CRC Calculation Programming Guide

This document will mainly introduce one programming approach to calculate the CRC value based on the CRC polynomial, $X^8 + X^5 + X^4 + 1$, that used in TPS929120-Q1 FlexLED protocol. The related theory knowledge of CRC will not be explained in this document.

1. FlexLED Protocol Introduction

Take an example of the single-byte write command frame of FlexLED protocol, as showed in the Figure 1. The CRC byte is the calculated result based on all the prior writing bytes, except for the "SYNC" byte. Every time after specifying the command frame, DEV_ADDR, REG_ADDR and DATA_x(s), we need to calculate the CRC byte based on these data bytes and append it to the end of them to construct the entire command frame.

Figure 1. Single-byte FlexLED Write Command

2. CRC Calculation Implementation

As showed in figure 2,the CRC data byte contain 8 bits, once there is new data bit input, all the bit will shift 1 bit left with 3 XOR operations.

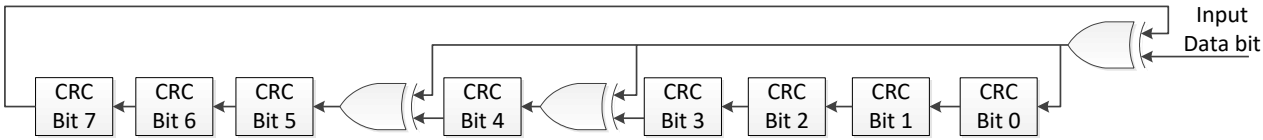

Figure 2. Implementation Flow Diagram

3. Lookup Table Method to Calculate CRC

In order to improve the calculation efficiency, we can calculate and store the CRC of all 256 data, from 0x00 to 0xFF, firstly in order. Later we can take the input byte data as the index to retrieve the corresponding CRC directly.

4. Programming

4.1 Basic CRC calculation

As defined in the datasheet, the CRC initial value is 0xFF, so the first thing is to set the CRC byte with the initial value. Then follow the below steps to calculate the final CRC byte.

1) Convert the first input date byte into 0/1 bit sequence, for example, the 0/1 bit sequence of "0xB1" is "1011 0001".
2) From the LSB to MSB, pass the bit data to do the shift and XOR operations based on Figure 2. After 8 loops, the fist input data byte's CRC value is calculated out.
3) Repeat first step to convert next input data byte.
4) Repeat second step to calculate the CRC value. Please note that when pass the LSB of the second data to do the shift and XOR operations, the CRC initial value is the calculated value in second step not 0xFF at the moment.
5) After the last input data byte having been taken into calculation, we get the final CRC byte.

4.2 Lookup Table Method

1) Calculate the CRC of data 0x00 to 0xFF and store them into one array in order
2) Do one XOR operation with the input data byte and the initial byte
3) Use the calculated result in above second step as the array index to retrieve its CRC value from the CRC array built in first step
4) Reverse the retrieved value in above third step to get the input data byte's final CRC

4.3 Example code

4.3.1 Basic CRC Calculation

Below is the CRC Calculation example code based on MSP430 series MCU.

```
#include <msp430.h>
#include <FlexLED.h>

#define BIT0                    (0x01)
#define BIT1                    (0x02)
#define BIT2                    (0x04)
#define BIT3                    (0x08)
#define BIT4                    (0x10)
#define BIT5                    (0x20)
#define BIT6                    (0x40)
#define BIT7                    (0x80)

unsigned int CRC(unsigned int commandFrame_withoutCRC[], unsigned int byteLength)
{
    unsigned int j;
    unsigned int CRCtemp;

    for(j=0; j<byteLength-1; j++)
    {
        if(j==0)
        {
            CRCtemp = CRC_Calculation(0xFF, commandFrame_withoutCRC[j+1]); //CRC
initial value is 0xFF, and SYNC byte is not engaged in CRC calculation
        }
        else
        {
```

```
            CRCtemp = CRC_Calculation(CRCtemp, commandFrame_withoutCRC[j+1]);//assign
the last calculated CRC to CRC_initial and calculate the second, third,..., byte's
CRC value until all bytes have been calculated
        }
    }
    return CRCtemp; // after calculating the final byte's CRC value we get this
frame's CRC
}

unsigned int CRC_Calculation(unsigned int CRC_Initial, unsigned int Input_Data)
{
    unsigned int Temp_Bit, Input_Bit;
    unsigned int bit0, bit1, bit2, bit3, bit4, bit5, bit6, bit7;// store every bit
value of Input_Data
    unsigned int CRC=0; //store the Input_Data (byte) 's CRC
    unsigned int i=0;
    bit0 = CRC_Initial & BIT0; //get every bit of CRC initial value
    bit1 = (CRC_Initial & BIT1)>>1;
    bit2 = (CRC_Initial & BIT2)>>2;
    bit3 = (CRC_Initial & BIT3)>>3;
    bit4 = (CRC_Initial & BIT4)>>4;
    bit5 = (CRC_Initial & BIT5)>>5;
    bit6 = (CRC_Initial & BIT6)>>6;
    bit7 = (CRC_Initial & BIT7)>>7;
    while(i<8)
    {
        Input_Bit = (Input_Data >> i) & 0x01; //extract one bit of Input Data (from
LSB to MSB)
        Temp_Bit = Input_Bit ^ bit7; // Do Input_Bit XOR bit7
        bit7 = bit6;
        bit6 = bit5;
        bit4 = bit4 ^ Temp_Bit; //Do bit4 XOR Temp_Bit
        bit5 = bit4;
        bit3 = bit3 ^ Temp_Bit; //Do bit3 XOR Tmep_Bit
        bit4 = bit3;
        bit3 = bit2;
        bit2 = bit1;
        bit1 = bit0;
        bit0 = Temp_Bit;
        i++;
    }
    CRC = (bit7<<7)|(bit6<<6)|(bit5<<5)|(bit4<<4)|(bit3<<3)|(bit2<<2)|(bit1<<1)|bit0;
    return CRC;
}
```

4.3.2 Look Up Table Method

Below is the CRC Calculation example code based on MSP430 series MCU.

```
#include <msp430.h>
#include <FlexLED.h>
#define polynomialINV 0x8C //as UART transmit from LSB to MSB, invert the polynomial
0x31 (0011 0001) to 0x8C (1000 1100)
#define LSB 0x01

extern unsigned int crcArray[256];
```

```
void crcInitial() //calculate and store the CRC of data from 0x00 to 0xFF
{
    unsigned int k, j, remainder;

    for(k=0;k<256;k++)
    {
        remainder=k;
        for(j=8;j>0;j--)
        {
            if(remainder & LSB)
            {
                remainder = (remainder>>1) ^ polynomialINV; //right shift 1 bit and
do the XOR operation with 0x8C
            }
            else
            {
                remainder = remainder>>1;
            }
        }
        crcArray[k]=remainder;
    }
}

unsigned int CRC_LUT(unsigned int commandFrame_withoutCRC[], unsigned int byteLength)
//calculate CRC of command frame
{
    unsigned int remainder, tempData, k;

    remainder = 0xFF; //assign the initial value 0xFF

    for(k=1;k<byteLength;k++)     //the first SYNC byte not engage CRC calculation
    {
        tempData = remainder ^ commandFrame_withoutCRC[k]; //input data byte XOR
remainder
        remainder = crcArray[tempData]; //use tempData as the index to retrieve its
CRC from crcArray
    }

    //we have to reverse the final remainder to get the CRC value, for example, if
final remainder = 0010 1100, we need to reverse it to 0011 0100
    remainder = ((remainder & 0x80)>>7) + ((remainder & 0x40) >>5) + ((remainder &
0x20) >>3) + ((remainder & 0x10)>>1) + ((remainder & 0x08)<<1) + ((remainder & 0x04)
<<3) + ((remainder & 0x02) <<5) + ((remainder & 0x01)<<7);
    return remainder;
}
```