Specification for DMR SMS/Text Messages – Revision 1.0

Summary
This document will cover the message specification for unconfirmed DMR Text messages. At this revision, formats for Motorola and DMR_Standard (identified as such in in Anytone radios) will be covered.

Overview
Unconfirmed DMR Text messages are transmitted as IP UDP data packets that are fragmented into HalfRate DMR PDUs. The typical sequence that is seen are multiple CSBK packets that count down the remaining packets in the message, followed by a DMR unconfirmed data header and multiple HalfRate data packets.

Example

CSBK #1 - 10 packets remaining
CSBK #2 - 9 packets remaining
CSBK #3 - 8 packets remaining
CSBK #4 - 7 packets remaining
CSBK #5 - 6 packets remaining
UnconfirmedDataHeader – 5 Data Fragments
HalfRate #1 – 12 bytes of data
HalfRate #2 – 12 bytes of data
HalfRate #3 – 12 bytes of data
HalfRate #4 – 12 bytes of data
HalfRate #5 – 12 bytes of data – Last 4 bytes contains CRC32 and 0 padding if needed

Motorola Compatible Format

The text message is transmitted in the form of an IP UDP data packet broken into 12 byte fragments of half rate data. The IP layer is formatted as follows.

Offset	Length	Description
0	1	0x45 – const IPv4 protocol identifier
1	1	0x00 – const IP Type of Service
2	2	IP Packet Length BE (big-endian) – not counting padding or final CRC32
4	2	IP Sequence Number BE, increment for each new data packet
6	2	0x0000 – const IP flags and fragment offset (not used)
8	1	0x01 – const IP Time to live
9	1	0x11 – const UDP protocol identifier
10	2	IP Header Checksum – BE from RFC 791 (See CRC Appendix)
12	4	Source IP contains the DMR Id of the sender of the message. The first byte is 0x0C for user DMR Id, and the remaining 3 bytes are the DMR Id in BE order. Ex. 12.0.1.1 for DMR Id 257.
16	4	Dest IP Contains either a user or a group id. The first byte of a group

		address is 0xE1 and the remaining 3 are Group Id in BE. Ex. 225.0.0.1 is group Id 1. User Ids are prefixed with 0x0C.
20	2	UDP Source Port – const 0x0FA7 (4007) for Motorola Compatible
22	2	UDP Dest Port – const 0x0FA7 (4007) for Motorola Compatible
24	2	UDP Data Length – double byte message length + 18. Ex. “HELLO” would be 5*2+18=28.
26	2	UDP Checksum - BE from RFC 768 (See CRC Appendix)
28	1	const 0x00 – This is the start of the UDP Data Message.
29	1	Internal Length – double byte message length + 8. Ex. “HELLO” would be 5*2+8=18.
30	2	const 0xA0
31	1	const 0x00
32	1	Internal sequence – use IP seq num | 0x80
33	1	const 0x04
34	1	const 0x0D
35	1	const 0x00
36	1	const 0x0A
37	1	const 0x00
38	Text Len*2	Text Message. Ex. “HELLO” is 0x480045004C004C004F00

DMR_Standard Compatible Format

The DMR_Standard format is also a UDP IP packet with a different port and less header in the payload.

Offset	Length	Description
0	20	Standard IP Header (same as Motorola format above)
20	2	UDP Source Port – const 0x1398 (5016) for DMR_Standard
22	2	UDP Dest Port – const 0x1398 (5016) for DMR_Standard
24	2	UDP Data Length – double byte message length + 12. Ex. “HELLO” would be 5*2+12=22.
26	2	UDP Checksum - BE from RFC 768 (See CRC Appendix)
28	1	const 0x00 – This is the start of the UDP Data Message.
29	1	const 0x0D
30	2	const 0x00
31	1	const 0x0A
32	Text Len*2	Text Message. Ex. “HELLO” is 0x480045004C004C004F00

IP Fragmentation and Packaging

The total buffer size must be a multiple or 12 consisting of IP Fragments + Zero_Padding + 4_byte_CRC32. If there are four bytes free in the final fragment then fill in the remaining space with 0’s and put the CRC32 in the last 4 bytes of the last fragment. The number of padding bytes is included in the Unconfirmed Data Header.

Appendix

IP Header Checksum

This is a code fragment that outlines the IP and UDP checksum.

```
//IP checksum RFC 791
public static byte[] ipHeaderChecksum(byte[] buf, int length) {
   int i = 0;
   long sum = 0;
   while (length > 0) {
      sum += (buf[i++]&0xff) << 8;
      if ((--length)==0) break;
      sum += (buf[i++]&0xff);
      --length;
   }
   byte[] ret = new byte[2];
   int cs = (int) (~((sum & 0xFFFF)+(sum >> 16)))&0xFFFF;
   ret[0]=(byte) (cs>>8);
   ret[1]=(byte) (cs);
   return ret ;
}
```

The IP header checksum is computed by setting the check sum values in position 10-11 to zero and passing the first 20 bytes of the buffer to the code sample above.

UDP Checksum

The UDP checksum is computed with the UDP payload, header and a 12 byte pseudo header. The most optimal way to handle this would be to fill the pseudo header values in the main packet buffer, compute the checksum and then correct with the proper IP format values.

```
//add UDP checksum
byte[] b = new byte[udpLength+12];
System.arraycopy(ipPacket,12, b,0, 4); //ip src
System.arraycopy(ipPacket,16, b,4, 4); //ip dst
b[8]=0;
b[9]=0x11; //udp protocol id
b[10] = (byte) (udpLength>>8);
b[11] = (byte) (udpLength);
System.arraycopy(ipPacket,20, b,12, udpLength); //UDP packet
cs = CRC.ipHeaderChecksum(b,b.length) ;
ipPacket[26]=cs[0];
ipPacket[27]=cs[1];
```

Unconfirmed Data CRC32

The ESTI DMR CRC32 is calculated by sending the buffer containing the IP Packet + any Zero_padding. Don't send the last 4 bytes. Fill in the return CRC value into the last four bytes.

```
public static byte[] itu_crc32(byte[] in, int length) {
        byte[] bar = new byte[in.length];
        System.arraycopy(in,0,bar,0,in.length) ;
        flip(bar);

        int val = digital_update_crc32(0, bar, length);

        byte[] ret = new byte[4];
        ret[3] = (byte) ((val>>24)&0xFF) ;
        ret[2] = (byte) ((val>>16)&0xFF) ;
        ret[1] = (byte) ((val>>8)&0xFF) ;
        ret[0] = (byte) (val&0xFF) ;
        return ret ;
}

public static void flip(byte [] bar) {
        for(int i=0; i<bar.length; i+=2 ){
                byte b = bar[i] ;
                bar[i] = bar[i+1];
                bar[i+1]=b;
        }
}

public static int digital_update_crc32(int crc_initial, byte[] data, int length) {
        int crc = crc_initial;
        int pos=0;
   for(int i=0;i<length;i++)
   {
    crc = table32[((int)data[i]&0xFF) ^ ((crc >> 24) & 0xff)] ^ (crc << 8);
   }
   return crc;
}


public static  int[] table32 = {
   0x00000000,0x04C11DB7,0x09823B6E,0x0D4326D9,
   0x130476DC,0x17C56B6B,0x1A864DB2,0x1E475005,
   0x2608EDB8,0x22C9F00F,0x2F8AD6D6,0x2B4BCB61,
   0x350C9B64,0x31CD86D3,0x3C8EA00A,0x384FBDBD,
   0x4C11DB70,0x48D0C6C7,0x4593E01E,0x4152FDA9,
   0x5F15ADAC,0x5BD4B01B,0x569796C2,0x52568B75,
   0x6A1936C8,0x6ED82B7F,0x639B0DA6,0x675A1011,
   0x791D4014,0x7DDC5DA3,0x709F7B7A,0x745E66CD,
```

0x9823B6E0,0x9CE2AB57,0x91A18D8E,0x95609039,
0x8B27C03C,0x8FE6DD8B,0x82A5FB52,0x8664E6E5,
0xBE2B5B58,0xBAEA46EF,0xB7A96036,0xB3687D81,
0xAD2F2D84,0xA9EE3033,0xA4AD16EA,0xA06C0B5D,
0xD4326D90,0xD0F37027,0xDDB056FE,0xD9714B49,
0xC7361B4C,0xC3F706FB,0xCEB42022,0xCA753D95,
0xF23A8028,0xF6FB9D9F,0xFBB8BB46,0xFF79A6F1,
0xE13EF6F4,0xE5FFEB43,0xE8BCCD9A,0xEC7DD02D,
0x34867077,0x30476DC0,0x3D044B19,0x39C556AE,
0x278206AB,0x23431B1C,0x2E003DC5,0x2AC12072,
0x128E9DCF,0x164F8078,0x1B0CA6A1,0x1FCDBB16,
0x018AEB13,0x054BF6A4,0x0808D07D,0x0CC9CDCA,
0x7897AB07,0x7C56B6B0,0x71159069,0x75D48DDE,
0x6B93DDDB,0x6F52C06C,0x6211E6B5,0x66D0FB02,
0x5E9F46BF,0x5A5E5B08,0x571D7DD1,0x53DC6066,
0x4D9B3063,0x495A2DD4,0x44190B0D,0x40D816BA,
0xACA5C697,0xA864DB20,0xA527FDF9,0xA1E6E04E,
0xBFA1B04B,0xBB60ADFC,0xB6238B25,0xB2E29692,
0x8AAD2B2F,0x8E6C3698,0x832F1041,0x87EE0DF6,
0x99A95DF3,0x9D684044,0x902B669D,0x94EA7B2A,
0xE0B41DE7,0xE4750050,0xE9362689,0xEDF73B3E,
0xF3B06B3B,0xF771768C,0xFA325055,0xFEF34DE2,
0xC6BCF05F,0xC27DEDE8,0xCF3ECB31,0xCBFFD686,
0xD5B88683,0xD1799B34,0xDC3ABDED,0xD8FBA05A,
0x690CE0EE,0x6DCDFD59,0x608EDB80,0x644FC637,
0x7A089632,0x7EC98B85,0x738AAD5C,0x774BB0EB,
0x4F040D56,0x4BC510E1,0x46863638,0x42472B8F,
0x5C007B8A,0x58C1663D,0x558240E4,0x51435D53,
0x251D3B9E,0x21DC2629,0x2C9F00F0,0x285E1D47,
0x36194D42,0x32D850F5,0x3F9B762C,0x3B5A6B9B,
0x0315D626,0x07D4CB91,0x0A97ED48,0x0E56F0FF,
0x1011A0FA,0x14D0BD4D,0x19939B94,0x1D528623,
0xF12F560E,0xF5EE4BB9,0xF8AD6D60,0xFC6C70D7,
0xE22B20D2,0xE6EA3D65,0xEBA91BBC,0xEF68060B,
0xD727BBB6,0xD3E6A601,0xDEA580D8,0xDA649D6F,
0xC423CD6A,0xC0E2D0DD,0xCDA1F604,0xC960EBB3,
0xBD3E8D7E,0xB9FF90C9,0xB4BCB610,0xB07DABA7,
0xAE3AFBA2,0xAAFBE615,0xA7B8C0CC,0xA379DD7B,
0x9B3660C6,0x9FF77D71,0x92B45BA8,0x9675461F,
0x8832161A,0x8CF30BAD,0x81B02D74,0x857130C3,
0x5D8A9099,0x594B8D2E,0x5408ABF7,0x50C9B640,
0x4E8EE645,0x4A4FFBF2,0x470CDD2B,0x43CDC09C,
0x7B827D21,0x7F436096,0x7200464F,0x76C15BF8,
0x68860BFD,0x6C47164A,0x61043093,0x65C52D24,
0x119B4BE9,0x155A565E,0x18197087,0x1CD86D30,
0x029F3D35,0x065E2082,0x0B1D065B,0x0FDC1BEC,
0x3793A651,0x3352BBE6,0x3E119D3F,0x3AD08088,

```
0x2497D08D,0x2056CD3A,0x2D15EBE3,0x29D4F654,
0xC5A92679,0xC1683BCE,0xCC2B1D17,0xC8EA00A0,
0xD6AD50A5,0xD26C4D12,0xDF2F6BCB,0xDBEE767C,
0xE3A1CBC1,0xE760D676,0xEA23F0AF,0xEEE2ED18,
0xF0A5BD1D,0xF464A0AA,0xF9278673,0xFDE69BC4,
0x89B8FD09,0x8D79E0BE,0x803AC667,0x84FBDBD0,
0x9ABC8BD5,0x9E7D9662,0x933EB0BB,0x97FFAD0C,
0xAFB010B1,0xAB710D06,0xA6322BDF,0xA2F33668,
0xBCB4666D,0xB8757BDA,0xB5365D03,0xB1F740B4} ;
```