INTRODUCTION

The 13.56 MHz read/write devices (MCRF4XX) use a 16-bit Cyclic Redundancy Code (CRC) to ensure the integrity of data. Its polynomial and initial values are:

CRC Polynomial: \(X^{0}+X^{5}+X^{12}+X^{16} = 1000-0100-0000-1000-(1) = 8408\) (hex)

Initial Value: $FFFF$

This polynomial is also known as CRC CCITT-16. The interrogator applies the same polynomial to the incoming and transmitting data.

FIGURE 1: CCITT-16 CRC ENCODER

COMPUTATION ALGORITHM

Figure 1 shows the CCITT-16 CRC encoder. Figure 2 is the computational flow chart for computer programming.

The encoder consists of 16 shift registers and 3 exclusive-OR gates. The registers start with 1111-1111-1111-1111 (or FFFF in hex). The encoder performs XOR and shifts its content until the last bit is entered. The final register's content after the last data bit is entered is the calculated CRC value of the data set.

Example: The following procedure shows a workout example of the CRC calculation using the encoder.


Table 1 shows each step of the calculation. The content of the register after the last bit is 07F1. This 07F1 is the calculated CRC of the data.

When transmitting data, this calculated CRC is attached to the data. The interrogator sends the data and CRC with LSN (Least Significant Nibble) first. Therefore, the hex string to be sent will be: 981F25581F70 and for data = 8552F189.
**FIGURE 2: FLOW-CHART OF CRC COMPUTATION**

- **CRC Poly = 8408** (1000-0100-0000-1000)
- **Input Data**
- **Initialize CRC = FFFF** (1111-1111-1111-1111)

1. **Test Data Bit = LSB of Data Bits**
   - **K = XOR CRC (LSB) with Test Data Bit**

2. **K = 0**
   - **Yes**
     - Shift CRC to right by 1
     - **CRC = XOR CRC with CRC Poly**
   - **No (K = 1)**
     - Shift CRC to right by 1

3. **Shift Data Bits to right by 1**

4. **Is Test Data Bit the MSB of Data Bits?**
   - **Yes**
     - **Stop**
   - **No**
     - **Shift Data Bits to right by 1**
     - **Is Test Data Bit the MSB of Data Bits?**
### TABLE 1: CRC WORKOUT EXAMPLE FOR DATA = 8552F189 (HEX)

<table>
<thead>
<tr>
<th>Bit No.</th>
<th>Input Data</th>
<th>Register Contents</th>
<th>Hex Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial</td>
<td>1 1 1 1 1 1 - 1 1 1 1 1 1 1 - 1 1 1 1</td>
<td>FFFF</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0 1 1 1 1 1 - 0 1 1 1 1 1 1 - 0 1 1 1</td>
<td>FBF7</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>0 1 1 1 1 1 - 0 0 1 1 1 1 1 - 0 0 1 1</td>
<td>F9F3</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>0 1 1 1 1 1 - 0 0 0 1 1 1 1 - 0 0 0 1</td>
<td>F8F1</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>1 0 1 1 1 1 - 1 0 0 0 1 1 1 - 1 0 0 0</td>
<td>7C78</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>1 1 0 1 1 1 - 0 1 0 0 0 1 1 - 0 1 0 0</td>
<td>BA34</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>0 0 1 0 1 1 - 1 0 1 0 0 0 1 - 1 0 1 0</td>
<td>5D1A</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>1 1 0 1 0 1 - 0 1 0 1 0 0 0 - 0 1 0 1</td>
<td>AA85</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>0 1 1 0 1 0 - 0 0 1 0 1 0 0 - 1 0 1 0</td>
<td>D14A</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>1 1 1 1 0 1 - 1 0 0 1 0 1 0 - 1 1 1 0</td>
<td>ECAD</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>0 1 1 1 1 0 - 0 1 0 0 0 1 0 1 - 1 1 1 0</td>
<td>F25E</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>1 1 1 1 1 1 - 1 0 1 0 0 1 0 0 - 0 1 1 1</td>
<td>FD27</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>0 1 1 1 1 1 - 0 1 0 1 0 0 1 1 - 1 0 1 1</td>
<td>FA9B</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>0 1 1 1 1 1 - 0 0 0 1 0 1 0 0 - 0 1 0 1</td>
<td>F945</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>1 0 1 1 1 1 - 1 0 0 1 0 1 0 0 - 0 0 1 0</td>
<td>7CA2</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>1 0 0 1 1 1 - 1 1 0 0 1 0 1 0 - 1 0 0 1</td>
<td>3E51</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>0 1 0 0 1 1 - 1 1 0 0 1 0 0 0 - 0 0 0 0</td>
<td>9B20</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>1 1 1 0 0 1 - 0 0 0 1 1 0 0 1 - 1 0 0 0</td>
<td>C998</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>1 1 1 1 0 0 - 0 0 0 1 1 0 0 0 - 1 1 1 0</td>
<td>E0C4</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>1 1 1 1 1 0 - 1 0 0 0 1 1 0 0 - 1 0 1 0</td>
<td>F46A</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>1 1 1 1 1 1 - 1 1 0 0 0 1 1 1 - 1 1 0 1</td>
<td>FE3D</td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>1 0 1 1 1 1 - 1 1 1 0 0 0 1 1 - 1 1 1 0</td>
<td>7F1E</td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>0 0 0 1 1 1 - 1 1 1 1 0 1 0 0 - 1 1 1 1</td>
<td>3F8F</td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>0 1 0 1 1 1 - 0 1 1 1 1 0 1 0 - 1 1 1 1</td>
<td>9BCF</td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>0 1 1 0 0 1 - 0 0 1 1 1 1 0 1 - 1 1 1 1</td>
<td>C9EF</td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>0 1 1 1 0 0 - 0 0 0 1 1 1 1 1 - 1 1 1 1</td>
<td>E0FF</td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>0 1 1 1 1 0 - 1 1 0 0 0 1 1 1 - 1 1 1 1</td>
<td>F477</td>
<td></td>
</tr>
<tr>
<td>27</td>
<td>0 1 1 1 1 1 - 1 1 0 0 0 1 1 1 - 0 0 1 1</td>
<td>FE33</td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>1 0 1 1 1 1 - 1 1 1 0 0 0 1 1 - 1 0 0 1</td>
<td>7F19</td>
<td></td>
</tr>
<tr>
<td>29</td>
<td>1 0 0 1 1 1 - 1 1 1 0 0 0 1 1 - 1 1 1 0</td>
<td>3F8C</td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>0 0 0 1 1 1 - 1 1 1 1 1 1 0 0 - 0 1 1 0</td>
<td>1FC6</td>
<td></td>
</tr>
<tr>
<td>31</td>
<td>0 0 0 0 0 1 - 1 1 1 1 1 1 1 0 - 0 0 1 1</td>
<td>0F6E</td>
<td></td>
</tr>
<tr>
<td>32</td>
<td>1 0 0 0 0 0 - 1 1 1 1 1 1 1 1 - 0 0 0 1</td>
<td>07F1</td>
<td></td>
</tr>
</tbody>
</table>
APPENDIX A: EXAMPLE WITH C-SOURCE CODE FOR CRC CALCULATION

```c
#include <stdio.h>
#include <stdlib.h>
#include "onescnt.h"
#define NULL 0
#define true 1
#define false 0

void main (int argc, char *argv[])
{
    int i, j, k, message[40], num_bits, bitcount, bytecount, crc, next_bit, crc_temp, message_temp;
    int maskreg[8] = {1, 2, 4, 8, 16, 32, 64, 128};
    int crc_nibble[4];
    char ch
    FILE *fin;
    if (argc != 2)
    {
        printf("proper usage is CCITT {indata file with data in hex}\n");
        abort();
    }
    if ( (fin =fopen(argv[1], "r")) ==NULL)
    {
        printf("Can't open %s\n", argv[1]);
        abort();
    }
    i = 0;
    while ( (ch=fgetc(fin)) !=EOF)
    {
        message_temp = 0;
        //retrieve the input data field and convert to an integer message field
        if ((ch >= 'a') && (ch <= 'f')) ch = ch - 0x20
        if ((ch >= 'A') && (ch <= 'F')) ch = ch - 0x70
        if ((ch >= '0') && (ch <= '?'))
        {
            message_temp = ch - '0';
            message[i++] = message_temp;
        }
        // At this point, message[] holds data with nibbles (4 bits on each array). This will be used for
        CRC calculation
        message[i] = -1;
        k = i;
        // The above is used for array checking and k value is the total number of nibbles.
        printf("Read in %d nibbles. \n", k);
        printf("Original data in hex read in from data file: \n");
        for (i = 0; i < k; i++)
        {
            printf("%x \n", message[i]);
        }
        printf("\n\n");
        // Now computing the CRC of data
        //-------- Initialization -----------------------------
        crc = 0xffff; //initial CRC value
        crc_poly = 0x8408; //1000-0100-0000-1000
        //-----------------------------------------------
        printf("Initial CRC value in hex: %x ... \n", crc);
        num_bits = k*4;
        for ( i = 0; i < num_bits; i++)
        {
            bitcount = i % 4;
            bytecount = i/4;
            next_bit = (message[bytecount] & maskreg[bitcount]); //This will find the next data bit to apply
            next_bit = ((next_bit >> bitcount) & 1); //This will move the current data bit to LSB of next_bit
            // and make all bits except LSB bit to zero
            crc_temp = crc^next_bit; //xor the last nibble of crc (actually the last bit of CRC) with next_bit if (crc_temp & 1)
            printf("xor = 1\n");
            crc = crc >> 1; //Shift the crc by 1 to right
            crc = crc^crc_poly; //xor current crc with crc_poly
        }
    }
}
```
crc = crc|0x8000; //this may not be necessary
}
if (crc_temp & 1)
{
  printf ("xor = 0\n"");
  crc = crc >> 1;
  crc = crc & 0x7fff; // this may not be necessary
}
printf("Temp CRC after iteration %d: ", i);
for (j = i; j<num_bits; j++)
  printf(" ");
printf("\n", crc);
crc_nibble [0] = crc & x000f;
crc_nibble [1] = (crc & x000f >> 4);
crc_nibble [2] = (crc & x000f >> 8);
crc_nibble [3] = (crc & x000f >> 12);
printf("Bit order for shifting in nibbles in LSB first. \n");
printf("\n CRC at end: %x ", crc);
printf ("Send %x %x %x %x \n", crc_nibble[0], crc_nibble[1],crc_nibble[2],crc_nibble[3],);
printf("\n\n");
fclose(fin);
“All rights reserved. Copyright © 2001, Microchip Technology Incorporated, USA. Information contained in this publication regarding device applications and the like is intended through suggestion only and may be superseded by updates. No representation or warranty is given and no liability is assumed by Microchip Technology Incorporated with respect to the accuracy or use of such information, or infringement of patents or other intellectual property rights arising from such use or otherwise. Use of Microchip's products as critical components in life support systems is not authorized except with express written approval by Microchip. No licenses are conveyed, implicitly or otherwise, under any intellectual property rights. The Microchip logo and name are registered trademarks of Microchip Technology Inc. in the U.S.A. and other countries. All rights reserved. All other trademarks mentioned herein are the property of their respective companies. No licenses are conveyed, implicitly or otherwise, under any intellectual property rights.”

Trademarks

The Microchip name, logo, PIC, PICmicro, PICMASTER, PICSTART, PRO MATE, KEELQ, SEEVAL, MPLAB and The Embedded Control Solutions Company are registered trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

Total Endurance, ICSP, In-Circuit Serial Programming, FilterLab, MXDEV, microID, FlexROM, fuzzyLAB, MPASM, MPLINK, MPLIB, ICEPIC, Migratable Memory, FanSense, ECONOMONITOR, SelectMode and microPort are trademarks of Microchip Technology Incorporated in the U.S.A.

Serialized Quick Term Programming (SQTP) is a service mark of Microchip Technology Incorporated in the U.S.A.

All other trademarks mentioned herein are property of their respective companies.

© 2001, Microchip Technology Incorporated, Printed in the U.S.A., All Rights Reserved.