Error Detection & Correction of MPT1327 Formatted Messages using MX429A or MX809 devices

1.1 Background

MPT1327 messages are transmitted as 64-bit 'codewords', where each codeword contains 48 information bits followed by 16 check bits:

Bit No:	1 48	49 64
	information field	check bits

(Bit number 1 is transmitted first.)

These check bits allow the receive terminal to detect all odd numbers of errors, any 2 or 4 errors, and any error-burst up to length 16 in a codeword, and also to correct errors in the received codeword, although it should be noted that the higher the degree of error correction applied, the more likely is false decoding.

This document gives algorithms for:

Generation of the check bits of a transmitted codeword.
Received codeword error detection.
Limited error correction of a received codeword.

These algorithms may be used with any bit or byte oriented modem, such as the **MX429A** or MX809, although the **MX429A** and MX809 devices can perform check bit generation and error detection automatically and the **MX429A** also provides a 16-bit 'Syndrome' output which may be used to aid error correction.

1.2 Generation of Transmit Codeword Check Bits

1.2.1 Theory

The first 15 check bits are derived from a (63,48) cyclic code by using codeword bits 1 to 48 as the coefficients X62 to X15 (in that order) of a 63 bit polynomial, which is then divided modulo-2 by the generating polynomial;

$$X15 + X14 + X13 + X11 + X4 + X2 + X0 \qquad \text{(11101000 00010101 binary)}$$

On completion of the division, the 15 coefficients X14 to X0 of the remainder are used as the first 15 check bits (codeword bits 49 to 63), with the X0 coefficient (bit 63 of the complete codeword) inverted.

Finally, bit 64 of the codeword is added to provide an even parity check of the whole 64-bit codeword.

1.2.2 Example of Transmit Codeword Generation

Information field; 6 data bytes

89	AB	CD	EF	12	34	Hex
10001001	10101011	11001101	11101111	00010010	00110100	Binary

Polynomial division

```
x62…………………………………………………………………………………………………..x0
10001001 10101011 11001101 11101111 00010010 00110100 00000000 0000000
11101000 00010101
 1100001 10111110 1
 1110100 00001010 1
   10101 10110100 010
   11101 00000010 101
    1000 10110110 1110
    1110 10000001 0101
     110 00110111 10111
     111 01000000 10101
       1 01110111 0001010
       1 11010000 0010101
         10100111 00111111
         11101000 00010101
          1001111 00101010 1
          1110100 00001010 1
           111011 00100000 01
           111010 00000101 01
                1 00100101 0010111
                1 11010000 0010101
                  11110101 00000101
                  11101000 00010101
                     11101 00010000 000
                     11101 00000010 101
                     10010 10110010 001
                     11101 00000010 101
                      1111 10110000 1001
                      1110 10000001 0101
                         1 00110001 1100010
                         1 11010000 0010101
                           11100001 11101110
                           11101000 00010101
                               1001 11111011 0000
                               1110 10000001 0101
                                111 01111010 01010
                                111 01000000 10101
                                      111010 11111000 00
                                      111010 00000101 01
                                             11111101 0100000
```

Remainder with last bit inverted:

```
                                             11111101 0100001
```

Complete codeword, including parity bit:

```
Bit; 1                                                                  64
     10001001 10101011 11001101 11101111 00010010 00110100 11111101 01000010
         89       AB       CD       EF       12       34       FD       42
```

1.2.3 'C' Language Algorithm

```
/*******************************************************************************/
/*   Function gen_ckbits() returns the first 15 check bits of a transmit     */
/*   codeword (codeword bits 49 to 63). Bit 15 of the returned value will  */
/*   be codeword bit 49, bit 1 of the returned value will be codeword bit   */
/*   63, and the lsb (bit 0) should be ignored.                              */
/*   The last bit (64) of the codeword must be derived separately, to       */
/*   give even parity of the whole 64-bit codeword.                          */

gen_ckbits()
   {
   int n,bit;
   unsigned int ckbits = 0;          /* Clear check bits          */
   for(n=1;n <= 48;n++)              /* 48 information bits       */
      {                              /*                           */
      bit = getbit_tx(n);            /* Get each bit in turn      */
      if( 1 & (bit ^ (ckbits >> 15)))  /*   XOR tx bit with MSB    */
                                     /*   of checkbits and if     */
                                     /*   the result == 1         */
         ckbits ^= 0x6815;           /*   then XOR checkbits      */
                                     /*   with 6815 Hex           */
      ckbits <<= 1;                  /* ... Shift check bit word  */
                                     /*      one bit left,        */
      }
   return(ckbits ^ 0x0002);          /* Return checkbits with     */
                                     /* codeword bit 63 inverted */
   }

/*   Function getbit_tx(n) should return bit 'n' (1 to 48) of the transmit*/
/*   codeword information field.                               */

getbit_tx(n)
   {
   return(/* 1 or 0 */);
   }
```

1.3 Receive Codeword Checking & Error Correction

1.3.1 Theory

The parity of the received 64-bit codeword is checked, then bit 63 of the codeword is inverted. The first 63 bits of the resulting codeword are then used as the coefficients X77 to X15 of a 77 bit polynomial, which is then divided modulo-2 by the 'generating polynomial'. If the remainder is zero, and the parity check is met, then no errors have been detected.

The 15-bit remainder of this division is used as the least significant 15 bits of the 16-bit 'Syndrome' word generated by the MX429 (and by the algorithm of section 3.4), while the msb of the Syndrome word is set to '1' if the parity of the received codeword is incorrect. The resulting Syndrome word value can give an indication of which bit(s) of the codeword have been received incorrectly; see section 3.4.

1.3.2 Example of Receive Codeword Checking: No Errors

Received codeword: 6 bytes:

	89	AB	CD	EF	12	34	FD	42
	10001001	10101011	11001101	11101111	00010010	00110100	11111101	01000010
Bit;	1…………	…………	…………	…………	…………	…………	…………	…………64

Step 1: even parity checked OK

Step 2: invert bit 63 then divide first 63 bits (shifted left 15 places) by generating polynomial:

x^{77}……x^{0}

```
10001001 10101011 11001101 11101111 00010010 00110100 11111101 01000000 00000000 000000
11101000 00010101
 1100001 10111110 1
 1110100 00001010 1
   10101 10110100 010
   11101 00000010 101
    1000 10110110 1110
    1110 10000001 0101
     110 00110111 10111
     111 01000000 10101
       1 01110111 0001010
       1 11010000 0010101
         10100111 00111111
         11101000 00010101
          1001111 00101010 1
          1110100 00001010 1
           111011 00100000 01
           111010 00000101 01
                1 00100101 0010111
                1 11010000 0010101
                  11110101 00000101
                  11101000 00010101
                     11101 00010000 000
                     11101 00000010 101
                              10010 10110010 001
                              11101 00000010 101
                               1111 10110000 1001
                               1110 10000001 0101
                                  1 00110001 1100010
                                  1 11010000 0010101
                                    11100001 11101110
                                    11101000 00010101
                                        1001 11111011 1111
                                        1110 10000001 0101
                                         111 01111010 10101
                                         111 01000000 10101
                                              111010 00000101 01
                                              111010 00000101 01

                                              000000 00000000 00000000 00000000 000000
```

Remainder = zero
MX429 'Syndrome' word: 00000000 00000000
No errors detected

1.3.3 Example of Receive Codeword Checking: 2 Errors

Received codeword: 6 bytes: bits 9 & 10 in error

```
             89         6B         CD         EF         12         34         FD         42
          10001001   01101011   11001101   11101111   00010010   00110100   11111101   01000010
             errors;    xx
     Bit;  1                                                                                   64
```

Step 1: even parity checked OK

Step 2: invert bit 63 then divide first 63 bits (shifted left 15 places) by generating polynomial:

```
x^77 ...........................................................................................x^0
10001001 01101011 11001101 11101111 00010010 00110100 11111101 01000000 00000000 000000
11101000 00010101
 1100001 01111110 1
 1110100 00001010 1
   10101 01110100 010
   11101 00000010 101
    1000 01110110 1110
    1110 10000001 0101
     110 11110111 10111
     111 01000000 10101
       1 10110111 0001010
       1 11010000 0010101
          1100111 00111111 1
          1110100 00001010 1
            10011 00110101 011
            11101 00000010 101
             1110 00110111 1100
             1110 10000001 0101
                  10110110 10011111
                  11101000 00010101
                   1011110 10001010 0
                   1110100 00001010 1
                    101010 10000000 10
                    111010 00000101 01
                     10000 10000101 110
                     11101 00000010 101
                      1101 10000111 0111
                      1110 10000001 0101
                        11 00000110 001000
                        11 10100000 010101
                           10100110 01110110
                           11101000 00010101
                            1001110 01100011 0
                            1110100 00001010 1
                             111010 01101001 10
                             111010 00000101 01
                                    1101100 11110100 1
                                    1110100 00001010 1
                                      11000 11111110 011
                                      11101 00000010 101
                                        101 11111100 11011
                                        111 01000000 10101
                                         10 10111100 011101
                                         11 10100000 010101
                                          1 00011100 0010000
                                          1 11010000 0010101
                                            11001100 00001011
                                            11101000 00010101
                                              100100 00011110 01
                                              111010 00000101 01
                                               11110 00011011 000
                                               11101 00000010 101
                                                  11 00011001 101000
                                                  11 10100000 010101
```

```
10111001 11110100
11101000 00010101
 1010001 11100001 0
 1110100 00001010 1
  100101 11101011 10
  111010 00000101 01
   11111 11101110 110
   11101 00000010 101
      10 11101100 011000
      11 10100000 010101
       1 01001100 0011010
       1 11010000 0010101
         10011100 00011110
         11101000 00010101
          1110100 00001011 0
          1110100 00001010 1
```

Remainder; non zero

```
                       1 100000
```

MX429 'Syndrome' word: 00000000 01100000

Therefore, from the table in section 3.4, codeword bits 9 & 10 of the received codeword are incorrect.

1.3.4 'C' Language Algorithm

The following algorithm produces a 16-bit 'Syndrome' similar to that generated by the MX429, which will have a value of zero only if no errors have been detected in the received codeword.

```
/******************************************************************************************/
/*   Function calc_syndrome() returns the 16-bit 'Syndrome' of a received      */
/*   MPT1327 64-bit codeword.                                                  */

  calc_syndrome()
      {
      int n,bit;
      int parity=0;                                  /* Clear parity register       */
      int syndrome=0;                                /* Clear 16-bit syndrome       */
      for(n = 1;n <= 64;n++)                     /* 64-bit codeword...       */
          {                                          /*                             */
            bit = getbit_rx(n);                      /* Get each bit in turn;..     */
            parity ^= bit;                           /* .. update parity            */
            if(n == 63) bit ^= 1;                    /* .. then invert bit 63       */
            if(n < 64)                               /* .. for bits 1 to 63;....    */
              {                                      /*    .. shift parity word     */
              syndrome <<= 1;                        /*        one bit left.        */
               if( 1 & (bit ^ (syndrome >> 15)))     /*    .. XOR rx bit with       */
                                                     /*       MSB of parity word,   */
                                                     /*       and if result == 1    */
                    syndrome ^= 0x6815;              /*       then XOR syndrome     */
                                                     /*       with 6815 Hex.        */
               }                                     /*                             */
            }
      syndrome &= 0x7FFF;                        /* Finally, replace MSB of   */
      if(parity)                                 /*  syndrome word with the   */
              syndrome |= 0x8000;                    /*  calculated parity bit      */
      return(syndrome);
      }

/*      Function getbit_rx(n) should return the bit 'n' of the received      */
/*      codeword; Bit '1' is the first bit to be received, bit '64' the last.   */

      getbit_rx(n)
           {
           return(/* 1 or 0 */);
           }
```

1.4 Error Correction

Single-bit and bit-pair errors in a received codeword may be corrected by comparing the ‘Syndrome’ word (generated by the MX429 or the algorithm of section 3.4) against the entries in the following table, and if a match is found inverting the corresponding bits.

Syndrome (Hex)	Error bits	Syndrome (Hex)	Error bits	Syndrome (Hex)	Error bits	Syndrome (Hex)	Error bits
0003	14, 15	468D	40, 41	8001	15	B456	25
0006	13, 14	4841	61, 62	8002	14	B484	19
000C	12, 13	4989	33, 34	8004	13	B83F	62
0018	11, 12	4B7B	45, 46	8008	12	B887	34
0030	10, 11	4BD7	22, 23	8010	11	B929	46
0060	9, 10	4E0F	16, 17	8020	10	B94D	23
00C0	8, 9	502A	62, 63	8040	9	BA05	7
0180	7, 8	50CE	34, 35	8080	8	C000	1
0300	6, 7	51B7	46, 47	8100	7	C02E	36
0600	5, 6	51E1	23, 24	8200	6	C31C	50
0C00	4, 5	530D	17, 18	8400	5	C60A	39
15D3	43, 44	574C	41, 42	8800	4	C748	57
1763	20, 21	5A62	48, 49	88E9	60	C885	28
1800	3, 4	5CD1	47, 48	8A09	32	CA3E	54
18CD	28, 29	5CFA	24, 25	8CB1	44	D048	29
193B	59, 60	5D8C	18, 19	8D21	21	E401	38
1E1B	31, 32	6000	1, 2	9000	3	E588	41
21CD	56, 57	6039	36, 37	90C7	52	E685	56
220B	38, 39	6292	50, 51	91D2	59	E815	63
2867	35, 36	6334	26, 27	9412	31	E849	35
2BA6	42, 43	64EC	57, 58	9962	43	E89E	47
2D31	49, 50	650F	39, 40	9A2B	26	E8AC	24
2E7D	25, 26	6815	63, 64	9A42	20	E908	18
2EC6	19, 20	6CAE	52, 53	A000	2	EE2D	49
3000	2, 3	6F21	54, 55	A017	37	F07E	61
3149	51, 52	740B	15, 16	A18E	51	F10E	33
319A	27, 28	786C	29, 30	A305	40	F252	45
3276	58, 59	7897	60, 61	A3A4	58	F29A	22
3657	53, 54	7B07	32, 33	A51F	55	F40A	16
3C36	30, 31	7EE3	44, 45	A824	30	F91F	27
439A	55, 56	7FBB	21, 22	B2C4	42	FC69	53
4416	37, 38	8000	64	B44F	48		

Example:

Transmitted codeword:

```
Bit;  1                                                                                  64
      10001001   10101011   11001101   11101111   00010010   00110100   11111101   01000010
          errors;   xx
```

Received codeword:

```
   10001001   01101011   11001101   11101111   00010010   00110100   11111101   01000010
```

For this received codeword, the ‘Syndrome’ will be 0060H, which appears in the table, indicating that the 9th & 10th bits received are incorrect and should be inverted.