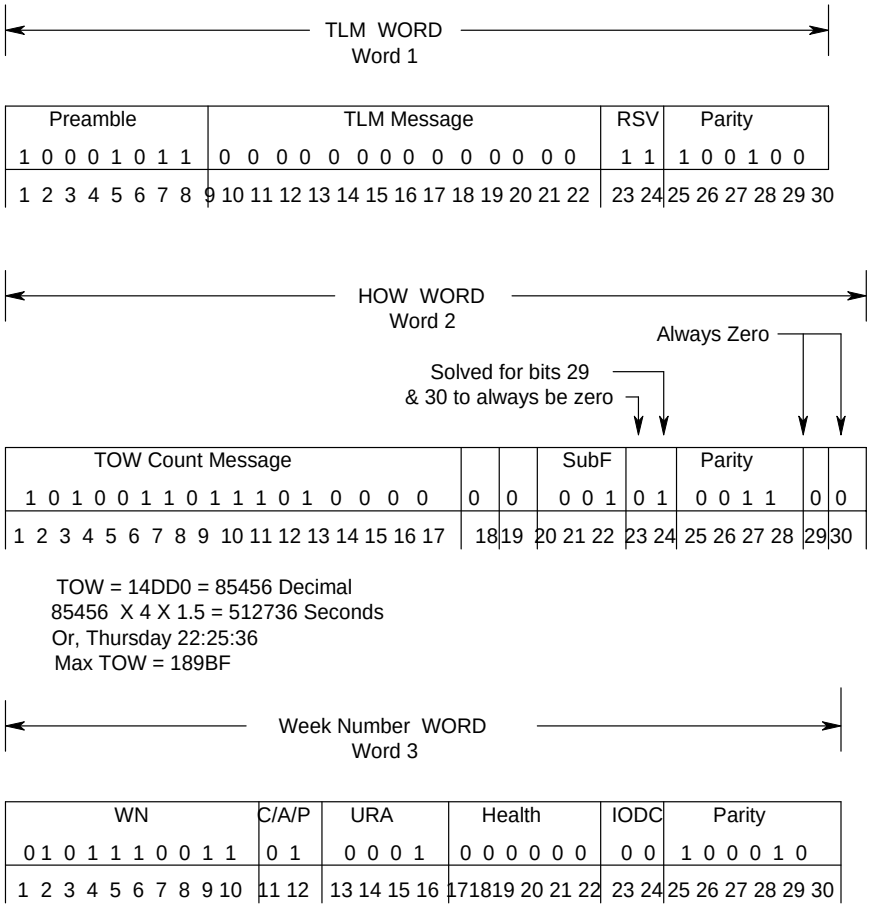
The first word is commonly referred to as the Telemetry Word or TLM Word. The second word is referred to as the Hand Over Word or HOW. The second word also contains the sub-frame number, bits 20, 21 and 22. The sub-frame number is "One" through "Five". There are five sub-frames making up a frame or a total of 1500 bits of data. The third word shown has the week number information.

Note that the drawing has a breakdown of the TOW and "Week Number" bits, what they represent and how the time in seconds change by six seconds at the start of each sub-frame. The week number (173 hex in this example) can represent the week of February 15, 1987 or it could be the week of October 1, 2006 or any multiple of 1024 weeks from February 15, 1987. It is up to the GPS receiver to determine which 1024 week epoch it is in.

How the program works

The C++ program shown in appendix A starts by listing a number of pre-defined values. There are ten "Long Words" which are the 30 bit values taken from what was transmitted by a GPS satellite. These 30-bit words each contain 24 bits of data and 6 bits of parity. The "parity_check" subroutine is called ten times, once for each of these words. The parity_check routine pulls out the six parity bits of each word and assigns it the name of "w_parity". Then the parity_check routine does a parity check on the full 30-bit word by doing the exclusive-or test using the bit assignments described above. The results of each six bits of parity are assigned the name "parity". If "parity" equals "w_parity", then the test assumes the 24 bits of data are correct. Note: this process may permit undetected bit errors of three or more bits. It is only useful for detecting bit errors of two or less! Remember – this data scheme goes back to the1970's and data error detection and correction methods were not as sophisticated as they are today. On the last page of Appendix A, a screen shot is taken showing the output of this program.

Example of Data Bits for First 3 Words of Sub Frame # 1



WN = 173 = 371 Decimal = 2/15/1987 or, could be: 10/1/2006

Max WN = 1111111111 = 3FF = 1023 Decimal

Figure 1

Appendix A

```
/*************************************************************************
  Partest

  Program written by D.Baker - by taking Cliff Kelley's routines and modifying
  them to be used where a 30 bit string is used as a "word" value ( 24 bits
  of data and 6 bits of parity) and checks the parity to see if correct.


*************************************************************************/
/*
Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions are
met:
                                CONDITIONS

1. Redistributions of GPSRCVR source code must retain the above copyright
notice, this list of conditions, and the following disclaimer.

2. Redistributions in binary form must contain the above copyright
notice, this list of conditions and the following disclaimer in the
documentation and/or other materials provided with the distribution.

3. All modifications to the source code must be clearly marked as
such.  Binary redistributions based on modified source code must be
clearly marked as modified versions in the documentation and/or other
materials provided with the distribution.

4. Notice must be given of the location of the availability of the
unmodified current source code, e.g.,
        http://www.Kelley.com/
or
        ftp://ftp.Kelley.com
in the documentation and/or other materials provided with the
distribution.

5. All advertising and published materials mentioning features or use
of this software must display the following acknowledgment:  "This
product includes software developed by Clifford Kelley and other
contributors."

6. The name of Clifford Kelley may not be used to endorse or promote
products derived from this software without specific prior written
permission.

                                DISCLAIMER

This software is provided by Clifford Kelley and contributors "as is" and
any expressed or implied warranties, including, but not limited to, the
implied warranties of merchantability and fitness for a particular
purpose are disclaimed.  In no event shall Clifford Kelley or
contributors be liable for any direct, indirect, incidental, special,
exemplary, or consequential damages (including, but not limited to,
procurement of substitute goods or services; loss of use, data, or
profits; or business interruption) however caused and on any theory of
liability, whether in contract, strict liability, or tort (including
negligence or otherwise) arising in any way out of the use of this
software, even if advised of the possibility of such damage.
*/
```

```
#include  <stdio.h>
#include  <stdlib.h>
#include  <conio.h>
#include  <math.h>
#include  <string.h>
#include  <io.h>
#include  <Dos.h>

unsigned long test_l[33]={0x00000000L,              // single bit set numbers
  0x00000001L,0x00000002L,0x00000004L,0x00000008L,  // for testing bit positions
  0x00000010L,0x00000020L,0x00000040L,0x00000080L,
  0x00000100L,0x00000200L,0x00000400L,0x00000800L,
  0x00001000L,0x00002000L,0x00004000L,0x00008000L,
  0x00010000L,0x00020000L,0x00040000L,0x00080000L,
  0x00100000L,0x00200000L,0x00400000L,0x00800000L,
  0x01000000L,0x02000000L,0x04000000L,0x08000000L,
  0x10000000L,0x20000000L,0x40000000L,0x80000000L};

 void  parity_check();
 inline int  bit_test_l(unsigned long ,char);

 // The following are 30 bit SV messages (words) and the last 6 bits of each
 // are the parity bits.  These words were taken from live SV transmissions.
 // and they are in the order as received, i.e., word 1 is first.
 // These are 10 words making up a complete subframe.
 long word[10]={0x22c000e4L,0x29ba014cL,0x17344022L,0x00000029L,0x3fffffd6L,
                          0x00000029L,0x3fffffd6L,0x01dfa435L,0x3fc00086L,0x3fe9cfd8L};
 char d29=0,d30=0;
 int j;
 void main(void)
{
   printf("Program used to examine various GPS words (30 bits) for parity\n");
   for(j=0;j<=9;j++)  //Do it for all ten words listed above
   {
   //printf("d29= %d,d30= %d, Test # %d\n",d29,d30,j);
   parity_check();  // call parity_check routine
   } //end for
  } //end main
/*****************************************************************************
FUNCTION parity_check(void)
RETURNS  None.

PARAMETERS None.

PURPOSE  checks the parity of the 5 subframes of the nav message

WRITTEN BY
          Clifford Kelley

*****************************************************************************/
```

```
void  parity_check()
{
  long pb1=0x3b1f3480L,pb2=0x1d8f9a40L,pb3=0x2ec7cd00L;
  long pb4=0x1763e680L,pb5=0x2bb1f340L,pb6=0x0b7a89c0L;
//These are the expressions developed from the six parity encoding equations
//listed on the first page of the document
  int exor(char, long);
  int  parity,w_parity,p_error;  //parity and w_parity are 6 bit nos
  int err_bit=0;
  char b_1,b_2,b_3,b_4,b_5,b_6;  //b_1 = bit 1,,, b_6 = bit 6
              p_error=0;

              w_parity = int(word[j] &0x3f);        //take all 30 bits and "AND" it
                                                    // with 3F to get the 6 LSB's
                                                     // i.e., get rid of bits 1 thru 24

      printf("w_parity value = %2x\n",w_parity);

      // now do the exclusive OR functions
      b_1=exor(d29,word[j] & pb1) << 5; //MSB of parity
      b_2=exor(d30,word[j] & pb2) << 4; //
      b_3=exor(d29,word[j] & pb3) << 3; //
      b_4=exor(d30,word[j] & pb4) << 2; //
      b_5=exor(0,word[j] & pb5) << 1;   //
      b_6=exor(d29^d30,word[j] & pb6);  //LSB of parity

      parity=b_1+b_2+b_3+b_4+b_5+b_6; // so now, the parity value becomes a 6 bit
                                                  // number by combining b_1 thru b_6
      printf("Calculated parity value = %2x\n",parity);
      err_bit=0;
      if (parity != w_parity)
      {
          err_bit=1;
      }
      if (err_bit != 0)printf("Error - parity test does not equal w_parity\n");
      p_error=(p_error << 1) + err_bit;
                    if (d30==1)
                    {
                    word[j]=0x03fffffc0L & ~word[j]; //if d30=1, set current word
                    // to all one's and do a 1's complement
                    // above value = 0000 0011 1111 1111 1111 1111 1111 1100 0000
          }
      word[j]=word[j]>>6;
      // shift right 6 bits to get rid of the parity bits, leaving 24 good data bits
      // not used here but is needed in GPSRCVR program

      d29=(w_parity & 0x2) >>1;   // save d29 & d30 for next word calculation
      d30=w_parity & 0x1;     //d29 and d30 are last two bits of the six parity bits

    }  //end parity_check
```

```
/*****************************************************************************
FUNCTION exor(char bit, long parity)
RETURNS  None.

PARAMETERS
                        bit     char
                        parity  long

PURPOSE

WRITTEN BY: Clifford Kelley

*****************************************************************************/

  int exor(char bit, long parity)
  {
  char i;
  int result = 0;
  result=0;
  for (i=7;i<=30;i++)
  {
    if (bit_test_l(parity,i)) result++;
  }
  result=result%2;
  result=(bit ^ result) & 0x1;
  return(result);
  }

/*****************************************************************************
FUNCTION bit_test_l(unsigned long data, char bit_n)
RETURNS  int

PARAMETERS
          data    unsigned long
          bit_n   char

PURPOSE
        This function returns a 1 if bit number bit_n of data is 1
        else it returns a 0

WRITTEN BY: Clifford Kelley
*****************************************************************************/

inline int  bit_test_l(unsigned long data,char bit_n)
{
  int result;
  result=0;
  if (data & test_l[bit_n])result=1;
  return(result);
}
```

/***/

Screen shot of results running above program: