Section: 61A.

Comment:
The 64/65 encapsulation is new to this document. In one of our previous comments, we still spotted a typo for one of the values, in addition members of our team had to confer quite a bit to come to agreements on what needs to be sent for a variety of cases. In order to increase the likeliness that everyone comes up with the same interpretation, we propose to include a C program that simulates the TPS-TC and includes a set of corner cases. Everyone would then be able to check the result of their TPS-TC output against the program.

Remedy:
Enclose a simple 'C' program and it's output logfile in a new section of 61A. The program is a simulation of the SHDSL EFM TC transmitter. The logfile contains a valid EFM bitstream reading left to right and then top to bottom. The stream includes an assortment of corner test cases. The program and output file is provided in the associated file kimpe_1_0309

```
/*
 *   802.3ah (EFM) 2BASE-TL (SHDSL) TC Transmitter simulator  from Section 61.2.3
*
 */


#include <stdio.h>

/* turn off to see unscrambled data for testing */
int useScrambler = 1;

/* test frame data */
char p0[] = {0x00, 0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07,
          0x08, 0x09, 0x0a, 0x0b, 0x0c, 0x0d, 0x0e, 0x0f,
          0x10, 0x11, 0x12, 0x13, 0x14, 0x15, 0x16, 0x17,
          0x18, 0x19, 0x1a, 0x1b, 0x1c, 0x1d, 0x1e, 0x1f,
          0x20, 0x21, 0x22, 0x23, 0x24, 0x25, 0x26, 0x27,
          0x28, 0x29, 0x2a, 0x2b, 0x2c, 0x2d, 0x2e, 0x2f,
          0x30, 0x31, 0x32, 0x33, 0x34, 0x35, 0x36, 0x37,
          0x38, 0x39, 0x3a, 0x3b, 0x3c, 0x3d, 0x3e, 0x3f};
char p1[] = {0x40, 0x41, 0x42, 0x43, 0x44, 0x45, 0x46, 0x47,
          0x48, 0x49, 0x4a, 0x4b, 0x4c, 0x4d, 0x4e, 0x4f,
          0x50, 0x51, 0x52, 0x53, 0x54, 0x55, 0x56, 0x57,
          0x58, 0x59, 0x5a, 0x5b, 0x5c, 0x5d, 0x5e, 0x5f,
          0x60, 0x61, 0x62, 0x63, 0x64, 0x65, 0x66, 0x67,
          0x68, 0x69, 0x6a, 0x6b, 0x6c, 0x6d, 0x6e, 0x6f,
          0x70, 0x71, 0x72, 0x73, 0x74, 0x75, 0x76, 0x77,
          0x78, 0x79, 0x7a, 0x7b, 0x7c, 0x7d, 0x7e, 0x7f,
```

```
        0x80, 0x81, 0x82, 0x83, 0x84, 0x85, 0x86, 0x87,
        0x88, 0x89, 0x8a, 0x8b, 0x8c, 0x8d, 0x8e, 0x8f};
char p2[] = {0x65, 0x43, 0x21};

#define NUM_CODEWORDS 14    /* number of 65 byte EFM codewords to transmit */

/*
 * Define a list of user frames to transmit
 *  NOTE: This list defines the set of test cases to transmit.
 */
struct frame {
  int  startingByteNum;    /* byte position at which frame is available to send */
  int  length;             /* number of bytes in ethernet frame */
  char *theBytes;          /* pointer to ethernet frame bytes */
} FrameList[] = {      /* To test: */
    {200, 64, p0},   /* vanilla frame, scrambler, C(k), crc */
    {389, 64, p0},   /* all data sync byte, sync splitting S/data/crc */
    {465, 50, p1},   /* end frame & start new frame in same codeword, C(0) */
    {530,  3, p2},   /* align small frame to span sync byte */
    {650, 64, p0},   /* S following sync byte */
    {700, 55, p1},   /* back-to-back frames, sync byte within crc */
    {0,0,0}          /* end test */
};


/* constants as per TC spec */

#define CODEWORD_BYTE_COUNT     65

#define SYNC_ALL_DATA           0x0f     /* all data sync byte */
#define SYNC_NOT_ALL_DATA       0xf0     /* not all data sync byte */

#define START_OF_FRAME_BYTE     0x50     /* start data byte */
#define IDLE_BYTE               0x00     /* idle data byte */

#define EFM_CRC_POLY 0x82f63b78  /* X**28 + X**27 + X**26 + X**25 + X**23
+
                          X**22 + X**20 + X**19 + X**18 + X**14 +
                          X**13 + X**11 + X**10 + X**09 + X**08 +
                          X**06 + X**00  (lsb is x**31) */


/* the EFM TC crc accumulator */
unsigned long CrcAccum;
void EfmCrcReset(void) {
  CrcAccum = 0xffffffff;
```

```
}

void EfmCrc(unsigned char TheByte) {
  int i;

  /* for all the bits in TheByte, lsb first */
  for( i=0; i<8; i++) {

    /* xor the lsb of TheByte with the x**31 of CrcAccum */
    int FeedBack = 0x01 & (CrcAccum ^ TheByte);

    TheByte  = TheByte  >> 1;
    CrcAccum = CrcAccum >> 1;
    if(FeedBack) {
      CrcAccum = CrcAccum ^ EFM_CRC_POLY;
    }
  }
}

unsigned long EfmScramblerHistory = 0;
unsigned char EfmScrambler(unsigned char ByteToScramble) {
  int i;
  unsigned char Result = 0;

  for(i=0; i<8; i++) {
    int Temp = 0;
    if(ByteToScramble & 1) {
      Temp = 1;
    }

    /* Efm scrambler polynomial as per figure 61-15 */
         /*  Payloads run thru scrambler lsb first as per figure 61-18 */
    if(useScrambler) {
      if(EfmScramblerHistory & (1 << 17)) {
        Temp ^= 1;
      }
      if(EfmScramblerHistory & (1 << 22)) {
        Temp ^= 1;
      }
    }

    ByteToScramble     >>= 1;
    EfmScramblerHistory <<= 1;
    Result             >>= 1;
    if( Temp ) {
      EfmScramblerHistory |= 1;
```

```
        Result              |= 0x80;
      }
    }
    return(Result);
}


/* run with an agrument to get test tags in output, else just the numbers */
main(int argc, char * argv[])
{
    int   ByteNum;
    int   UserFrameIndex   = 0;
    int   HaveUserFrame    = 0;

    int   FrameBytesToGo   = 0;
    char *FrameBytePointer = 0;
    int   NeedCZero        = 0;
    int   b;
    char  Foo[50];


    for(ByteNum=0; ByteNum < (CODEWORD_BYTE_COUNT *
NUM_CODEWORDS) ; ByteNum++) {
      unsigned char ByteToSend;
      int BytesLeftInCodeword = CODEWORD_BYTE_COUNT - (ByteNum %
CODEWORD_BYTE_COUNT);
      char *FrameTag = 0;

      /* decide what I'm doing */
      switch(ByteNum % CODEWORD_BYTE_COUNT) {
      case 0: /* output start of a codeword */
        if(FrameBytesToGo >= (CODEWORD_BYTE_COUNT-1)) {
          /* 64 or more bytes to go, send an all data codeword */
          ByteToSend = SYNC_ALL_DATA;
          FrameTag = "CODEWORD START (all data)";
        } else {
          ByteToSend = SYNC_NOT_ALL_DATA;
          FrameTag = "CODEWORD START (not all data)";
                              if( ByteNum == 0) FrameTag = "EFM bitstream reading right to
left.";
        }
        break;
      case 1: /* if a C(k) byte is needed */
        if((FrameBytesToGo && (FrameBytesToGo < (CODEWORD_BYTE_COUNT-
1))) || NeedCZero) {
          int kVal = FrameBytesToGo;
```

```
        /* output a C(k) */
        ByteToSend = 0x10 + kVal;
        /* calculate even parity */
        for(b=0x40; b; b=b>>1) {
          if(ByteToSend & b) {
            ByteToSend ^= 0x80;
          }
        }
        NeedCZero = 0;
        /* display C(k) with decimal k */
        sprintf(Foo,"    C(%d)",kVal);
        FrameTag = Foo;
        break;
      }
      /* else fall into default case */
    default:
      /* if I'm
       *   not sending a frame and
       *   there are more to send, and
       *   it's time to start (next frame is available), and
       *   the frame is not too short to start now (including S and crc bytes)
       */
      if(   (FrameBytesToGo == 0)
        && (FrameList[UserFrameIndex].length != 0)
        && (ByteNum >= FrameList[UserFrameIndex].startingByteNum)
        && ((FrameList[UserFrameIndex].length+5) >= BytesLeftInCodeword) )
      {
        /* then start a new frame */
        FrameTag = "  Start Frame";
        ByteToSend = START_OF_FRAME_BYTE;
        FrameBytesToGo = FrameList[UserFrameIndex].length + 4;
        FrameBytePointer = FrameList[UserFrameIndex].theBytes;
        UserFrameIndex++;
        EfmCrcReset();
      } else if(FrameBytesToGo) {
        /* else if inside a frame then handle outputting a data byte (or the crc to go with it)
*/
        switch(FrameBytesToGo) {
        case 4: /* send first crc byte */
          FrameTag = "    First Crc Byte";
          ByteToSend = 0xff & ~CrcAccum;
          break;
        case 3:
          ByteToSend = 0xff & ~(CrcAccum >> 8);
          break;
        case 2:
```

```
            ByteToSend = 0xff & ~(CrcAccum >> 16);
            break;
          case 1: /* send last crc byte */
            FrameTag = "    Last Crc Byte";
            ByteToSend = 0xff & ~(CrcAccum >> 24);
            /* if crc ends just before sync byte, prepare to send C(0) byte */
            if (ByteNum % CODEWORD_BYTE_COUNT == 64) {
              NeedCZero = 1;
            }
            break;
          default: /* just send next data byte and update crc */
            ByteToSend = (unsigned char)*FrameBytePointer++;
            /* scramble only the payload data (61.2.3.3.1) */
            ByteToSend = EfmScrambler( ByteToSend );
            /* calculate CRC on scrambled data (61.2.3.3) */
            EfmCrc(ByteToSend);
            break;
          }
          FrameBytesToGo--;
        } else {
          /* else just output an idle byte */
          ByteToSend = IDLE_BYTE;
        }
      }

      /* output the byte (msb on left) (transmission order is right to left)*/
      printf("%05.5d: %02.2X ", ByteNum, ByteToSend);
      for(b=0x80; b; b = b >> 1) {
        if(ByteToSend & b) {
          printf("1");
        } else {
          printf("0");
        }
      }
      if((argc > 1) && FrameTag) {
        printf("    ;%s", FrameTag);
      }
      printf("\n");
    }

    return(0);
}
```

```
The logfile contains a valid EFM bitstream reading right to left and
then top to bottom.
```

00000: F0 11110000 ;EFM bitstream reading right to left.
00001: 00 00000000
00002: 00 00000000
00003: 00 00000000
00004: 00 00000000
00005: 00 00000000
00006: 00 00000000
00007: 00 00000000
00008: 00 00000000
00009: 00 00000000
00010: 00 00000000
00011: 00 00000000
00012: 00 00000000
00013: 00 00000000
00014: 00 00000000
00015: 00 00000000
00016: 00 00000000
00017: 00 00000000
00018: 00 00000000
00019: 00 00000000
00020: 00 00000000
00021: 00 00000000
00022: 00 00000000
00023: 00 00000000
00024: 00 00000000
00025: 00 00000000
00026: 00 00000000
00027: 00 00000000
00028: 00 00000000
00029: 00 00000000
00030: 00 00000000
00031: 00 00000000
00032: 00 00000000
00033: 00 00000000
00034: 00 00000000
00035: 00 00000000
00036: 00 00000000
00037: 00 00000000
00038: 00 00000000
00039: 00 00000000
00040: 00 00000000
00041: 00 00000000
00042: 00 00000000
00043: 00 00000000

00044: 00 00000000
00045: 00 00000000
00046: 00 00000000
00047: 00 00000000
00048: 00 00000000
00049: 00 00000000
00050: 00 00000000
00051: 00 00000000
00052: 00 00000000
00053: 00 00000000
00054: 00 00000000
00055: 00 00000000
00056: 00 00000000
00057: 00 00000000
00058: 00 00000000
00059: 00 00000000
00060: 00 00000000
00061: 00 00000000
00062: 00 00000000
00063: 00 00000000
00064: 00 00000000
00065: F0 11110000 ;CODEWORD START (not all data)
00066: 00 00000000
00067: 00 00000000
00068: 00 00000000
00069: 00 00000000
00070: 00 00000000
00071: 00 00000000
00072: 00 00000000
00073: 00 00000000
00074: 00 00000000
00075: 00 00000000
00076: 00 00000000
00077: 00 00000000
00078: 00 00000000
00079: 00 00000000
00080: 00 00000000
00081: 00 00000000
00082: 00 00000000
00083: 00 00000000
00084: 00 00000000
00085: 00 00000000
00086: 00 00000000
00087: 00 00000000
00088: 00 00000000
00089: 00 00000000

00090: 00 00000000
00091: 00 00000000
00092: 00 00000000
00093: 00 00000000
00094: 00 00000000
00095: 00 00000000
00096: 00 00000000
00097: 00 00000000
00098: 00 00000000
00099: 00 00000000
00100: 00 00000000
00101: 00 00000000
00102: 00 00000000
00103: 00 00000000
00104: 00 00000000
00105: 00 00000000
00106: 00 00000000
00107: 00 00000000
00108: 00 00000000
00109: 00 00000000
00110: 00 00000000
00111: 00 00000000
00112: 00 00000000
00113: 00 00000000
00114: 00 00000000
00115: 00 00000000
00116: 00 00000000
00117: 00 00000000
00118: 00 00000000
00119: 00 00000000
00120: 00 00000000
00121: 00 00000000
00122: 00 00000000
00123: 00 00000000
00124: 00 00000000
00125: 00 00000000
00126: 00 00000000
00127: 00 00000000
00128: 00 00000000
00129: 00 00000000
00130: F0 11110000 ;CODEWORD START (not all data)
00131: 00 00000000
00132: 00 00000000
00133: 00 00000000
00134: 00 00000000
00135: 00 00000000

00136: 00 00000000
00137: 00 00000000
00138: 00 00000000
00139: 00 00000000
00140: 00 00000000
00141: 00 00000000
00142: 00 00000000
00143: 00 00000000
00144: 00 00000000
00145: 00 00000000
00146: 00 00000000
00147: 00 00000000
00148: 00 00000000
00149: 00 00000000
00150: 00 00000000
00151: 00 00000000
00152: 00 00000000
00153: 00 00000000
00154: 00 00000000
00155: 00 00000000
00156: 00 00000000
00157: 00 00000000
00158: 00 00000000
00159: 00 00000000
00160: 00 00000000
00161: 00 00000000
00162: 00 00000000
00163: 00 00000000
00164: 00 00000000
00165: 00 00000000
00166: 00 00000000
00167: 00 00000000
00168: 00 00000000
00169: 00 00000000
00170: 00 00000000
00171: 00 00000000
00172: 00 00000000
00173: 00 00000000
00174: 00 00000000
00175: 00 00000000
00176: 00 00000000
00177: 00 00000000
00178: 00 00000000
00179: 00 00000000
00180: 00 00000000
00181: 00 00000000

00182: 00 00000000
00183: 00 00000000
00184: 00 00000000
00185: 00 00000000
00186: 00 00000000
00187: 00 00000000
00188: 00 00000000
00189: 00 00000000
00190: 00 00000000
00191: 00 00000000
00192: 00 00000000
00193: 00 00000000
00194: 00 00000000
00195: F0 11110000 ;CODEWORD START (not all data)
00196: 00 00000000
00197: 00 00000000
00198: 00 00000000
00199: 00 00000000
00200: 50 01010000 ; Start Frame
00201: 00 00000000
00202: 01 00000001
00203: 02 00000010
00204: 87 10000111
00205: 0C 00001100
00206: 98 10011000
00207: 77 01110111
00208: 61 01100001
00209: 1A 00011010
00210: 37 00110111
00211: 53 01010011
00212: 5A 01011010
00213: DB 11011011
00214: 4D 01001101
00215: CE 11001110
00216: D5 11010101
00217: 0F 00001111
00218: A1 10100001
00219: C7 11000111
00220: 10 00010000
00221: DA 11011010
00222: 35 00110101
00223: 76 01110110
00224: 2D 00101101
00225: DA 11011010
00226: 17 00010111
00227: 64 01100100

```
00228: A9 10101001
00229: 87 10000111
00230: 0A 00001010
00231: D4 11010100
00232: 76 01110110
00233: 75 01110101
00234: 90 10010000
00235: 4C 01001100
00236: 58 01011000
00237: 5E 01011110
00238: 62 01100010
00239: 73 01110011
00240: 81 10000001
00241: 54 01010100
00242: 95 10010101
00243: 38 00111000
00244: D4 11010100
00245: 84 10000100
00246: 61 01100001
00247: 57 01010111
00248: 6B 01101011
00249: DD 11011101
00250: 37 00110111
00251: F2 11110010
00252: 02 00000010
00253: E7 11100111
00254: 47 01000111
00255: 2B 00101011
00256: DB 11011011
00257: 36 00110110
00258: C0 11000000
00259: 8C 10001100
00260: F0 11110000   ;CODEWORD START (not all data)
00261: 99 10011001   ;   C(9)
00262: 20 00100000
00263: 6F 01101111
00264: F9 11111001
00265: 12 00010010
00266: 6D 01101101
00267: E4 11100100   ;   First Crc Byte
00268: 3D 00111101
00269: BB 10111011
00270: BB 10111011   ;   Last Crc Byte
00271: 00 00000000
00272: 00 00000000
00273: 00 00000000
```

00274: 00 00000000
00275: 00 00000000
00276: 00 00000000
00277: 00 00000000
00278: 00 00000000
00279: 00 00000000
00280: 00 00000000
00281: 00 00000000
00282: 00 00000000
00283: 00 00000000
00284: 00 00000000
00285: 00 00000000
00286: 00 00000000
00287: 00 00000000
00288: 00 00000000
00289: 00 00000000
00290: 00 00000000
00291: 00 00000000
00292: 00 00000000
00293: 00 00000000
00294: 00 00000000
00295: 00 00000000
00296: 00 00000000
00297: 00 00000000
00298: 00 00000000
00299: 00 00000000
00300: 00 00000000
00301: 00 00000000
00302: 00 00000000
00303: 00 00000000
00304: 00 00000000
00305: 00 00000000
00306: 00 00000000
00307: 00 00000000
00308: 00 00000000
00309: 00 00000000
00310: 00 00000000
00311: 00 00000000
00312: 00 00000000
00313: 00 00000000
00314: 00 00000000
00315: 00 00000000
00316: 00 00000000
00317: 00 00000000
00318: 00 00000000
00319: 00 00000000

00320: 00 00000000
00321: 00 00000000
00322: 00 00000000
00323: 00 00000000
00324: 00 00000000
00325: F0 11110000 ;CODEWORD START (not all data)
00326: 00 00000000
00327: 00 00000000
00328: 00 00000000
00329: 00 00000000
00330: 00 00000000
00331: 00 00000000
00332: 00 00000000
00333: 00 00000000
00334: 00 00000000
00335: 00 00000000
00336: 00 00000000
00337: 00 00000000
00338: 00 00000000
00339: 00 00000000
00340: 00 00000000
00341: 00 00000000
00342: 00 00000000
00343: 00 00000000
00344: 00 00000000
00345: 00 00000000
00346: 00 00000000
00347: 00 00000000
00348: 00 00000000
00349: 00 00000000
00350: 00 00000000
00351: 00 00000000
00352: 00 00000000
00353: 00 00000000
00354: 00 00000000
00355: 00 00000000
00356: 00 00000000
00357: 00 00000000
00358: 00 00000000
00359: 00 00000000
00360: 00 00000000
00361: 00 00000000
00362: 00 00000000
00363: 00 00000000
00364: 00 00000000
00365: 00 00000000

00366: 00 00000000
00367: 00 00000000
00368: 00 00000000
00369: 00 00000000
00370: 00 00000000
00371: 00 00000000
00372: 00 00000000
00373: 00 00000000
00374: 00 00000000
00375: 00 00000000
00376: 00 00000000
00377: 00 00000000
00378: 00 00000000
00379: 00 00000000
00380: 00 00000000
00381: 00 00000000
00382: 00 00000000
00383: 00 00000000
00384: 00 00000000
00385: 00 00000000
00386: 00 00000000
00387: 00 00000000
00388: 00 00000000
00389: 50 01010000 ; Start Frame
00390: 0F 00001111 ;CODEWORD START (all data)
00391: 37 00110111
00392: 3C 00111100
00393: 69 01101001
00394: E8 11101000
00395: 3E 00111110
00396: 90 10010000
00397: 89 10001001
00398: 58 01011000
00399: E6 11100110
00400: 2F 00101111
00401: BF 10111111
00402: 47 01000111
00403: 67 01100111
00404: CC 11001100
00405: 30 00110000
00406: 0D 00001101
00407: B5 10110101
00408: BD 10111101
00409: 40 01000000
00410: 3F 00111111
00411: 48 01001000

00412: 48 01001000
00413: 29 00101001
00414: 12 00010010
00415: 19 00011001
00416: 45 01000101
00417: F7 11110111
00418: 83 10000011
00419: 63 01100011
00420: E9 11101001
00421: 51 01010001
00422: 0B 00001011
00423: 93 10010011
00424: A4 10100100
00425: EB 11101011
00426: F8 11111000
00427: 58 01011000
00428: B3 10110011
00429: 39 00111001
00430: 46 01000110
00431: 17 00010111
00432: 2D 00101101
00433: D4 11010100
00434: 14 00010100
00435: 6A 01101010
00436: 14 00010100
00437: 8C 10001100
00438: 4B 01001011
00439: 0A 00001010
00440: D9 11011001
00441: 3E 00111110
00442: D2 11010010
00443: A3 10100011
00444: 62 01100010
00445: 50 01010000
00446: EC 11101100
00447: 48 01001000
00448: A0 10100000
00449: 6F 01101111
00450: 9E 10011110
00451: 52 01010010
00452: 73 01110011
00453: 3B 00111011
00454: 5B 01011011
00455: F0 11110000 ;CODEWORD START (not all data)
00456: 14 00010100 ; C(4)
00457: 5D 01011101 ; First Crc Byte

00458: 04 00000100
00459: CB 11001011
00460: F7 11110111 ; Last Crc Byte
00461: 00 00000000
00462: 00 00000000
00463: 00 00000000
00464: 00 00000000
00465: 50 01010000 ; Start Frame
00466: 14 00010100
00467: B0 10110000
00468: 3E 00111110
00469: 89 10001001
00470: E6 11100110
00471: FE 11111110
00472: 98 10011000
00473: CF 11001111
00474: 54 01010100
00475: BB 10111011
00476: 7E 01111110
00477: 0C 00001100
00478: EB 11101011
00479: 43 01000011
00480: 64 01100100
00481: B5 10110101
00482: E0 11100000
00483: 36 00110110
00484: 8A 10001010
00485: F8 11111000
00486: 67 01100111
00487: F2 11110010
00488: 35 00110101
00489: AD 10101101
00490: 76 01110110
00491: 77 01110111
00492: D6 11010110
00493: 3D 00111101
00494: 3E 00111110
00495: 41 01000001
00496: B8 10111000
00497: C4 11000100
00498: A1 10100001
00499: 2F 00101111
00500: 07 00000111
00501: 0D 00001101
00502: EF 11101111
00503: D2 11010010

00504: 5C 01011100
00505: 5B 01011011
00506: 72 01110010
00507: AA 10101010
00508: 8E 10001110
00509: FB 11111011
00510: 03 00000011
00511: 44 01000100
00512: 9C 10011100
00513: 7E 01111110
00514: 23 00100011
00515: C5 11000101
00516: 14 00010100 ; First Crc Byte
00517: B0 10110000
00518: 67 01100111
00519: 49 01001001 ; Last Crc Byte
00520: F0 11110000 ;CODEWORD START (not all data)
00521: 90 10010000 ; C(0)
00522: 00 00000000
00523: 00 00000000
00524: 00 00000000
00525: 00 00000000
00526: 00 00000000
00527: 00 00000000
00528: 00 00000000
00529: 00 00000000
00530: 00 00000000
00531: 00 00000000
00532: 00 00000000
00533: 00 00000000
00534: 00 00000000
00535: 00 00000000
00536: 00 00000000
00537: 00 00000000
00538: 00 00000000
00539: 00 00000000
00540: 00 00000000
00541: 00 00000000
00542: 00 00000000
00543: 00 00000000
00544: 00 00000000
00545: 00 00000000
00546: 00 00000000
00547: 00 00000000
00548: 00 00000000
00549: 00 00000000

```
00550: 00 00000000
00551: 00 00000000
00552: 00 00000000
00553: 00 00000000
00554: 00 00000000
00555: 00 00000000
00556: 00 00000000
00557: 00 00000000
00558: 00 00000000
00559: 00 00000000
00560: 00 00000000
00561: 00 00000000
00562: 00 00000000
00563: 00 00000000
00564: 00 00000000
00565: 00 00000000
00566: 00 00000000
00567: 00 00000000
00568: 00 00000000
00569: 00 00000000
00570: 00 00000000
00571: 00 00000000
00572: 00 00000000
00573: 00 00000000
00574: 00 00000000
00575: 00 00000000
00576: 00 00000000
00577: 50 01010000    ;  Start Frame
00578: 57 01010111
00579: C6 11000110
00580: 9C 10011100
00581: 48 01001000    ;    First Crc Byte
00582: 64 01100100
00583: A3 10100011
00584: 11 00010001    ;    Last Crc Byte
00585: F0 11110000    ;CODEWORD START (not all data)
00586: 90 10010000    ;    C(0)
00587: 00 00000000
00588: 00 00000000
00589: 00 00000000
00590: 00 00000000
00591: 00 00000000
00592: 00 00000000
00593: 00 00000000
00594: 00 00000000
00595: 00 00000000
```

00596: 00 00000000
00597: 00 00000000
00598: 00 00000000
00599: 00 00000000
00600: 00 00000000
00601: 00 00000000
00602: 00 00000000
00603: 00 00000000
00604: 00 00000000
00605: 00 00000000
00606: 00 00000000
00607: 00 00000000
00608: 00 00000000
00609: 00 00000000
00610: 00 00000000
00611: 00 00000000
00612: 00 00000000
00613: 00 00000000
00614: 00 00000000
00615: 00 00000000
00616: 00 00000000
00617: 00 00000000
00618: 00 00000000
00619: 00 00000000
00620: 00 00000000
00621: 00 00000000
00622: 00 00000000
00623: 00 00000000
00624: 00 00000000
00625: 00 00000000
00626: 00 00000000
00627: 00 00000000
00628: 00 00000000
00629: 00 00000000
00630: 00 00000000
00631: 00 00000000
00632: 00 00000000
00633: 00 00000000
00634: 00 00000000
00635: 00 00000000
00636: 00 00000000
00637: 00 00000000
00638: 00 00000000
00639: 00 00000000
00640: 00 00000000
00641: 00 00000000

```
00642: 00 00000000
00643: 00 00000000
00644: 00 00000000
00645: 00 00000000
00646: 00 00000000
00647: 00 00000000
00648: 00 00000000
00649: 00 00000000
00650: F0 11110000    ;CODEWORD START (not all data)
00651: 50 01010000    ;  Start Frame
00652: 32 00110010
00653: 11 00010001
00654: 86 10000110
00655: DE 11011110
00656: 14 00010100
00657: 3C 00111100
00658: 3A 00111010
00659: FD 11111101
00660: FE 11111110
00661: 60 01100000
00662: 8F 10001111
00663: F7 11110111
00664: 81 10000001
00665: 14 00010100
00666: F2 11110010
00667: 1D 00011101
00668: D2 11010010
00669: 9F 10011111
00670: 54 01010100
00671: 85 10000101
00672: 09 00001001
00673: AA 10101010
00674: F2 11110010
00675: BB 10111011
00676: 87 10000111
00677: 0F 00001111
00678: D9 11011001
00679: E6 11100110
00680: FF 11111111
00681: EA 11101010
00682: 12 00010010
00683: CB 11001011
00684: 1E 00011110
00685: 84 10000100
00686: 3C 00111100
00687: 3C 00111100
```

```
00688: 94 10010100
00689: CB 11001011
00690: 68 01101000
00691: C3 11000011
00692: EE 11101110
00693: 90 10010000
00694: F0 11110000
00695: 1F 00011111
00696: A6 10100110
00697: AA 10101010
00698: B9 10111001
00699: D6 11010110
00700: 03 00000011
00701: 37 00110111
00702: D6 11010110
00703: 6E 01101110
00704: 77 01110111
00705: E5 11100101
00706: 5C 01011100
00707: 19 00011001
00708: 39 00111001
00709: F2 11110010
00710: 52 01010010
00711: EF 11101111
00712: 0E 00001110
00713: 29 00101001
00714: 72 01110010
00715: F0 11110000    ;CODEWORD START (not all data)
00716: 95 10010101    ;    C(5)
00717: 1C 00011100
00718: 4B 01001011    ;    First Crc Byte
00719: F7 11110111
00720: F8 11111000
00721: E3 11100011    ;    Last Crc Byte
00722: 50 01010000    ;  Start Frame
00723: 9C 10011100
00724: 09 00001001
00725: 3C 00111100
00726: AB 10101011
00727: B0 10110000
00728: 77 01110111
00729: D1 11010001
00730: 41 01000001
00731: B6 10110110
00732: A6 10100110
00733: B3 10110011
```

00734: 8A 10001010
00735: 51 01010001
00736: 3E 00111110
00737: CD 11001101
00738: 9E 10011110
00739: FB 11111011
00740: 4C 01001100
00741: 73 01110011
00742: 1D 00011101
00743: 3F 00111111
00744: 99 10011001
00745: 24 00100100
00746: AC 10101100
00747: 86 10000110
00748: FB 11111011
00749: 16 00010110
00750: 76 01110110
00751: 7A 01111010
00752: 8E 10001110
00753: 8C 10001100
00754: 5B 01011011
00755: 15 00010101
00756: C9 11001001
00757: 9A 10011010
00758: CD 11001101
00759: 6B 01101011
00760: 9E 10011110
00761: 2F 00101111
00762: 2B 00101011
00763: 19 00011001
00764: 52 01010010
00765: 9B 10011011
00766: 2F 00101111
00767: A8 10101000
00768: 1E 00011110
00769: D9 11011001
00770: 41 01000001
00771: 9B 10011011
00772: 9A 10011010
00773: BF 10111111
00774: 54 01010100
00775: 47 01000111
00776: 78 01111000
00777: C1 11000001
00778: 5F 01011111 ; First Crc Byte
00779: 18 00011000

```
00780: F0 11110000    ;CODEWORD START (not all data)
00781: 12 00010010    ;    C(2)
00782: BF 10111111
00783: BF 10111111    ;    Last Crc Byte
00784: 00 00000000
00785: 00 00000000
00786: 00 00000000
00787: 00 00000000
00788: 00 00000000
00789: 00 00000000
00790: 00 00000000
00791: 00 00000000
00792: 00 00000000
00793: 00 00000000
00794: 00 00000000
00795: 00 00000000
00796: 00 00000000
00797: 00 00000000
00798: 00 00000000
00799: 00 00000000
00800: 00 00000000
00801: 00 00000000
00802: 00 00000000
00803: 00 00000000
00804: 00 00000000
00805: 00 00000000
00806: 00 00000000
00807: 00 00000000
00808: 00 00000000
00809: 00 00000000
00810: 00 00000000
00811: 00 00000000
00812: 00 00000000
00813: 00 00000000
00814: 00 00000000
00815: 00 00000000
00816: 00 00000000
00817: 00 00000000
00818: 00 00000000
00819: 00 00000000
00820: 00 00000000
00821: 00 00000000
00822: 00 00000000
00823: 00 00000000
00824: 00 00000000
00825: 00 00000000
```

00826: 00 00000000
00827: 00 00000000
00828: 00 00000000
00829: 00 00000000
00830: 00 00000000
00831: 00 00000000
00832: 00 00000000
00833: 00 00000000
00834: 00 00000000
00835: 00 00000000
00836: 00 00000000
00837: 00 00000000
00838: 00 00000000
00839: 00 00000000
00840: 00 00000000
00841: 00 00000000
00842: 00 00000000
00843: 00 00000000
00844: 00 00000000
00845: F0 11110000 ;CODEWORD START (not all data)
00846: 00 00000000
00847: 00 00000000
00848: 00 00000000
00849: 00 00000000
00850: 00 00000000
00851: 00 00000000
00852: 00 00000000
00853: 00 00000000
00854: 00 00000000
00855: 00 00000000
00856: 00 00000000
00857: 00 00000000
00858: 00 00000000
00859: 00 00000000
00860: 00 00000000
00861: 00 00000000
00862: 00 00000000
00863: 00 00000000
00864: 00 00000000
00865: 00 00000000
00866: 00 00000000
00867: 00 00000000
00868: 00 00000000
00869: 00 00000000
00870: 00 00000000
00871: 00 00000000

00872: 00 00000000
00873: 00 00000000
00874: 00 00000000
00875: 00 00000000
00876: 00 00000000
00877: 00 00000000
00878: 00 00000000
00879: 00 00000000
00880: 00 00000000
00881: 00 00000000
00882: 00 00000000
00883: 00 00000000
00884: 00 00000000
00885: 00 00000000
00886: 00 00000000
00887: 00 00000000
00888: 00 00000000
00889: 00 00000000
00890: 00 00000000
00891: 00 00000000
00892: 00 00000000
00893: 00 00000000
00894: 00 00000000
00895: 00 00000000
00896: 00 00000000
00897: 00 00000000
00898: 00 00000000
00899: 00 00000000
00900: 00 00000000
00901: 00 00000000
00902: 00 00000000
00903: 00 00000000
00904: 00 00000000
00905: 00 00000000
00906: 00 00000000
00907: 00 00000000
00908: 00 00000000
00909: 00 00000000