Below are C programs and equations to generate CRC and scrambled output for Annex A and C in the SAS draft. This revision includes an example scrambler diagram, changes sample scramble code to a serial shift register, scrambled data output for all zeros input, and example XOR equations for generating a 32 bit scramble value.

The following is an example C program that generates the value for the CRC field in frames. The inputs are the data dwords for the frame and the number of data dwords.

```c
static unsigned long data_dwords[] = {0x06D0B992L, 0x00B5DF59L, 0x00000000L, 0x00000000L, 0x1234FFFFL, 0x00000000L, 0x00000000L, 0x00000000L, 0x08000012L, 0x01000000L, 0x00000000L, 0x00000000L}; /* example data dwords */

unsigned long *frame; /* pointer to the data dwords */
unsigned long int length; /* number of data dwords */

unsigned long calculate_crc(*frame, length);
unsigned long crc;

crc = calculate_crc(data_dwords, 13);

unsigned long calculate_crc(*frame, length) /* returns crc value */
{
  long poly = 0x04C11DB7L;
  unsigned long crc_gen, x;
  union {unsigned long lword; unsigned char byte[4];} b_access;
  static unsigned char xpose[] = {0x0, 0x8, 0x4, 0xC, 0x2, 0xA, 0x6, 0xE, 0x1, 0x9, 0x5, 0xD, 0x3, 0xB, 0x7, 0xF};

  int i, j, fb;

  crc_gen = ~0; /* seed generator with all ones */
  for (i=0; i<length; i++)
  {
    x = *frame++; /* get word */
    b_access.lword = x; /* transpose bits in byte */
    for (j=0; j<4; j++)
    {
      b_access.byte[j] = xpose[b_access.byte[j]>>4] | xpose[b_access.byte[j]&0xF]<<4;
    }
    x = b_access.lword;
  }

  for (j=0; j<32; j++) /* serial shift register implementation */
  {
    fb = ((x & 0x80000000L)>0) ^ ((crc_gen & 0x80000000L)>0);
    x <<= 1;
    crc_gen <<= 1;
    if (fb)
      crc_gen ^= poly;
  }
}
```
These equations generate the 32 bit Cyclic Redundancy Check for frame transmission. The ^ symbol represents an XOR operation.

crc00 = d00^d06^d09^d10^d12^d16^d24^d26^d28^d30^d31;
crc01 = d00^d01^d06^d07^d09^d11^d12^d13^d16^d17^d18^d24^d25^d26^d28^d30^d31;
crc02 = d00^d01^d02^d06^d07^d08^d09^d13^d14^d15^d16^d18^d19^d20^d24^d26^d28^d30^d31;
crc03 = d00^d01^d02^d03^d06^d07^d08^d09^d14^d15^d16^d17^d18^d19^d20^d25^d27^d31;
crc04 = d00^d02^d03^d04^d06^d08^d11^d12^d15^d18^d19^d20^d24^d25^d29^d30^d31;
crc05 = d00^d01^d03^d04^d06^d07^d08^d09^d10^d11^d12^d15^d18^d20^d21^d24^d27^d31;
crc06 = d01^d02^d04^d05^d06^d07^d08^d11^d14^d20^d21^d22^d25^d29^d30^d31;
crc07 = d00^d02^d03^d05^d07^d08^d10^d15^d16^d21^d22^d23^d24^d25^d28^d29^d31;
crc08 = d00^d01^d03^d04^d08^d10^d11^d12^d17^d22^d23^d24^d25^d28^d29^d31;
crc09 = d01^d02^d04^d05^d09^d11^d12^d13^d18^d21^d23^d24^d25^d28^d31;
crc10 = d00^d02^d03^d05^d09^d13^d14^d16^d19^d26^d28^d29^d31;
crc11 = d00^d01^d03^d04^d09^d12^d14^d15^d16^d17^d20^d24^d25^d26^d27^d28^d31;
crc12 = d00^d01^d02^d04^d05^d06^d09^d12^d13^d15^d17^d18^d21^d24^d27^d30^d31;
crc13 = d01^d02^d03^d05^d06^d07^d10^d13^d14^d16^d18^d19^d22^d25^d28^d31;
crc14 = d02^d03^d04^d06^d07^d08^d11^d14^d15^d17^d19^d20^d23^d26^d29^d31;
crc15 = d03^d04^d05^d07^d08^d09^d12^d15^d16^d18^d20^d21^d24^d27^d30;
crc16 = d00^d04^d05^d08^d12^d13^d17^d19^d21^d22^d24^d26^d29^d30^d31;
crc17 = d01^d05^d06^d09^d13^d14^d18^d20^d22^d23^d25^d27^d30^d31;
crc18 = d02^d06^d07^d10^d14^d15^d19^d21^d23^d24^d26^d28^d31;
crc19 = d03^d07^d08^d11^d15^d16^d19^d20^d22^d24^d25^d27^d29^d31;
crc20 = d04^d08^d09^d12^d16^d17^d21^d23^d25^d26^d28^d30;
crc21 = d05^d09^d10^d13^d17^d18^d22^d24^d26^d27^d29^d31;
crc22 = d00^d09^d11^d12^d14^d16^d18^d19^d23^d24^d26^d27^d29^d31;
crc23 = d00^d01^d06^d09^d13^d15^d16^d17^d19^d20^d26^d27^d29^d31;
crc24 = d01^d02^d07^d10^d14^d16^d17^d18^d20^d21^d27^d28^d30;
crc25 = d02^d03^d08^d11^d15^d17^d18^d19^d21^d22^d28^d29^d31;
crc26 = d00^d03^d04^d06^d10^d18^d19^d20^d22^d23^d24^d25^d26^d28^d31;
crc27 = d01^d04^d05^d07^d11^d19^d20^d21^d23^d24^d25^d26^d27^d29;
crc28 = d02^d05^d06^d08^d12^d20^d21^d22^d24^d25^d26^d27^d28^d30;
crc29 = d03^d06^d07^d09^d13^d21^d22^d23^d25^d26^d27^d28^d29^d31;
crc30 = d04^d07^d08^d10^d14^d22^d23^d24^d26^d27^d28^d29^d30;
crc31 = d05^d08^d09^d11^d15^d23^d24^d25^d27^d28^d29^d30^d31;

The following is an example of the CRC calculation for a 6 byte Read command IU.

INFORMATION UNIT TYPE

```
06h
```

2 of 6
HASHED DESTINATION DEVICE NAME for 500107534F0CFC88h  D0B992h
HASHED SOURCE DEVICE NAME for 50010B92B3CBF639h  B5DF59h
TIMEOUT  0
NUMBER OF FILL BYTES  0
COMMAND ID  0
TAG  1234h
TARGET PORT TRANSFER TAG  FFFFh
INFORMATION UNIT (CMD)
  LUN  0
  TASK ATTRIBUTE  0
  ADDITIONAL CDB LENGTH  0
  CDB (6 byte read block 12h)  080000120100h

Data dwords

06D0B992h
00B5DF59h
00000000h
00000000h
1234FFFFh
00000000h
00000000h
00000000h
00000000h
08000012h
01000000h
00000000h
00000000h

CRC = 3F4F1C26h

========================================================================

Data scrambler implementation example

Figure xx shows an example of a data scrambler. This example generates the value
to XOR with the dword input with two 16 bit parallel multipliers. 16 bits wide
is the maximum width for the multiplier as the generating polynomial is 16 bits.
The following is an example C program that generates the scrambled data dwords for transmission. The inputs are the data dword to scramble and control indication to reinitialize the residual value following an SOF.

The generator polynomial specified is:

\[ G(x) = x^{16} + x^{15} + x^{13} + x^{4} + 1 \]

For parallelized versions of the scrambler, the initialized value is selected to produce a first dword output of 0xC2D2768D for a dword input of all zeros.

```c
#include <stdio.h>

unsigned long scramble(bool reset, unsigned long dword);

void main1(void)
{
    FILE *f = fopen("crc_report.txt","w");
    if (f)
    {
        for (int i=0; i<12; i++)
            fprintf(f," %08X 
",scramble(i==0, 0));
        fclose(f);
    }
}

/* sample output (if input == 0)
0xC2D2768D
0x1F26B368
0xA508436C
0x3452D354
0x8A559502
0xBB1ABE1B
0xFA56B73D */
```
unsigned long scramble(bool reset, unsigned long dword)\n{ static unsigned short scramble;\nif (reset)\n    scramble = 0xFFFF;\nfor (int i=0; i<32; i++) /* serial shift register implementation */\n{ dword ^= scramble&0x8000?1<<i:0;\n    scramble = (scramble<<1) ^ (scramble&0x8000? poly : 0);\n}\nreturn dword;\n}

These equations generate the scrambled bytes to XOR with dwords before transmission and dword reception to recover the original data. The ^ symbol represents an XOR operation. The initialized value for d[15-0] is 0xF0F6 in this example.

The following examples of the scrambled data output.

<table>
<thead>
<tr>
<th>Data</th>
<th>scrambled</th>
<th>Dword</th>
<th>output</th>
</tr>
</thead>
<tbody>
<tr>
<td>06D0B992h</td>
<td>C402CF1Fh</td>
<td>00B5DF59h</td>
<td>1F936C31h</td>
</tr>
<tr>
<td>00000000h</td>
<td>A508436Ch</td>
<td>00000000h</td>
<td>3452D354h</td>
</tr>
<tr>
<td>1234FFFFh</td>
<td>98616AFDh</td>
<td>00000000h</td>
<td>B81ABE1Bh</td>
</tr>
<tr>
<td>00000000h</td>
<td>F0809C41h</td>
<td>08000012h</td>
<td>7C7FC358h</td>
</tr>
<tr>
<td>01000000h</td>
<td>BF865291h</td>
<td>00000000h</td>
<td>7A6FA7B6h</td>
</tr>
<tr>
<td>3F4F1C26h</td>
<td>CF79E22Ah</td>
<td>00000000h</td>
<td>3163E6D6h</td>
</tr>
<tr>
<td></td>
<td>C2D2768Dh</td>
<td>00000000h</td>
<td>3452D354h</td>
</tr>
<tr>
<td></td>
<td>1F26B368h</td>
<td>00000000h</td>
<td>8A559502h</td>
</tr>
<tr>
<td></td>
<td>A508436Ch</td>
<td>00000000h</td>
<td>53F60B1Bh</td>
</tr>
<tr>
<td></td>
<td>FA56B73Dh</td>
<td>00000000h</td>
<td>FA56B73Dh</td>
</tr>
<tr>
<td></td>
<td>BB1ABE1Bh</td>
<td>00000000h</td>
<td>BB1ABE1Bh</td>
</tr>
<tr>
<td></td>
<td>F0809C41h</td>
<td>08000012h</td>
<td>7C7FC358h</td>
</tr>
<tr>
<td></td>
<td>BF865291h</td>
<td>00000000h</td>
<td>7A6FA7B6h</td>
</tr>
<tr>
<td></td>
<td>7A6FA7B6h</td>
<td>00000000h</td>
<td>3163E6D6h</td>
</tr>
<tr>
<td></td>
<td>3163E6D6h</td>
<td>3F4F1C26h</td>
<td>CF79E22Ah</td>
</tr>
</tbody>
</table>

B00F2BCCh → 4039D5C0h \(\text{CRC dword}\)