编　　制	沈胜文
审　　核	
批　　准	
实施责任人	

Amendment history 修改历史记录

版本号	修改说明	修改批准人	修改人	日期	签收人
101	创建文档		沈胜文	2008-3-21	
102	修改		沈胜文	2008-4-11	

编　　制	沈胜文
审　　核	
批　　准	
实施责任人	

Table of Contents 目录

1. Introduction 简介

1.1. Objective 编写目的

本文档是在《CRC-32 MD5概要设计》的基础上，就文件或分段完整性校验所需要的扩展库，进行详细设计而完成的详细设计说明；

在 MuFTAD 项目中，涉及二种不同的校验方式：CRC 和 MD5，对此，在本设计文档中，将会详细说明二者的实现；

在概要设计文档中，介绍了待校验文件的内容，因此，针对具体的校验内容，必须提供具体的校验方法；

本文档将尽可能详尽地说明相关数据结构的设计和开发，但是设计和开发不相符之处，需讨论决定，并且修改本文档；

最终的设计以代码为准；

1.2. Background 背景

本程序是法电 MuFTAD 项目的一部分，提供内容完整性校验。但是程序按扩展库的形式组织和开发，以使得该程序独立于本项目，方便于其他项目的使用和移植；

本软件的提出者：沈胜文
本软件的开发者：沈胜文
本软件的用　户：MuFTAD 和微元其他项目

1.3. Terms & Abberviation 术语&缩写

Terms&Abbreviation 术语&缩写	Description 解释

CRC	循环冗余检验（Cyclic Redundant Check），使原始数据通过某种算法，得到一个新的数据，而这个数据与原数据有着固有的内在关系。把原数据和新数据组合在一起，使得新数据具有自我校验功能；
MD5	md5的全称是 message-digest algorithm 5，MD5的典型应用是对一段信息（Message）产生信息摘要（Message-Digest），以防止被篡改。
openssl	OpenSSL 是一个开放源代码的实现了 SSL 及相关加密技术的软件包，由加拿大的 Eric Yang 等发起编写的。

1.4. Reference Material 参考资料

CRC-32 MD5 概要设计
AN42-循环冗余检验（CRC）原理与实现1.1.pdf
并行 CRC_32校验码生成算法研究及其实现.pdf
openssl.org

2. Rules 规则

2.1. Name Rules 命名规则

该程序的建立不仅仅是为了 MuFTAD 项目，我们希望其能为微元其他项目提供服务，因此在开发过程中，对其按照扩展库的形式进行组织和开发，所有的函数形如：Mu_XXXX()；

2.2. Illuminate 说明

针对每个程序，都必须注明其开发目的，开发者，开发时间，等等。以下字段必须被包含于程序的开头部分。

```
/*************************************************************
*                    ====Microunit Techonogy Co,.LTD.====
* File Name:
*
```

```
*          CRC-32 & MD5.c
*
* Description:
*
*        Use the Routines contained in this file, to checksum the files ,
*        or the segment of the file.
*
* Revision History:
*
*          20-3-2008 ver1.0
*
*Author:
*
*          ssw    (fzqing@gmail.com)
*
*                        ***PROTECTED BY COPYRIGHT***
**********************************************************************/
```

2.3. Note Rules 注释规则

程序中的各个函数均需要明确注释其功能，并能简要描述其实现，及注意点。特别应该注意的是：在描述时，应该详细包括对锁，输入和输出进行详细说明。

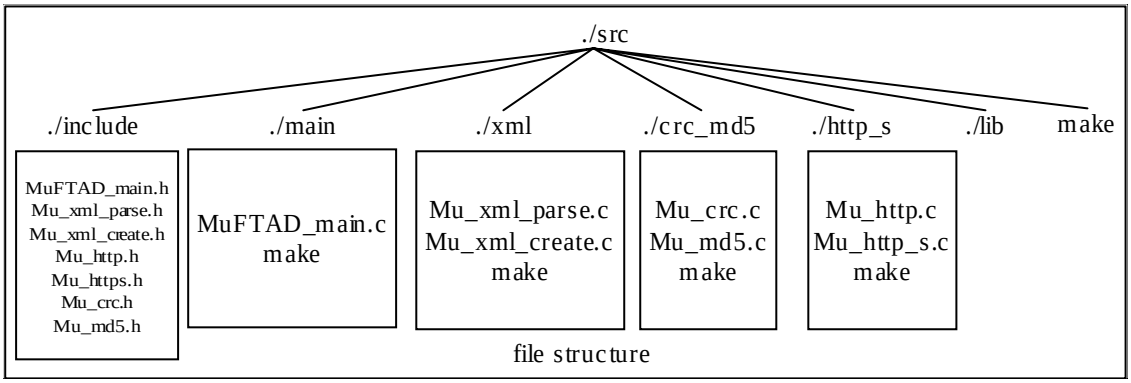
可参考模板

```
/**********************************************************************
*Description:
*          This Function checksum the crc-32,
*          Value pointer;
*Input:
*          filename: the file name , which stored programs
*Output:
*          Pointer:
*LOCK:
*          NONE
*Modify:
*          ssw (fzqing@gmail.com    10-3-2008)
**********************************************************************/
```

2.4. File Structure 文件结构

[注意]:

1、此为整个项目的文件组织方式;



./src/include：文件夹，包含该项目中的所有头文件；

./src/main：文件夹，包含所有按法电《do3c00_SoftProtocol_0.1.0_RC1》流程所开发的程序；

./src/xml：文件夹，包含项目中所需要的 xml 处理库函数源代码；

./src/crc_md5：文件夹，包含项目中所需要的校验函数源代码，包括 CRC 和 MD5校验代码；

./src/http_s：文件夹，包含项目中所需要的与服务器交互的方式，包括 HTTP(s) GET、POST 方式；

./src/lib：文件夹，用于存储编译所生成的 xml、http 和 https、crc/md5库。软件编译连接时使用该文件夹下的库；

./src/make：文件，总的编译入口；

[注意]:

1、各对应文件夹下的源文件按需要添加，但是所作修改必须对 makefile 文件作相应的修改，以正确编译;

3. Structure Of Routines 程序结构

3.1. Overview 总述

待开发的各程序之间是相互独立的，它们为项目 MuFTAD 中的其他部分服务，按功能可以分为如下二类：

1、CRC-32校验

使用 CRC-32对完整的文件或是文件分段进行完整性校验；

2、MD5校验

使用 MD5散列算法，对完整的文件或是文件分段进行完整性校验；

3.2. Routines List 函数列表

4. Global Description 全局描述

4.1. Global Type 全局类型

4.1.1. Macro 宏定义

```
#define MAX_CRC_LEN 1023
#define unit_fast32_t unsigned int
```

4.1.2. Structure 结构体

```
static uint_fast32_t CRC32_Tab[256] =
{
    0x0,
    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
};

[注意]:

1、该 CRC32校验表来自 Linux 工具代码 cheksum.c

4.2. Global Error 全局错误码

[注意]:

1、该错误码应该被放在一个单独的，只用来定义错误的头文件中;

2、该头文件定义为: mu_error.h

```
#define NO_MUERROR 0
#define MUERROR_BUFFER_EMPTY -1
```

```
#define MUERROR_OVER_LEN -10
#define MUERROR_READ_FILE -11
#define MUERROR_NOT_MATCH -12
……
```

5. Routines Detail 函数细节

5.1. Mu_Crc32File

5.1.1. Name 函数名称

int Mu_Crc32File(int fd, unsigned int crc32)

5.1.2. Description 函数描述

本函数用于校验文件的完整性，使用 CRC32方式完成。函数对 fd 文件句柄所关联的文件内的所有内容，与 crc32一并进行 CRC32运算，返回校验状态；

本函数对进行校验的文件不作任何修改，并且在校验完毕后，也不关闭该文件句柄，该操作由调用者完成；

为了提高校验效率，使用查表法完成，CRC32表见4.1.章节所述；

5.1.3. Function 功能

用 CRC-32的校验方式对整个文件内容进行校验，以函数返回值的形式返回校验结果；

5.1.4. Capability 性能

产用查表方式，以提高 CRC-32的校验速度；

5.1.5. Input 输入

fd：待校验文件的文件句柄；
crc32：待校验文件的 CRC 值，同原文件一起校验文件的完整性；

5.1.6. Output 输出

校验状态值；
[注意]:
1、可以参考4. 2. 章节;

5.1.7. Arithmetic 算法

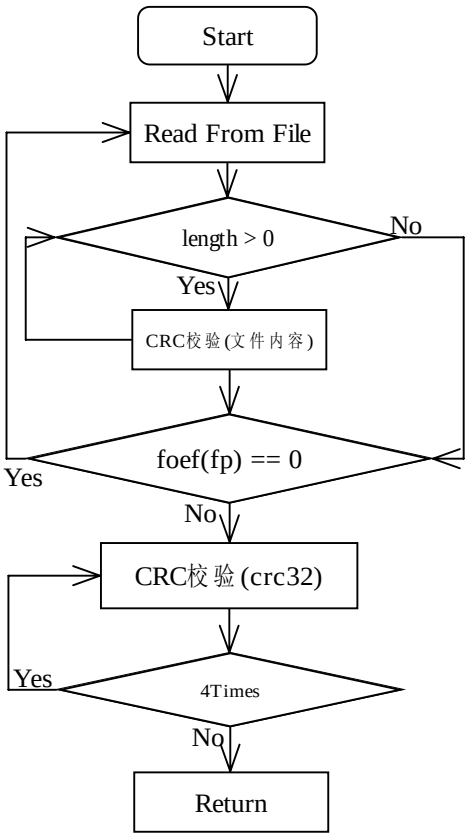
CRC-32 查表法在实现原理：
本字节后的 CRC 码等于上一字节余式 CRC 码的低8位左移8位后，再加上一字节 CRC 右移24位（也即 CRC 的高8位）和本字节之和后所求得的 CRC 码；

核心代码如下：

```
crc = (crc << 8) ^ crctab[((crc >> 24) ^ *cp++) & 0xFF];
```

[注意]:
1、本章节所说的和是指机器语言中的异或运算;

5.1.8. Process Flow 处理流程



5.1.9. Pseudocode 伪代码

```
int Mu_Crc32File(int fd, unsigned int crc32)
{
	unsigned int crc = 0;
	unsigned char buf[MAX_CRC_LEN + 1];
	unsigned char *cp =NULL;
	unsigned char temp;
	size_t bytes_read;
	size_t length = 0;

	int i;

	memset(buf, 0, MAX_CRC_LEN + 1);
	while((bytes_read = read(fd, buf, MAX_CRC_LEN)) > 0){
		cp = buf;
```

```
        if(length + bytes_read < length){
            //file is too long
            return MUEOVL;
        }

        length += bytes_read;

        while(bytes_read --)
            crc = (crc<<8)^CRC32_Tab[((crc>>24)^*cp++&0xFF];

        //memset(buf, 0, MAX_CRC_LEN + 1);

        if(feof(fp))
            break;
    }

    if(feof(fp)){
        do error
        return MUEEOF;
    }

    i = 4;
    while(i){
        temp = (crc32 >>((i-1)* 8))&0xFF;
        crc = (crc<<8)^CRC32_Tab[((crc>>24)^temp&0xFF];
    }

    if(0 == crc)
        return crc;
    else
        return MUNMAH;
}
```

5.1.10. Interface 接口

略

5.1.11. Malloc 存储分配

函数使用局部变量，申请栈中空间，函数将文件中内容读取到该空间，进行CRC 校验，在结束后，自动将内存区域归还系统；

5.1.12. Restrict 限制

略

5.1.13. Test 测试

准备一个待检测的文件，用工具得到其 CRC32值；
写一个单独的调用函数，按函数规定赋给参数，测试函数是否校验正确；

5.1.14. Unsolve 未解决情况

略

5.2. Mu_Crc32Segment

5.2.1. Name 函数名称

int Mu_Crc32Segment(int fd, off_t seek, size_t length, unsigned int crc32)

5.2.2. Description 函数描述

本函数用于校验文件分段的完整性，使用 CRC32方式完成。函数对 fd 文件句柄所关联的文件内，从 seek 标识位置开始，length 长度的内容及 crc32的值一并进行 CRC32运算，返回校验状态；

本函数对进行校验的文件不作任何修改，并且在校验完毕后，也不关闭该文件句柄，该操作由调用者完成；

为了提高校验效率，使用查表法完成，CRC32表见4.1.章节所述；

5.2.3. Function 功能

用 CRC-32的校验方式对文件内的部分内容进行校验，以函数返回值的形式返回校验结果；

5.2.4. Capability 性能

采用查表方式，以提高 CRC-32的校验速度；

5.2.5. Input 输入

fd：待校验文件的文件句柄；
crc32：待校验文件的 CRC 值，同原文件一起校验文件的完整性；
seek：标识待检验分段在文件中的起始位置；
length：标识进行检验时，需要输入的总的字符数；

5.2.6. Output 输出

校验状态值；
[注意]:
1、可以参考4.2.章节;

5.2.7. Arithmetic 算法

同5.1.7.中说明

5.2.8. Process Flow 处理流程

同5.1.8.中说明

[注意]:
1、在文件读取上稍的不同;

5.2.9. Pseudocode 伪代码

```
int Mu_Crc32Segment(int fd, off_t seek, size_t length, unsigned int crc32)
{
    unsigned int crc = 0;
    unsigned char buf[MAX_CRC_LEN + 1];
    unsigned char *cp =NULL;
```

```
    size_t bytes_read;
    size_t len = 0;
    int i;

    memset(buf, 0, MAX_CRC_LEN + 1);
    seek(fp, seek, SEEK_SET);
    while((bytes_read = read(fd, buf, MAX_CRC_LEN)) > 0){
        cp = buf;
        if(len + bytes_read < len){
            //file is too long
            return MUEOVL;
        }
        if(len + bytes_read > length)
            bytes_read = length – len;

        len += bytes_read;

        while(bytes_read --)
            crc = (crc<<8)^CRC32_Tab[((crc>>24)^*cp++&0xFF];

        memset(buf, 0, MAX_CRC_LEN + 1);

        if(feof(fp) || (len == length))
            break;
    }

    if(ferror(fp) || (len != length)){
        do error
        return MUERAD;
    }

    i = 4;
    while(i){
        temp = (crc32 >>((i-1)* 8))&0xFF;
        crc = (crc<<8)^CRC32_Tab[((crc>>24)^temp&0xFF];
    }

    if(0 == crc)
        return crc;
    else
        return MUEMAH;

}
```

5.2.10. Interface 接口

略

5.2.11. Malloc 存储分配

同5.1.11.中说明

5.2.12. Restrict 限制

略

5.2.13. Test 测试

同5.1.13.中说明

5.2.14. Unsolve 未解决情况

略

5.3. Mu_Md5File

5.3.1. Name 函数名称

int Mu_Md5File(int fd, char *md5)

5.3.2. Description 函数描述

本函数使用 MD5方式校验文件的完整性。函数对 fd 文件句柄所关联的文件内的所有内容进行 MD5运算，将动算结果与参数值 md5比较，返回校验状态；

本函数对进行校验的文件不作任何修改，并且在校验完毕后，也不关闭该文件句柄，该操作由调用者完成；

MD5校验使用 openssl 库完成；

5.1.3. Function 功能

用 MD5的校验方式对整个文件内容进行校验，以函数返回值的形式返回校验结果；

5.1.4. Capability 性能

略；

5.1.5. Input 输入

fd：待校验文件的文件句柄；
md5：待校验文件的 MD5值，同原文件一起校验文件的完整性；

5.1.6. Output 输出

校验状态值；
[注意]:
1、可以参考4. 2. 章节;

5.1.7. Arithmetic 算法

略

5.1.8. Process Flow 处理流程

略

5.1.9. Pseudocode 伪代码

```
int Mu_Md5File(int fd, char *md5)
{
```

```
    char ouput[33] = {“”};

    MD5_CTX context;
    size_t len = 0;
    size_t bytes_reaed;
    int retval = MUOK;

    unsigned char buf[MAX_CRC_LEN + 1];
    unsigned char digest[16];

    Md5Init(&context);

    memset(buf, 0, MAX_CRC_LEN + 1);
    while((bytes_read = read(fd, buf, MAX_CRC_LEN)) > 0){
        if(len + bytes_read < len){
            //file is too long
            return MUEOVL;
        }

        len += bytes_read;
        Md5Update(&context, buf, bytes_read);

        memset(buf, 0, MAX_CRC_LEN + 1);

        if(feof(fp))
            break;
    }

    if(ferror(fp)){
        do error
        return MUEEOF;
    }

    MD5Final(digest, &context);

    for(int i = 0; i < 16; i++){
        sprintf(&(ouput[2*i], “%02x”, (unsigned char)digest[i]);
        sprintf(&(ouput[2*i+1], “%02x”, (unsigned char)(digest[i]<<4));
    }

    retval = strncmp(output, md5, 32);
    return retval;
}
```

5.3.10. Interface 接口

略

5.3.11. Malloc 存储分配

同5.1.11.中说明

5.3.12. Restrict 限制

略

5.3.13. Test 测试

同5.1.13.中说明

5.3.14. Unsolve 未解决情况

略

5.4. Mu_Md5Segment

创建函数接口
int Mu_Md5Segment(int fd, off_t seek, size_t length, char *md5)

具体实现参考5.2和5.3.章节