//4字节16进制数转浮点数子函数

function TfrmMain.Hex4ToFloat(TmpInteger : Integer; nType : Integer):String;

const                                  //nType 小数点后面几位

    c\_nagative : Byte = $01; //负数

    c\_plus : Byte = $02; //正数

    c\_HighOne : Byte = $01; //x>1

    c\_LowOne : Byte = $02; //0<x<1

    c\_Zero : Byte = $00;

var

    Sec\_n : Byte;

    n\_sign : Byte;

    n\_sign1 : Byte;

    i : Integer;

    n\_JE : Byte;

    nMode : Byte;

    retStr : String;

    TmpDouB : Double;

begin

    nMode := 0;

    if nMode = 0 then

    begin

        if TmpInteger = 0 then

        begin

            n\_sign := c\_Zero;

            nMode := $ff;

        end

        else

        begin

            nMode := 1;

        end;

    end;

    if nMode = 1 then //判断是正数还是负数

    begin

        if (TmpInteger and $00800000) <> 0 then

        begin

            n\_sign := c\_nagative; //负数

        end

        else

        begin

            n\_sign := c\_plus; //正数

        end;

        nMode := 2;

    end;

    if nMode = 2 then //提取阶码

    begin

        TmpInteger := TmpInteger and $ff7fffff;

        n\_JE := ((TmpInteger and $ff000000) shr 24) and $ff;

        if n\_JE >= ($80-1) then //x>=1

        begin

            n\_sign1 := c\_HighOne;

        end

        else                   //0<x<1

        begin

            n\_sign1 := c\_LowOne;

        end;

        nMode := 3;

    end;

    if nMode = 3 then

    begin

        if n\_sign1 = c\_HighOne then

        begin

            Sec\_n := n\_JE - ($80-1);

        end

        else //if n\_sign1 = c\_LowOne then

        begin

            Sec\_n := ($80-1) - n\_JE;

        end;

        nMode := 4;

    end;

    if nMode = 4 then

    begin

        TmpInteger := TmpInteger and $ffffff; //去掉阶码部分

        if n\_sign1 = c\_HighOne then

        begin

            TmpInteger := TmpInteger or $00800000;

            TmpDouB := TmpInteger / (Trunc(Power(2,(24-Sec\_n))));

        end

        else //if n\_sign1 = c\_LowOne then

        begin

            TmpInteger := TmpInteger or $00800000;

            TmpDouB := TmpInteger / (Trunc(Power(2,(24+Sec\_n))));

        end;

        nMode := 5;

    end;

    if nMode = 5 then

    begin

        if nType = 0 then

        begin

            retStr := FormatFloat('0',TmpDouB);

        end

        else if nType = 1 then

        begin

            retStr := FormatFloat('0.0',TmpDouB);

        end

        else if nType = 2 then

        begin

            retStr := FormatFloat('0.00',TmpDouB);

        end

        else if nType = 3 then

        begin

            retStr := FormatFloat('0.000',TmpDouB);

        end

        else if nType = 4 then

        begin

            retStr := FormatFloat('0.0000',TmpDouB);

        end

        else if nType = 5 then

        begin

            retStr := FormatFloat('0.00000',TmpDouB);

        end

        else if nType = 6 then

        begin

            retStr := FormatFloat('0.000000',TmpDouB);

        end

        else if nType = 7 then

        begin

            retStr := FormatFloat('0.0000000',TmpDouB);

        end

        else if nType = 8 then

        begin

            retStr := FormatFloat('0.00000000',TmpDouB);

        end

        else if nType = 9 then

        begin

            retStr := FormatFloat('0.000000000',TmpDouB);

        end

        else if nType = 10 then

        begin

            retStr := FormatFloat('0.0000000000',TmpDouB);

        end

        else if nType = 11 then

        begin

            retStr := FormatFloat('0.00000000000',TmpDouB);

        end

        else if nType = 12 then

        begin

            retStr := FormatFloat('0.000000000000',TmpDouB);

        end;

        nMode := $ff;

    end;

    if nMode = $ff then

    begin

        if n\_sign = c\_nagative then

        begin

            retStr := '-' + retStr;

        end

        else if n\_sign = c\_Zero then

        begin

            retStr := '0';

        end;

        Result := retStr;

    end;

end;