Clarification for CCSDS CRC-16 Computation Algorithm

Jackson Pang, Kenneth Andrews and J Leigh Torgerson
Section 332M – Communications Networks
{jpang, kenneth.s.andrews, ltorgerson}@jpl.nasa.gov

May 4, 2006

Abstract – The description of CCSDS CRC-16 computation algorithm as explained in the *Telemetry: Summary of Concept and Rationale* [1] may cause misinterpretation of the algorithm. The computation procedure is clarified and example source code is provided. A recommended change to the CCSDS Green Book is also described so that it conveys the CRC computation procedure clearly.

Background

The CCSDS CRC-16 calculation, based on the International Communication Union's (formerly known as CCITT) ITU-T v.41 [2], uses the generator polynomial $G(X) = X^{16} + X^{12} + X^5 + 1$. The CRC-16 calculation procedure can be described using a Linear Feedback Shift Registers (LFSR) as follows

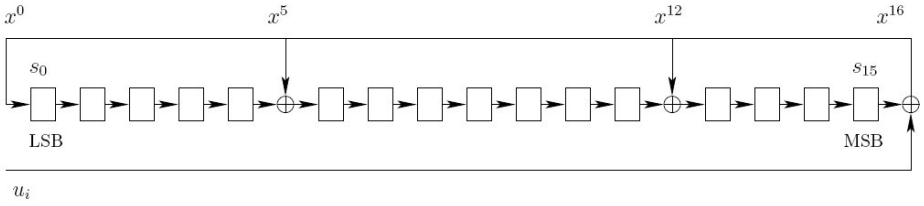

Figure 1: CCSDS implementation of CRC-16 [3]

Recently, some discrepancies were found in several software implementations of the algorithm. Specifically the ordering of the LFSR was reversed, effectively changing the generator polynomial to $G_w(X) = X^{16} + X^{10} + X^3 + 1$. In addition, the CRC was appended to the message stream in a reversed order at the byte level. Consequently, the resulting CRC is not compliant with the CCSDS Blue Book recommendation, and will not be interoperable with commercial ground station equipment or spacecraft that implement the algorithm correctly.

Recommended CCSDS Document Modification

The figures described on page D-4 of [1] should be modified so that the shift registers are numbered as shown in Figure 1. This convention is consistent with the CCSDS bit numbering scheme where the most significant bit is the first bit transferred (See Fig. 2).

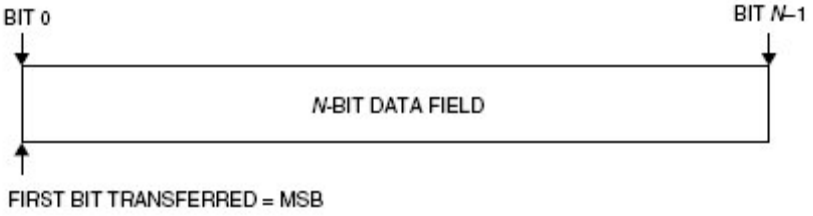

Figure 2: CCSDS Bit Numbering Convention [5]

The following are the improved Figures D-1 and D-2 in [1].

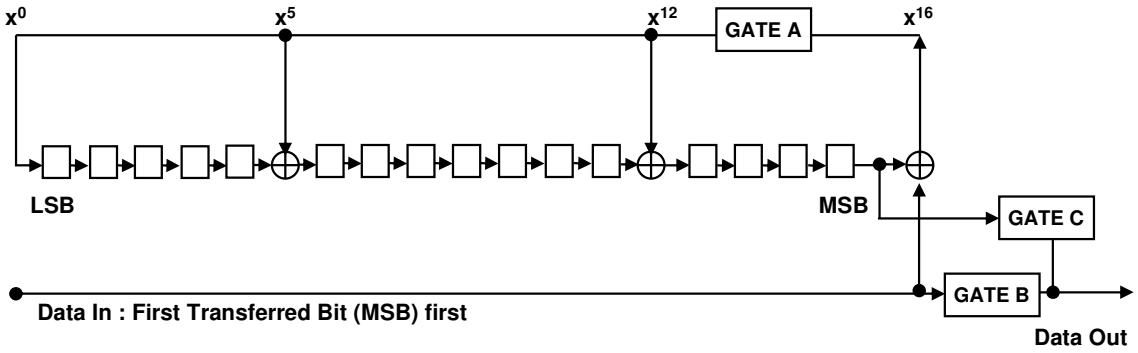

Figure 3: Improved CCSDS CRC-16 Encoder Diagram

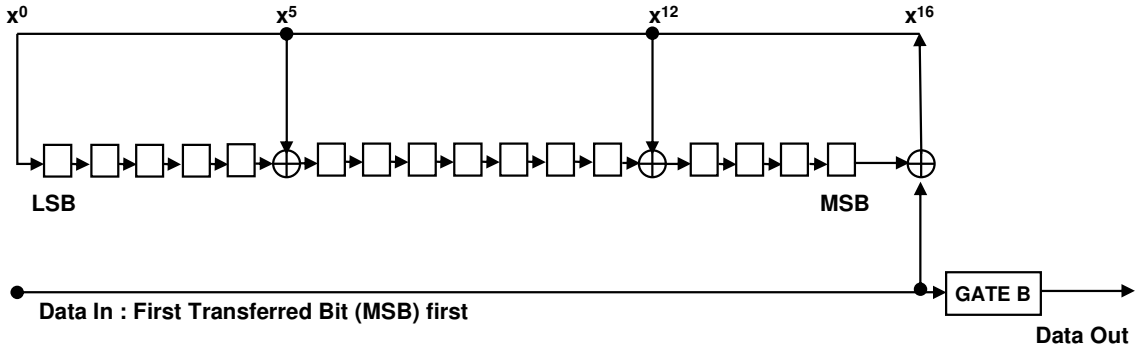

Figure 4: Improved CCSDS CRC-16 Decoder Diagram

To be consistency across all the CCSDS documentation of CRC calculation diagrams, the following two diagrams are the improved version of Figures A-1 and A-2 of [6]. They use the CCSDS CRC-32 generator polynomial
$G(X) = X^{32} + X^{23} + X^{21} + X^{11} + X^2 + 1$.

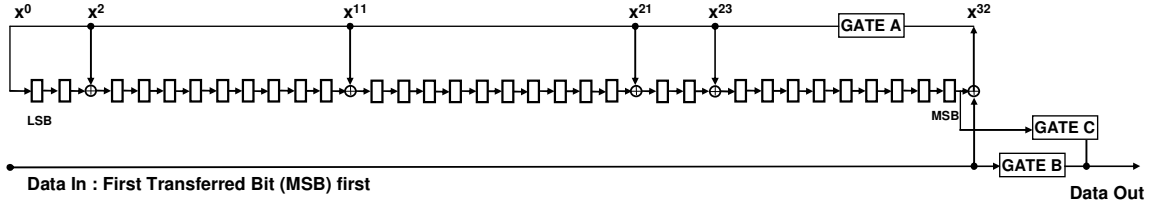

Figure 5: Improved CCSDS CRC-32 Encoder Diagram

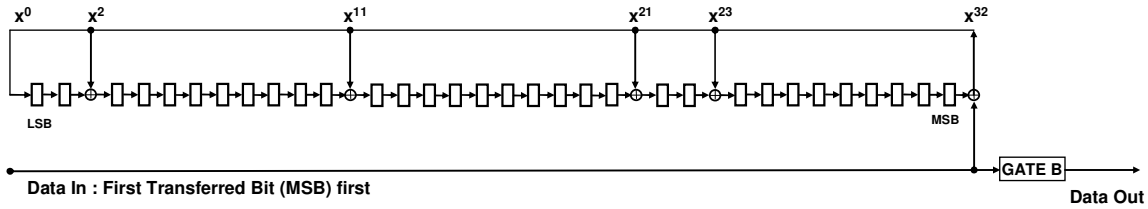

Figure 6: Improved CCSDS CRC-32 Decoder Diagram

Recommended Implementation

The CCSDS CRC-16 calculation produces the 16-bit CRC syndrome value that needs to be appended at the end of the message being encoded for transmission. The appending shall be done so that the bit ordering does not change. For example, the following message stream produces the CRC syndrome registers to hold 0x75FB (most significant bit to least significant bit).

0x06 0x00 0x0c 0xf0 0x00 0x04 0x00 0x55 0x88 0x73 0xc9 0x00 0x00 0x05 0x21

The CRC value shall be appended to the message stream as follows to produce the correct CRC value of 0x0000 at the receiving entity when it calculates the CRC over the whole message stream.

0x06 0x00 0x0c 0xf0 0x00 0x04 0x00 0x55 0x88 0x73 0xc9 0x00 0x00 0x05 0x21 **0x75 0xFB**

A correct example CRC-16 calculation in C programming language using both table driven and serial calculation approaches are described below [4].

```
static U16 crc_table[256] = {
    0x0000, 0x1021, 0x2042, 0x3063, 0x4084, 0x50a5, 0x60c6, 0x70e7,
    0x8108, 0x9129, 0xa14a, 0xb16b, 0xc18c, 0xd1ad, 0xe1ce, 0xf1ef,
    0x1231, 0x0210, 0x3273, 0x2252, 0x52b5, 0x4294, 0x72f7, 0x62d6,
    0x9339, 0x8318, 0xb37b, 0xa35a, 0xd3bd, 0xc39c, 0xf3ff, 0xe3de,
    0x2462, 0x3443, 0x0420, 0x1401, 0x64e6, 0x74c7, 0x44a4, 0x5485,
    0xa56a, 0xb54b, 0x8528, 0x9509, 0xe5ee, 0xf5cf, 0xc5ac, 0xd58d,
    0x3653, 0x2672, 0x1611, 0x0630, 0x76d7, 0x66f6, 0x5695, 0x46b4,
    0xb75b, 0xa77a, 0x9719, 0x8738, 0xf7df, 0xe7fe, 0xd79d, 0xc7bc,
    0x48c4, 0x58e5, 0x6886, 0x78a7, 0x0840, 0x1861, 0x2802, 0x3823,
    0xc9cc, 0xd9ed, 0xe98e, 0xf9af, 0x8948, 0x9969, 0xa90a, 0xb92b,
    0x5af5, 0x4ad4, 0x7ab7, 0x6a96, 0x1a71, 0x0a50, 0x3a33, 0x2a12,
    0xdbfd, 0xcbdc, 0xfbbf, 0xeb9e, 0x9b79, 0x8b58, 0xbb3b, 0xab1a,
    0x6ca6, 0x7c87, 0x4ce4, 0x5cc5, 0x2c22, 0x3c03, 0x0c60, 0x1c41,
    0xedae, 0xfd8f, 0xcdec, 0xddcd, 0xad2a, 0xbd0b, 0x8d68, 0x9d49,
    0x7e97, 0x6eb6, 0x5ed5, 0x4ef4, 0x3e13, 0x2e32, 0x1e51, 0x0e70,
    0xff9f, 0xefbe, 0xdfdd, 0xcffc, 0xbf1b, 0xaf3a, 0x9f59, 0x8f78,
    0x9188, 0x81a9, 0xb1ca, 0xa1eb, 0xd10c, 0xc12d, 0xf14e, 0xe16f,
    0x1080, 0x00a1, 0x30c2, 0x20e3, 0x5004, 0x4025, 0x7046, 0x6067,
    0x83b9, 0x9398, 0xa3fb, 0xb3da, 0xc33d, 0xd31c, 0xe37f, 0xf35e,
    0x02b1, 0x1290, 0x22f3, 0x32d2, 0x4235, 0x5214, 0x6277, 0x7256,
    0xb5ea, 0xa5cb, 0x95a8, 0x8589, 0xf56e, 0xe54f, 0xd52c, 0xc50d,
    0x34e2, 0x24c3, 0x14a0, 0x0481, 0x7466, 0x6447, 0x5424, 0x4405,
    0xa7db, 0xb7fa, 0x8799, 0x97b8, 0xe75f, 0xf77e, 0xc71d, 0xd73c,
    0x26d3, 0x36f2, 0x0691, 0x16b0, 0x6657, 0x7676, 0x4615, 0x5634,
```

```
    0xd94c, 0xc96d, 0xf90e, 0xe92f, 0x99c8, 0x89e9, 0xb98a, 0xa9ab,
    0x5844, 0x4865, 0x7806, 0x6827, 0x18c0, 0x08e1, 0x3882, 0x28a3,
    0xcb7d, 0xdb5c, 0xeb3f, 0xfb1e, 0x8bf9, 0x9bd8, 0xabbb, 0xbb9a,
    0x4a75, 0x5a54, 0x6a37, 0x7a16, 0x0af1, 0x1ad0, 0x2ab3, 0x3a92,
    0xfd2e, 0xed0f, 0xdd6c, 0xcd4d, 0xbdaa, 0xad8b, 0x9de8, 0x8dc9,
    0x7c26, 0x6c07, 0x5c64, 0x4c45, 0x3ca2, 0x2c83, 0x1ce0, 0x0cc1,
    0xef1f, 0xff3e, 0xcf5d, 0xdf7c, 0xaf9b, 0xbfba, 0x8fd9, 0x9ff8,
    0x6e17, 0x7e36, 0x4e55, 0x5e74, 0x2e93, 0x3eb2, 0x0ed1, 0x1ef0,
};
/*****************************************************************
 *
 * FUNCTION:
 *      tc__fec_sdlc
 *
 * INPUTS:
 *      seed            - (U16) initial value of check bits.
 *      buf             - (unsigned char *) pointer to the buffer of
 *                        data over which you wish to generate check
 *                        bits.
 *      len             - (int) number to bytes of data in the buffer.
 *
 * OUTPUTS:
 *
 * RETURNS:
 *                      - (U16) the checkbits.
 *
 * EXTERNALLY READ:
 *      sdlc_table      - (U16)[256] the lookup table for the CCITT SDLC
 *                        generator polynomial (local to this module).
 *
 * EXTERNALLY MODIFIED:
 *
 * DESCRIPTION:
 *      This function implements CRC generation with the CCITT SDLC error
 *      polynomial (X16 + X12 + X5 + 1).  You must provide it with an
 *      initial seed value, a pointer to a data buffer, and the byte length
 *      of the data buffer.  It will return the unsigned 16-bit CRC.
 *
 *      You may use this function to generate a CRC over data in scattered
 *      storage by making multiple calls to it.  Just make sure that you
 *      pass a seed of 0xFFFF on the first call.  On subsequent calls, pass
 *      a seed containing the return value of the previous call.
 *
 */
U16
tc__fec_sdlc_s(U16 seed, unsigned char *buf, int len)
{
    U16 crc, t;
    unsigned char *p;
    crc = seed;
    p = buf;
    while (len--) {
        crc = crc_table[((crc >> 8) ^ *p++) & 0xff] ^ (crc << 8);
    }
    return (U16)crc;
}
#ifdef SLOWER_METHOD
U16
tc__fec_sdlc_f(U16 seed, unsigned char *buf, int len)
{
    unsigned short crc, t;
    unsigned short ch, xor_flag;
    int i, count;
    crc = seed;
    count = 0;
    while (len--)     {
        ch = buf[count++];
        ch<<=8;
        for(i=0; i<8; i++)
        {
            if ((crc ^ ch) & 0x8000)
```

```
            {
                xor_flag = 1;
            }
            else
            {
                xor_flag = 0;
            }
            crc = crc << 1;
            if (xor_flag)
            {
                crc = crc ^ 0x1021;
            }
            ch = ch << 1;
        }
    }
    return (unsigned short)crc;
}
#endif
```

References

[1] *Telemetry: Summary of Concept and Rationale*, Consultative Committee for Space Data Systems (CCSDS), Dec 1987. [Online]. Available: http://public.ccsds.org/publications/archive/100x0g1.pdf

[2] *Code-Independent Error-Control System: Data Communication over the Telephone Network*, International Telecommunication Union, ITU-T Fascicle VIII.1 – Rec. V.41.

[3] Andrews, Kenneth, *Fast Methods for Performing CRC Computation,* Feb 2006, Internal JPL Memo.

[4] Ho, Son, AMMOS Command Subsystem Source Code Repository, March 24, 2006.

[5] *Telemetry Space Data Link Protocol*, Consultative Committee for Space Data Systems (CCSDS), Sept 2003. [Online]. Available: http://public.ccsds.org/publications/archive/132x0b1.pdf

[6] *Proximity-1 Space Link Protocol – Coding and Synchronization Sublayer*, Consultative Committee for Space Data Systems (CCSDS), April 2003. [Online]. Available: http://public.ccsds.org/publications/archive/221x2b1.pdf