Misty　を利用した暗号ソフト

（C）宇山　靖政
2012.04.08

　三菱電機が開発した暗号ソフト Misty は、軽量で処理速度が速いのが特徴です。
利用規定は、

三菱電機の HP によれば、

三菱電機株式会社は、通信ネットワークにおける個人のプライバシや機密情報を保護するための必須技術である暗号方式について、当社の開発した暗号アルゴリズム MISTY の基本特許（現在出願中）を無償許諾にすることにいたしました。

MISTY は、米国標準暗号 DES を解読した当社の持つ世界最高水準の暗号強度評価技術をもとに 1995 年に設計された暗号方式です。

1996 年 7 月には第三者による安全性評価研究の目的で仕様を公開しました。当社では MISTY をもちいたインターネット・イントラネットセキュリティ製品群を開発しております。仕様の公開に先立ち、暗号アルゴリズム MISTY の基本特許を主要各国に向け出願いたしましたが、このたび、MISTY の一層の普及をめざし、前述の出願により得られる MISTY の基本特許を無償で許諾することを決定いたしました。なお、日本国内において、企業が保有する暗号技術の特許許諾を無償化する試みは今回が初めてです。

又、この無償許諾の方針は、情勢、その他の事情により将来変更する場合があります。

その場合も、締結済みの上記無償実施許諾契約につきましては、方針変更後も有効に存続させる所存です。

背景と狙い

インターネットの普及と共に、オープンな環境で情報の安全性を確保することが重要な課題となっています。この情報の安全性を確保するためには、安全性の高い暗号方式を使用した使いやすい製品や情報システムの開発が急務です。当社では MISTY 仕様公開以来、これを核としたセキュリティ製品群を提供してまいりました。MISTY は発表以来内外からその安全性と高速性に関して高い評価をうけており、社外からも MISTY を利用したいと要望が増加しています。

今回の基本特許の許諾の無償化は、これらの要望にこたえるとともに、MISTY の一層の普及による暗号製品の相互接続性を高め、各ベンダーの MISTY 搭載製品の開発・製品化が容易になり、それらベンダーの暗号製品市場、更には、暗号を必要とする情報通信市場への参入を促すことをめざしてます。暗号方式の特許実施料の無償化の例としては米国標準暗号 DES がありますが、日本国内では今回が初めてです。

ページトップに戻る

無償許諾について

ご存知のとおり弊社は、すでにこの暗号アルゴリズム MISTY の仕様を下記のとおり、学会などを通じて公開しております。

学会発表情報

- 松井充, : " ブロック暗号アルゴリズム MISTY",
  電子情報通信学会 情報セキュリティ研究会 ISEC96-11（1996）（PDF：206KB）

- Matsui, M., : " New Block Encryption Algorithm MISTY ",
  Proceedings of forth international workshop of fast software encryption,
  Lecture Notes in Computer Science 1267, Springer Verlag（1997）（PDF:166KB）

弊社の無償化の方針は、この公開仕様をもとにみずから暗号アルゴリズム MISTY を搭載した製品を開発、事業化していただける企業、法人を対象としております。 なお、本件は、あくまで特許の無償実施許諾であり、これにより暗号アルゴリズム MISTY に関して当社が保有する技術情報や設計技術などをご提供するものではございませんのであらかじめ御含みおきいただきたく存じます。

無償契約では、上述のとおり、公開情報をもとにみずから MISTY 搭載製品を開発いただく形となります。ただし、弊社はすでに、MISTY のソフトウェアインプリメントを容易にするソフトウェア開発キットなどを製品化しております。ご希望によりましては、無償化のアプローチに代えこの開発キットなどの製品をご提供（有償）することも可能でございます。

また、無償実施許諾を申し込まれた場合でも、ご希望に添えない場合もございますので予めご了承下さい。また、暗号アルゴリズムの互換性に関しましては、公開情報に基づき、お客様の責任においてご確認いただきますようお願い申しあげます

となっています。

これを利用するに当り、次のように確認をしました。

次のような場合は、特許ライセンス契約 が必要となりますか？

1. misty.exe これは、独立して動くプログラム。 これは、ソースコードを
   公開し無料で使用可能とする。このソフトは misty のソースコードを利用している。
2. another.exe これは、misty.exe を system 関数で呼び出し、データの暗号化を行う。
   another.exe には misty のソースコードは含まれない。
   another.exe を有料のソフトウエアとして販売する。

この場合、2のソフト another.exe を販売するときは、特許ライセンス契約が必要になるのでしょうか？

回答は、

宇山様
三菱電機 ****です。 ご連絡ありがとうございます。
また、ご回答が遅くなり申し訳ございませんでした。
弊知財部門に確認をした結果、ご回答申し上げます。

弊社としては、MISTY 暗号アルゴリズムが広く普及することを
望む立場でございます。
基本特許を無償開放させていただいたのもその趣旨でありますので、
貴方個人にてオープンソースとして配布される範囲では、
弊社ウェブサイトにて公開しております契約内容をご一読いただき、
特段の問題がなければ、そのままご利用いただくことで問題ございません。
上記以外のケースで、もし書面での契約書締結をご希望される場合には、
別途ライセンス部門よりご連絡差し上げますので、
お知らせいただければ幸いです。
よろしくお願い申し上げます。

ということでしたので、ありがたく利用させていただきます。

比較しやすいように、3種類のソースコードを掲載いたします。

1． 本来の、Misty のソースコード
2． MistyEC のソースコード
3． MistyDC のソースコード
このうち、2，3についての二次著作権の主張はいたしません。ご自由に変更して下さい。ただし、自作のソフトの中にソースコードを組み込む場合には、三菱電機へ問い合わせて確認をして下さい。また鍵の長さが56ビットを超える場合には、貿易管理令についてもご確認下さい。

1． 本来の、Misty のソースコード

つぎのものが、MISTYのソースコードです。

ブロック暗号アルゴリズムＭＩＳＴＹ
松井充　氏　　（三菱電機株式会社、情報技術総合研究所）によるものです。

Sample Programs of MISTY1 in C Language
Version 1.00 July 22 1996
Mitsubishi Electric Corporation
This document shows two sample programs of MISTY1 with eight rounds. The _rst one is a main program, which calls
the second program. The second one is a core logic of MISTY1 algorithm; It encrypts and decrypts data stream with
arbitrary number of blocks in ECB mode using given 128-bit key. Note: due to space constraint, the second program
is written in twocolumn format.

```
/*****************************************************
* *
* A Sample Main C Program of MISTY1 Algorithm *
* *
* Language : Highly Portable C Language *
* Coding by : Mitsuru Matsui / 22 July 1996 *
* Copyright : Mitsubishi Electric Corporation *
* *
*****************************************************/
typedef unsigned char uchar;
typedef unsigned short ushort;
extern ushort EXTKEY[4][8];
main()
{
static uchar text[16] = {0x01,0x23,0x45,0x67,0x89,0xab,0xcd,0xef,
0xfe,0xdc,0xba,0x98,0x76,0x54,0x32,0x10};
static uchar key[16] = {0x00,0x11,0x22,0x33,0x44,0x55,0x66,0x77,
0x88,0x99,0xaa,0xbb,0xcc,0xdd,0xee,0xff};
int i;
printf( "Secret Key " );
for( i=0; i<16; i++ ) printf( "%02x ", key[i] );
printf( "¥n" );
printf( "Plaintext " );
for( i=0; i<16; i++ ) printf( "%02x ", text[i] );
printf( "¥n" );
misty1( text, key, 2, 0 );
printf( "Extended Key " );
```

```
for( i=0; i<8; i++ ) printf( "%02x %02x ", (uchar)(EXTKEY[1][i]>>8), (uchar)(EXTKEY[1][i]&0xff) );
printf( "¥n" );
printf( "Ciphertext " );
for( i=0; i<16; i++ ) printf( "%02x ", text[i] );
printf( "¥n" );
misty1( text, key, 2, 1 );
printf( "Plaintext " );
for( i=0; i<16; i++ ) printf( "%02x ", text[i] );
printf( "¥n" );
}



/*************************************************
* *
* MISTY1 Block Cipher Algorithm (8-round/ECB) *
* *
* Language : Highly Portable C Language *
* Coding by : Mitsuru Matsui / 22 July 1996 *
* Copyright : Mitsubishi Electric Coporation *
* *
*************************************************/
typedef unsigned short ushort;
typedef unsigned char uchar;
ushort EXTKEY[4][8];
static uchar S7[128] = {
27, 50, 51, 90, 59, 16, 23, 84, 91, 26,114,115,107, 44,102, 73,
31, 36, 19,108, 55, 46, 63, 74, 93, 15, 64, 86, 37, 81, 28, 4,
11, 70, 32, 13,123, 53, 68, 66, 43, 30, 65, 20, 75,121, 21,111,
14, 85, 9, 54,116, 12,103, 83, 40, 10,126, 56, 2, 7, 96, 41,
25, 18,101, 47, 48, 57, 8,104, 95,120, 42, 76,100, 69,117, 61,
89, 72, 3, 87,124, 79, 98, 60, 29, 33, 94, 39,106,112, 77, 58,
1,109,110, 99, 24,119, 35, 5, 38,118, 0, 49, 45,122,127, 97,
80, 34, 17, 6, 71, 22, 82, 78,113, 62,105, 67, 52, 92, 88,125 };
static ushort S9[512] = {
451,203,339,415,483,233,251, 53,385,185,279,491,307, 9, 45,211,
199,330, 55,126,235,356,403,472,163,286, 85, 44, 29,418,355,280,
331,338,466, 15, 43, 48,314,229,273,312,398, 99,227,200,500, 27,
1,157,248,416,365,499, 28,326,125,209,130,490,387,301,244,414,
467,221,482,296,480,236, 89,145, 17,303, 38,220,176,396,271,503,
231,364,182,249,216,337,257,332,259,184,340,299,430, 23,113, 12,
71, 88,127,420,308,297,132,349,413,434,419, 72,124, 81,458, 35,
317,423,357, 59, 66,218,402,206,193,107,159,497,300,388,250,406,
481,361,381, 49,384,266,148,474,390,318,284, 96,373,463,103,281,
101,104,153,336, 8, 7,380,183, 36, 25,222,295,219,228,425, 82,
265,144,412,449, 40,435,309,362,374,223,485,392,197,366,478,433,
195,479, 54,238,494,240,147, 73,154,438,105,129,293, 11, 94,180,
329,455,372, 62,315,439,142,454,174, 16,149,495, 78,242,509,133,
253,246,160,367,131,138,342,155,316,263,359,152,464,489, 3,510,
189,290,137,210,399, 18, 51,106,322,237,368,283,226,335,344,305,
327, 93,275,461,121,353,421,377,158,436,204, 34,306, 26,232, 4,
391,493,407, 57,447,471, 39,395,198,156,208,334,108, 52,498,110,
```

```
202, 37,186,401,254, 19,262, 47,429,370,475,192,267,470,245,492,
269,118,276,427,117,268,484,345, 84,287, 75,196,446,247, 41,164,
14,496,119, 77,378,134,139,179,369,191,270,260,151,347,352,360,
215,187,102,462,252,146,453,111, 22, 74,161,313,175,241,400, 10,
426,323,379, 86,397,358,212,507,333,404,410,135,504,291,167,440,
321, 60,505,320, 42,341,282,417,408,213,294,431, 97,302,343,476,
114,394,170,150,277,239, 69,123,141,325, 83, 95,376,178, 46, 32,
469, 63,457,487,428, 68, 56, 20,177,363,171,181, 90,386,456,468,
24,375,100,207,109,256,409,304,346, 5,288,443,445,224, 79,214,
319,452,298, 21, 6,255,411,166, 67,136, 80,351,488,289,115,382,
188,194,201,371,393,501,116,460,486,424,405, 31, 65, 13,442, 50,
61,465,128,168, 87,441,354,328,217,261, 98,122, 33,511,274,264,
448,169,285,432,422,205,243, 92,258, 91,473,324,502,173,165, 58,
459,310,383, 70,225, 30,477,230,311,506,389,140,143, 64,437,190,
120, 0,172,272,350,292, 2,444,162,234,112,508,278,348, 76,450 };
#define FL_enc( k ){¥
r1 ^= r0 & EXTKEY[0][k];¥
r3 ^= r2 & EXTKEY[1][(k+2)&7];¥
r0 ^= r1 | EXTKEY[1][(k+6)&7];¥
r2 ^= r3 | EXTKEY[0][(k+4)&7];¥
}
#define FL_dec( k ){¥
r0 ^= r1 | EXTKEY[0][(k+4)&7];¥
r2 ^= r3 | EXTKEY[1][(k+6)&7];¥
r1 ^= r0 & EXTKEY[1][(k+2)&7];¥
r3 ^= r2 & EXTKEY[0][k];¥
}
#define FI_key( k ){¥
r0 = EXTKEY[0][k] >> 7;¥
r1 = EXTKEY[0][k] & 0x7f;¥
r0 = S9[r0] ^ r1;¥
r1 = S7[r1] ^ ( r0 & 0x7f );¥
r1 ^= EXTKEY[0][(k+1)&7] >> 9;¥
r0 ^= EXTKEY[0][(k+1)&7] & 0x1ff;¥
r0 = S9[r0] ^ r1;¥
EXTKEY[3][k] = r1;¥
EXTKEY[2][k] = r0;¥
EXTKEY[1][k] = r1 << 9 ^ r0;¥
}
#define FI_txt( a0, a1, k ){¥
a1 = a0 >> 7;¥
a0 &= 0x7f;¥
a1 = S9[a1] ^ a0;¥
a0 = S7[a0] ^ a1;¥
a1 ^= EXTKEY[2][k];¥
a0 ^= EXTKEY[3][k];¥
a0 &= 0x7f;¥
a1 = S9[a1] ^ a0;¥
a1 ^= a0 << 9;¥
}
#define FO_txt( a0, a1, a2, a3, k ){¥
t0 = a0 ^ EXTKEY[0][k];¥
```

```
FI_txt( t0, t1, (k+5)&7 );¥
t1 ^= a1;¥
t2 = a1 ^ EXTKEY[0][(k+2)&7];¥
FI_txt( t2, t0, (k+1)&7 );¥
t0 ^= t1;¥
t1 ^= EXTKEY[0][(k+7)&7];¥
FI_txt( t1, t2, (k+3)&7 );¥
t2 ^= t0;¥
t0 ^= EXTKEY[0][(k+4)&7];¥
a2 ^= t0;¥
a3 ^= t2;¥
}



/***********************************************
* *
* Encryption/Decryption Subroutine Body *
* *
* misty1( text, key, block, mode ) *
* *
* text : plain/ciphertext address I/O *
* key : secret-key address I *
* block : number of text blocks I *
* mode : 0:encryption 1:decryption I *
* *
***********************************************/
misty1( text, key, block, mode )
uchar *text,*key;
int block,mode;
{
register ushort t0, t1, t2;
register ushort r0, r1, r2, r3;
/*** Key Scheduling ***/
EXTKEY[0][0] = (ushort)key[0]<<8 ^ (ushort)key[1];
EXTKEY[0][1] = (ushort)key[2]<<8 ^ (ushort)key[3];
EXTKEY[0][2] = (ushort)key[4]<<8 ^ (ushort)key[5];
EXTKEY[0][3] = (ushort)key[6]<<8 ^ (ushort)key[7];
EXTKEY[0][4] = (ushort)key[8]<<8 ^ (ushort)key[9];
EXTKEY[0][5] = (ushort)key[10]<<8 ^ (ushort)key[11];
EXTKEY[0][6] = (ushort)key[12]<<8 ^ (ushort)key[13];
EXTKEY[0][7] = (ushort)key[14]<<8 ^ (ushort)key[15];
FI_key( 0 );
FI_key( 1 );
FI_key( 2 );
FI_key( 3 );
FI_key( 4 );
FI_key( 5 );
FI_key( 6 );
FI_key( 7 );
/*** Data Randomizing ***/
if( !(mode & 1) ){
```

```
/*** Encryption ***/
while( block-- > 0 ){
r0 = (ushort)text[0]<<8 ^ (ushort)text[1];
r1 = (ushort)text[2]<<8 ^ (ushort)text[3];
r2 = (ushort)text[4]<<8 ^ (ushort)text[5];
r3 = (ushort)text[6]<<8 ^ (ushort)text[7];
FL_enc( 0 );
FO_txt( r0, r1, r2, r3, 0 );
FO_txt( r2, r3, r0, r1, 1 );
FL_enc( 1 );
FO_txt( r0, r1, r2, r3, 2 );
FO_txt( r2, r3, r0, r1, 3 );
FL_enc( 2 );
FO_txt( r0, r1, r2, r3, 4 );
FO_txt( r2, r3, r0, r1, 5 );
FL_enc( 3 );
FO_txt( r0, r1, r2, r3, 6 );
FO_txt( r2, r3, r0, r1, 7 );
FL_enc( 4 );
text[0] = r2 >> 8;
text[1] = r2 & 0xff;
text[2] = r3 >> 8;
text[3] = r3 & 0xff;
text[4] = r0 >> 8;
text[5] = r0 & 0xff;
text[6] = r1 >> 8;
text[7] = r1 & 0xff;
text += 8;
}
}
else{
/*** Decryption ***/
while( block-- > 0 ){
r0 = (ushort)text[0]<<8 ^ (ushort)text[1];
r1 = (ushort)text[2]<<8 ^ (ushort)text[3];
r2 = (ushort)text[4]<<8 ^ (ushort)text[5];
r3 = (ushort)text[6]<<8 ^ (ushort)text[7];
FL_dec( 4 );
FO_txt( r0, r1, r2, r3, 7 );
FO_txt( r2, r3, r0, r1, 6 );
FL_dec( 3 );
FO_txt( r0, r1, r2, r3, 5 );
FO_txt( r2, r3, r0, r1, 4 );
FL_dec( 2 );
FO_txt( r0, r1, r2, r3, 3 );
FO_txt( r2, r3, r0, r1, 2 );
FL_dec( 1 );
FO_txt( r0, r1, r2, r3, 1 );
FO_txt( r2, r3, r0, r1, 0 );
FL_dec( 0 );
text[0] = r2 >> 8;
text[1] = r2 & 0xff;
```

```
text[2] = r3 >> 8;
text[3] = r3 & 0xff;
text[4] = r0 >> 8;
text[5] = r0 & 0xff;
text[6] = r1 >> 8;
text[7] = r1 & 0xff;
text += 8;
}
}
}
```

2. MistyEC のソースコード

ECMisty.cpp

```
////////////////////////////////////////////////////////////////////////
// ECMisty.cpp : コンソールアプリケーション用のエントリポイントの定義
//
#include "stdafx.h"

#include <iostream>
#include <string.h>
#include <stdlib.h>
#include "ECMisty.h"
#include <stdio.h>

using namespace std;

FILE *stream;
FILE *stream1;
FILE *stream2;

extern ushort EXTKEY[4][8];

// 暗号文のHEX表示用
char toChar( int c )
{
        if( c >= 0 && c <= 9 )          // 0～ならば
                return char( c + 0x30 ); // ASCIIに変換して返す
        else if( c >= 10 && c <= 15 )   // 10～ならば
                return char( c + 0x37 ); // A～FのASCIIを返す
        else
                return ' ';
}

// メイン
int main(                               // 正常終了時:0、異常時:1
                int argc,               // 引数の数
                char** argv )           // 引数へのポインタ
```

```
{
        int i;
        int block;
        int mode;
        int numclosed;
        char j,k;

        printf( "Start!¥n" );

          // 引数チェック
        if( argc != 4 ){                              // 使い方の誤り
                printf( "引数の数が不正ですから異常終了します。¥n" );
                exit(1);
        }

   /* 鍵ファイルを開く*/
   if( (stream  = fopen( argv[1], "rb" )) == NULL )
      printf( "Can not open Key file.¥n" );
//   else
  //    printf( "File 'eckeym0.bin' was opened.¥n" );

  /* 平文を読み出すファイルを開く*/
   if( (stream1  = fopen( argv[2], "rb" )) == NULL )
      printf( "Can not open Plane Text file.¥n");
//   else
  //    printf( "File 'encrypt0.bin' was opemed.¥n" );

   /* 暗号文を書き込むファイルを開く*/
   if( (stream2 = fopen( argv[3], "wb" )) == NULL )
      printf( "Can not open for Encrypted file.¥n" );
//   else
  //    printf( "File 'encrypt1.bin' was opened.¥n" );

        unsigned char key2[64],key[32];

        fseek(stream,0,0);
        fscanf(stream,"%s",key2);

        for(i=0; i<16; i++){
                j = *(key2+2*i);
                k = *(key2+2*i+1);
                if(j>=0x30 && j<=0x39) j = j-0x30;
                else{
                        if(j>=0x41 && j<=0x46) j = j-0x41+0x0A;
                }
                if(k>=0x30 && k<=0x39) k = k-0x30;
                else{
                        if(k>=0x41 && k<=0x46) k = k-0x41+0x0A;
                }
                key[i] = j*0x10 + k;
        }
        key[16] = NULL;
```

```
        int cnt;
        long mesLength; // 平文長（バイト）
        int c;
        char* bufp;

// 平文
        fseek(stream1, 0, SEEK_END);
        long filelen = ftell(stream1);
        fseek(stream1,0,0);
        int head = sizeof(long);
        mesLength = filelen + head;

        //暗号化
        if(mesLength <= 1024){
                block = mesLength/8 + ((mesLength%8)?1:0);
                bufp = (char*)new(char[block*8 + 1]);
                if(bufp == NULL){
                        printf("メモリ不足です。異常終了します。¥r¥n");
                        return(-1);
                        }
                (*(long*)(bufp)) = filelen;

        // 平文
                fseek(stream1, 0, 0);
                i = head;
                do{
                        c = fgetc(stream1);
                        bufp[i]=c;
                        i=i+1;
                }while(c!=EOF);
                bufp[i-1]=NULL;

  // 暗号化実行
                mode = 0;
                misty1( (unsigned char*)bufp, (unsigned char*)key,  block,  mode);
        // 暗号文を書き込む
                for( cnt = 0; cnt<block*8; cnt++ ){
                        fwrite( &(bufp[cnt]),sizeof(char),1, stream2);
                }
        }
        else{
                block = mesLength/8 + ((mesLength%8)?1:0);
                bufp = (char*)new(char[1024 + 2]);
                if(bufp == NULL){
                        printf("メモリ不足¥r¥n");
                        return(-1);
                        }
                (*(long*)(bufp)) = filelen;

                long rBlen = block;
                int r = 0;
```

```
                do{
                // 平文

                        if(r == 0){
                                i = head;
                                fseek(stream1, r*1024, 0);
                        }
                        if(r > 0 ){
                                i = 0;
                                fseek(stream1, r*1024-4, 0);
                        }
                        do{
                                c = fgetc(stream1);
                                bufp[i]=c;
                                i=i+1;
                        }while((c!=EOF) && (i<=1024));
                        bufp[i-1]=NULL;

                        if(rBlen >= 1024/8){ block = 1024/8; }
                        if(rBlen < 1024/8){ block = rBlen;}

                // 暗号化実行
                        mode = 0;
                        misty1( (unsigned char*)bufp, (unsigned char*)key,  block,  mode);
        // 暗号文を書き込む
                        for( cnt = 0; cnt<block*8; cnt++ ){
                                fwrite( &(bufp[cnt]),sizeof(char),1, stream2);
                        }
                        r += 1;
                        rBlen -= 1024/8;
                }while(rBlen>0);
        }

        if( fclose( stream1 ) )
        printf( "Can not close Plane Text file.¥n" );
        /* ストリームを閉じる*/
        if( fclose( stream2 ) )
                printf( "Can not close Encrypted file.¥n" );
   /* 他のすべてのファイルを閉じる*/
        numclosed = _fcloseall( );
        delete[] bufp;
        printf( "End!¥n" );
        return 0;
}
```

Misty1.cpp

```
////////////////////////////////////////////////////////////////////////
// Misty1.cpp : コンソールアプリケーション用のエントリポイントの定義
//
```

```
#include "stdafx.h"

typedef unsigned char uchar;
typedef unsigned short ushort;

ushort EXTKEY[4][8];

static uchar S7[128] = {
        27,50,51,90,59,16,23,84,91,26,114,115,107,44,102,73,
        31,36,19,108,55,46,63,74,93,15,64,86,37,81,26,4,
        11,70,32,13,123,53,68,66,43,30,65,20,75,121,21,111,
        14,85,9,54,116,12,103,83,40,10,126,56,2,7,96,41,
        25,18,101,47,48,57,8,104,95,120,42,76,100,69,117,61,
        89,72,3,87,124,79,98,60,29,33,94,39,106,112,77,58,
        1,109,110,99,24,119,35,5,38,118,0,49,45,122,127,97,
        80,34,17,6,71,22,82,78,113,62,105,67,52,92,88,125};

static ushort S9[512] = {
        451,203,339,415,483,233,251,53,385,185,279,491,307,9,45,211,
        199,330,55,126,235,356,403,472,163,286,85,44,29,418,355,280,
        331,338,466,15,43,48,314,229,273,312,398,99,227,200,500,27,
        1,157,248,416,365,499,28,326,125,209,130,490,387,301,244,414,
        467,221,482,296,480,236,89,145,17,303,38,220,176,396,271,503,
        231,364,182,249,216,337,257,332,259,184,340,299,430,23,113,12,
        71,88,127,420,308,297,132,349,413,434,419,72,124,81,458,35,
        317,423,357,59,66,218,402,206,193,107,159,497,300,388,250,406,
        481,361,381,49,384,266,148,474,390,318,284,96,373,463,103,281,
        101,104,153,336,8,7,380,183,36,25,222,295,219,228,425,82,
        265,144,412,449,40,435,309,362,374,223,485,392,197,366,478,433,
        195,479,54,238,494,240,147,73,154,438,105,129,293,11,94,180,
        329,455,372,62,315,439,142,454,174,16,149,495,78,242,509,133,
        253,246,160,367,131,138,342,155,316,263,359,152,464,489,3,510,
        189,290,137,210,399,18,51,106,322,237,368,283,226,335,344,305,
        327,93,275,461,121,353,421,377,158,436,204,34,306,26,232,4,
        391,493,407,57,447,471,39,395,198,156,208,334,108,52,498,110,
        202,37,186,401,254,19,262,47,429,370,475,192,267,470,245,492,
        269,118,276,427,117,268,484,345,84,287,75,196,446,247,41,164,
        14,496,119,77,378,134,139,179,369,191,270,260,151,347,352,360,
        215,187,102,462,252,146,453,111,22,74,161,313,175,241,400,10,
        426,323,379,86,397,358,212,507,333,404,410,135,504,291,167,440,
        321,60,505,320,42,341,282,417,408,213,294,431,97,302,343,476,
        114,394,170,150,277,239,69,123,141,325,83,95,376,178,46,32,
        469,63,457,487,428,68,56,20,177,363,171,181,90,386,456,468,
        24,375,100,207,109,256,409,304,346,5,288,443,445,224,79,214,
        319,452,298,21,6,255,411,166,67,136,80,351,488,289,115,382,
        188,194,201,371,393,501,116,460,486,424,405,31,65,13,442,50,
        61,465,128,168,87,441,354,328,217,261,98,122,33,511,274,264,
        448,169,285,432,422,205,243,92,258,91,473,324,502,173,165,58,
        459,310,383,70,225,30,477,230,311,506,389,140,143,64,437,190,
        120,0,172,272,350,292,2,444,162,234,112,508,278,348,76,450 };
```

```
#define FL_enc(k) {¥
        r1 ^= r0 & EXTKEY[0][k];¥
        r3 ^= r2 & EXTKEY[1][(k+2)&7];¥
        r0 ^= r1 | EXTKEY[1][(k+6)&7];¥
        r2 ^= r3 | EXTKEY[0][(k+4)&7];¥
        }

#define FL_dec(k) {¥
        r0 ^= r1 | EXTKEY[0][(k+4)&7];¥
        r2 ^= r3 | EXTKEY[1][(k+6)&7];¥
        r1 ^= r0 & EXTKEY[1][(k+2)&7];¥
        r3 ^= r2 & EXTKEY[0][k];¥
        }

#define FI_key(k) {¥
        r0 = EXTKEY[0][k] >> 7;¥
        r1 = EXTKEY[0][k] & 0x7f;¥
        r0 = S9[r0] ^ r1;¥
        r1 = S7[r1] ^ (r0 & 0x7f);¥
        r1 ^= EXTKEY[0][(k+1)&7] >> 9;¥
        r0 ^= EXTKEY[0][(k+1)&7] & 0x1ff;¥
        r0 = S9[r0] ^ r1;¥
        EXTKEY[3][k] = r1;¥
        EXTKEY[2][k] = r0;¥
        EXTKEY[1][k] = r1 << 9 ^ r0;¥
        }

#define FI_txt(a0, a1, k) {¥
        a1 = a0 >> 7;¥
        a0 &= 0x7f;¥
        a1 = S9[a1] ^ a0;¥
        a0 = S7[a0] ^ a1;¥
        a1 ^= EXTKEY[2][k];¥
        a0 ^= EXTKEY[3][k];¥
        a0 &= 0x7f;¥
        a1 = S9[a1] ^ a0;¥
        a1 ^= a0 << 9;¥
        }

#define FO_txt(a0, a1, a2, a3, k) {¥
        t0 = a0 ^ EXTKEY[0][k];¥
        FI_txt(t0, t1, (k+5)&7);¥
        t1 ^= a1;¥
        t2 = a1 ^ EXTKEY[0][(k+2)&7];¥
        FI_txt(t2, t0, (k+1)&7);¥
        t0 ^= t1;¥
        t1 ^= EXTKEY[0][(k+7)&7];¥
        FI_txt(t1, t2, (k+3)&7);¥
        t2 ^= t0;¥
        t0 ^= EXTKEY[0][(k+4)&7];¥
        a2 ^= t0;¥
        a3 ^= t2;¥
```

```
	}

void misty1(uchar* text, uchar* key, int block, int mode)
{
	register ushort t0, t1, t2;
	register ushort r0, r1, r2, r3;

	EXTKEY[0][0] = (ushort)key[0]<<8 ^ (ushort)key[1];
	EXTKEY[0][1] = (ushort)key[2]<<8 ^ (ushort)key[3];
	EXTKEY[0][2] = (ushort)key[4]<<8 ^ (ushort)key[5];
	EXTKEY[0][3] = (ushort)key[6]<<8 ^ (ushort)key[7];
	EXTKEY[0][4] = (ushort)key[8]<<8 ^ (ushort)key[9];
	EXTKEY[0][5] = (ushort)key[10]<<8 ^ (ushort)key[11];
	EXTKEY[0][6] = (ushort)key[12]<<8 ^ (ushort)key[13];
	EXTKEY[0][7] = (ushort)key[14]<<8 ^ (ushort)key[15];

	FI_key(0);
	FI_key(1);
	FI_key(2);
	FI_key(3);
	FI_key(4);
	FI_key(5);
	FI_key(6);
	FI_key(7);

	if(!(mode & 1)){
		while(block-- > 0){
			r0 = (ushort)text[0]<<8 ^ (ushort)text[1];
			r1 = (ushort)text[2]<<8 ^ (ushort)text[3];
			r2 = (ushort)text[4]<<8 ^ (ushort)text[5];
			r3 = (ushort)text[6]<<8 ^ (ushort)text[7];

			FL_enc(0);
			FO_txt(r0, r1, r2, r3, 0);
			FO_txt(r2, r3, r0, r1, 1);
			FL_enc(1);
			FO_txt(r0, r1, r2, r3, 2);
			FO_txt(r2, r3, r0, r1, 3);
			FL_enc(2);
			FO_txt(r0, r1, r2, r3, 4);
			FO_txt(r2, r3, r0, r1, 5);
			FL_enc(3);
			FO_txt(r0, r1, r2, r3, 6);
			FO_txt(r2, r3, r0, r1, 7);
			FL_enc(4);

			text[0] = r2 >> 8;
			text[1] = r2 & 0xff;
			text[2] = r3 >> 8;
			text[3] = r3 & 0xff;
			text[4] = r0 >> 8;
			text[5] = r0 & 0xff;
```

```
                        text[6] = r1 >> 8;
                        text[7] = r1 & 0xff;

                        text += 8;
                }
        }
        else{
                while(block-- > 0){
                        r0 = (ushort)text[0]<<8 ^ (ushort)text[1];
                        r1 = (ushort)text[2]<<8 ^ (ushort)text[3];
                        r2 = (ushort)text[4]<<8 ^ (ushort)text[5];
                        r3 = (ushort)text[6]<<8 ^ (ushort)text[7];

                        FL_dec(4);
                        FO_txt(r0, r1, r2, r3, 7);
                        FO_txt(r2, r3, r0, r1, 6);
                        FL_dec(3);
                        FO_txt(r0, r1, r2, r3, 5);
                        FO_txt(r2, r3, r0, r1, 4);
                        FL_dec(2);
                        FO_txt(r0, r1, r2, r3, 3);
                        FO_txt(r2, r3, r0, r1, 2);
                        FL_dec(1);
                        FO_txt(r0, r1, r2, r3, 1);
                        FO_txt(r2, r3, r0, r1, 0);
                        FL_dec(0);

                        text[0] = r2 >> 8;
                        text[1] = r2 & 0xff;
                        text[2] = r3 >> 8;
                        text[3] = r3 & 0xff;
                        text[4] = r0 >> 8;
                        text[5] = r0 & 0xff;
                        text[6] = r1 >> 8;
                        text[7] = r1 & 0xff;

                        text += 8;
                }
        }
}
```

Stdafx.cpp

```
////////////////////////////////////////////////////////////
// stdafx.cpp : 標準インクルードファイルを含むソースファイル
//              ECMisty.pch 生成されるプリコンパイル済ヘッダー
//              stdafx.obj 生成されるプリコンパイル済タイプ情報
#include "stdafx.h"
// TODO: STDAFX.H に含まれていて、このファイルに記述されていない
// ヘッダーファイルを追加してください。
```

以上、VC++2005 でソフトを作成するために必要なファイルです。

3. MistyDC のソースコード

DCMisty.cpp

```
////////////////////////////////////////////////////////////////////
// DCMisty.cpp : コンソールアプリケーション用のエントリポイントの定義
//

#include "stdafx.h"

#include <iostream>
#include <string.h>
#include <stdlib.h>
#include "DCMisty.h"
#include <stdio.h>

using namespace std;

FILE *stream;
FILE *stream1;
FILE *stream2;

extern ushort EXTKEY[4][8];

// 暗号文のHEX表示用
char toChar( int c )
{
        if( c >= 0 && c <= 9 )          // 0～ならば
                return char( c + 0x30 ); // ASCIIに変換して返す
        else if( c >= 10 && c <= 15 )   // 10～ならば
                return char( c + 0x37 ); // A～FのASCIIを返す
        else
                return ' ';
}

// メイン
int main(                               // 正常終了時:0、異常時:1
                int argc,               // 引数の数
                char** argv )           // 引数へのポインタ
{
        int i;
        int block;
        int mode;
        char j,k;
```

```
        printf( "Start!¥n" );

        // 引数チェック
        if( argc != 4 ){                        // 使い方の誤り
                exit(1);
        }

   int numclosed;

   /* 鍵ファイルを開く*/
   if( (stream  = fopen( argv[1], "rb" )) == NULL )
      printf( "Can not open Key file.¥n" );
//   else
  //    printf( "File 'eckeym0.bin' was opened.¥n" );

  /* 平文を書き込むファイルを開く*/
   if( (stream1  = fopen( argv[3], "wb" )) == NULL )
      printf( "Can not open file for Plane Text.¥n");
//   else
  //    printf( "File 'encrypt0.bin' was opemed.¥n" );

   /* 暗号文を読み出すファイルを開く*/
   if( (stream2 = fopen( argv[2], "rb" )) == NULL )
      printf( "Can not open Encrypted file.¥n" );
//   else
  //    printf( "File 'encrypt1.bin' was opened.¥n" );

        unsigned char key[32], key2[64];

        fseek(stream,0,0);
        fscanf(stream,"%s",key2);

        for(i=0; i<16; i++){
                j = *(key2+2*i);
                k = *(key2+2*i+1);
                if(j>=0x30 && j<=0x39) j = j-0x30;
                else{
                        if(j>=0x41 && j<=0x46) j = j-0x41+0x0A;
                }
                if(k>=0x30 && k<=0x39) k = k-0x30;
                else{
                        if(k>=0x41 && k<=0x46) k = k-0x41+0x0A;
                }
                key[i] = j*0x10 + k;
        }
        key[16] = NULL;

   // 暗号文
        fseek(stream2, 0, SEEK_END);
        long filelen = ftell(stream2);
        fseek(stream2,0,0);
        int head = sizeof(long);
```

```
	long mesLength = filelen;
	block = mesLength/8 + ((mesLength%8)?1:0) ;

	int cnt;
	long lenp;
	int c;
	char* bufp;

	if(mesLength <= 1024){
		block = mesLength/8 + ((mesLength%8)?1:0);
		bufp = (char*)new(char[block*8 + 2]);
		if(bufp == NULL){
			printf("メモリ不足¥r¥n");
			return(-1);
			}

	// 暗文
		fseek(stream2, 0, 0);
		i = 0;
		do{
			c = fgetc(stream2);
			bufp[i]=c;
			i=i+1;
		}while(c!=EOF);
		bufp[i-1]=NULL;

// 復号化実行
		mode = 1;
		misty1( (unsigned char*)bufp, key,  block,  mode);
	// 復号文を書き込む
		lenp = (*(long*)(bufp));
		bufp[lenp+head] = NULL;
		for( cnt = 0; cnt<lenp; cnt++ ){
			fwrite( &(bufp[cnt+head]),sizeof(char),1, stream1);
		}
	}
	else{
		block = mesLength/8 + ((mesLength%8)?1:0);
		bufp = (char*)new(char[1024 + 2]);
		if(bufp == NULL){
			printf("メモリ不足¥r¥n");
			return(-1);
			}
		long rBlen = block;
		int r = 0;
		do{
			// 暗文
			fseek(stream2, r*1024, 0);
			i = 0;
			do{
				c = fgetc(stream2);
				bufp[i]=c;
```

```
                        i=i+1;
                }while((c!=EOF) && (i<=1024));
                bufp[i-1]=NULL;

                if(rBlen >= 1024/8){ block = 1024/8; }
                if(rBlen < 1024/8){ block = rBlen;}

        // 復号化実行
                mode = 1;
                misty1( (unsigned char*)bufp, key,  block,  mode);
        // 復号文を書き込む
                if(r==0){
                        lenp = (*(long*)(bufp));
                        for( cnt = 0; cnt<(1024-head); cnt++ ){
                                fwrite( &(bufp[cnt+head]),sizeof(char),1, stream1);
                        }
                        lenp -= 1024-head;
                }
                else{
                        if(rBlen >= 1024/8){
                                for( cnt = 0; cnt<1024; cnt++ ){
                                        fwrite( &(bufp[cnt]),sizeof(char),1, stream1);
                                }
                                lenp -= 1024;
                        }
                        else{
                                for( cnt = 0; cnt<lenp; cnt++ ){
                                        fwrite( &(bufp[cnt]),sizeof(char),1, stream1);
                                }
                        }
                }
                r += 1;
                rBlen -= 1024/8;
        }while(rBlen>0);
    }

    if( fclose( stream1 ) )
    printf("平文用ファイルは閉じられませんでした。¥n");
    if( fclose( stream2 ) )
            printf("暗号文のファイルは閉じられませんでした。¥n");
  /* 他のすべてのファイルを閉じる*/
    numclosed = _fcloseall( );
    delete[] bufp;
    printf( "End!¥n" );
    return 0;
}
```

Misty1.cpp

```
////////////////////////////////////////////////////////////////////
```

```
// Misty1.cpp : コンソールアプリケーション用のエントリポイントの定義
//

#include "stdafx.h"

typedef unsigned char uchar;
typedef unsigned short ushort;

ushort EXTKEY[4][8];

static uchar S7[128] = {
        27,50,51,90,59,16,23,84,91,26,114,115,107,44,102,73,
        31,36,19,108,55,46,63,74,93,15,64,86,37,81,26,4,
        11,70,32,13,123,53,68,66,43,30,65,20,75,121,21,111,
        14,85,9,54,116,12,103,83,40,10,126,56,2,7,96,41,
        25,18,101,47,48,57,8,104,95,120,42,76,100,69,117,61,
        89,72,3,87,124,79,98,60,29,33,94,39,106,112,77,58,
        1,109,110,99,24,119,35,5,38,118,0,49,45,122,127,97,
        80,34,17,6,71,22,82,78,113,62,105,67,52,92,88,125};

static ushort S9[512] = {
        451,203,339,415,483,233,251,53,385,185,279,491,307,9,45,211,
        199,330,55,126,235,356,403,472,163,286,85,44,29,418,355,280,
        331,338,466,15,43,48,314,229,273,312,398,99,227,200,500,27,
        1,157,248,416,365,499,28,326,125,209,130,490,387,301,244,414,
        467,221,482,296,480,236,89,145,17,303,38,220,176,396,271,503,
        231,364,182,249,216,337,257,332,259,184,340,299,430,23,113,12,
        71,88,127,420,308,297,132,349,413,434,419,72,124,81,458,35,
        317,423,357,59,66,218,402,206,193,107,159,497,300,388,250,406,
        481,361,381,49,384,266,148,474,390,318,284,96,373,463,103,281,
        101,104,153,336,8,7,380,183,36,25,222,295,219,228,425,82,
        265,144,412,449,40,435,309,362,374,223,485,392,197,366,478,433,
        195,479,54,238,494,240,147,73,154,438,105,129,293,11,94,180,
        329,455,372,62,315,439,142,454,174,16,149,495,78,242,509,133,
        253,246,160,367,131,138,342,155,316,263,359,152,464,489,3,510,
        189,290,137,210,399,18,51,106,322,237,368,283,226,335,344,305,
        327,93,275,461,121,353,421,377,158,436,204,34,306,26,232,4,
        391,493,407,57,447,471,39,395,198,156,208,334,108,52,498,110,
        202,37,186,401,254,19,262,47,429,370,475,192,267,470,245,492,
        269,118,276,427,117,268,484,345,84,287,75,196,446,247,41,164,
        14,496,119,77,378,134,139,179,369,191,270,260,151,347,352,360,
        215,187,102,462,252,146,453,111,22,74,161,313,175,241,400,10,
        426,323,379,86,397,358,212,507,333,404,410,135,504,291,167,440,
        321,60,505,320,42,341,282,417,408,213,294,431,97,302,343,476,
        114,394,170,150,277,239,69,123,141,325,83,95,376,178,46,32,
        469,63,457,487,428,68,56,20,177,363,171,181,90,386,456,468,
        24,375,100,207,109,256,409,304,346,5,288,443,445,224,79,214,
        319,452,298,21,6,255,411,166,67,136,80,351,488,289,115,382,
        188,194,201,371,393,501,116,460,486,424,405,31,65,13,442,50,
        61,465,128,168,87,441,354,328,217,261,98,122,33,511,274,264,
        448,169,285,432,422,205,243,92,258,91,473,324,502,173,165,58,
        459,310,383,70,225,30,477,230,311,506,389,140,143,64,437,190,
```

```
        120,0,172,272,350,292,2,444,162,234,112,508,278,348,76,450 };

#define FL_enc(k) {¥
        r1 ^= r0 & EXTKEY[0][k];¥
        r3 ^= r2 & EXTKEY[1][(k+2)&7];¥
        r0 ^= r1 | EXTKEY[1][(k+6)&7];¥
        r2 ^= r3 | EXTKEY[0][(k+4)&7];¥
        }

#define FL_dec(k) {¥
        r0 ^= r1 | EXTKEY[0][(k+4)&7];¥
        r2 ^= r3 | EXTKEY[1][(k+6)&7];¥
        r1 ^= r0 & EXTKEY[1][(k+2)&7];¥
        r3 ^= r2 & EXTKEY[0][k];¥
        }

#define FI_key(k) {¥
        r0 = EXTKEY[0][k] >> 7;¥
        r1 = EXTKEY[0][k] & 0x7f;¥
        r0 = S9[r0] ^ r1;¥
        r1 = S7[r1] ^ (r0 & 0x7f);¥
        r1 ^= EXTKEY[0][(k+1)&7] >> 9;¥
        r0 ^= EXTKEY[0][(k+1)&7] & 0x1ff;¥
        r0 = S9[r0] ^ r1;¥
        EXTKEY[3][k] = r1;¥
        EXTKEY[2][k] = r0;¥
        EXTKEY[1][k] = r1 << 9 ^ r0;¥
        }

#define FI_txt(a0, a1, k) {¥
        a1 = a0 >> 7;¥
        a0 &= 0x7f;¥
        a1 = S9[a1] ^ a0;¥
        a0 = S7[a0] ^ a1;¥
        a1 ^= EXTKEY[2][k];¥
        a0 ^= EXTKEY[3][k];¥
        a0 &= 0x7f;¥
        a1 = S9[a1] ^ a0;¥
        a1 ^= a0 << 9;¥
        }

#define FO_txt(a0, a1, a2, a3, k) {¥
        t0 = a0 ^ EXTKEY[0][k];¥
        FI_txt(t0, t1, (k+5)&7);¥
        t1 ^= a1;¥
        t2 = a1 ^ EXTKEY[0][(k+2)&7];¥
        FI_txt(t2, t0, (k+1)&7);¥
        t0 ^= t1;¥
        t1 ^= EXTKEY[0][(k+7)&7];¥
        FI_txt(t1, t2, (k+3)&7);¥
        t2 ^= t0;¥
        t0 ^= EXTKEY[0][(k+4)&7];¥
```

```
        a2 ^= t0;¥
        a3 ^= t2;¥
        }

void misty1(uchar* text, uchar* key, int block, int mode)
{
        register ushort t0, t1, t2;
        register ushort r0, r1, r2, r3;

        EXTKEY[0][0] = (ushort)key[0]<<8 ^ (ushort)key[1];
        EXTKEY[0][1] = (ushort)key[2]<<8 ^ (ushort)key[3];
        EXTKEY[0][2] = (ushort)key[4]<<8 ^ (ushort)key[5];
        EXTKEY[0][3] = (ushort)key[6]<<8 ^ (ushort)key[7];
        EXTKEY[0][4] = (ushort)key[8]<<8 ^ (ushort)key[9];
        EXTKEY[0][5] = (ushort)key[10]<<8 ^ (ushort)key[11];
        EXTKEY[0][6] = (ushort)key[12]<<8 ^ (ushort)key[13];
        EXTKEY[0][7] = (ushort)key[14]<<8 ^ (ushort)key[15];

        FI_key(0);
        FI_key(1);
        FI_key(2);
        FI_key(3);
        FI_key(4);
        FI_key(5);
        FI_key(6);
        FI_key(7);

        if(!(mode & 1)){
                while(block-- > 0){
                        r0 = (ushort)text[0]<<8 ^ (ushort)text[1];
                        r1 = (ushort)text[2]<<8 ^ (ushort)text[3];
                        r2 = (ushort)text[4]<<8 ^ (ushort)text[5];
                        r3 = (ushort)text[6]<<8 ^ (ushort)text[7];

                        FL_enc(0);
                        FO_txt(r0, r1, r2, r3, 0);
                        FO_txt(r2, r3, r0, r1, 1);
                        FL_enc(1);
                        FO_txt(r0, r1, r2, r3, 2);
                        FO_txt(r2, r3, r0, r1, 3);
                        FL_enc(2);
                        FO_txt(r0, r1, r2, r3, 4);
                        FO_txt(r2, r3, r0, r1, 5);
                        FL_enc(3);
                        FO_txt(r0, r1, r2, r3, 6);
                        FO_txt(r2, r3, r0, r1, 7);
                        FL_enc(4);

                        text[0] = r2 >> 8;
                        text[1] = r2 & 0xff;
                        text[2] = r3 >> 8;
                        text[3] = r3 & 0xff;
```

```
                        text[4] = r0 >> 8;
                        text[5] = r0 & 0xff;
                        text[6] = r1 >> 8;
                        text[7] = r1 & 0xff;

                        text += 8;
                }
        }
        else{
                while(block-- > 0){
                        r0 = (ushort)text[0]<<8 ^ (ushort)text[1];
                        r1 = (ushort)text[2]<<8 ^ (ushort)text[3];
                        r2 = (ushort)text[4]<<8 ^ (ushort)text[5];
                        r3 = (ushort)text[6]<<8 ^ (ushort)text[7];

                        FL_dec(4);
                        FO_txt(r0, r1, r2, r3, 7);
                        FO_txt(r2, r3, r0, r1, 6);
                        FL_dec(3);
                        FO_txt(r0, r1, r2, r3, 5);
                        FO_txt(r2, r3, r0, r1, 4);
                        FL_dec(2);
                        FO_txt(r0, r1, r2, r3, 3);
                        FO_txt(r2, r3, r0, r1, 2);
                        FL_dec(1);
                        FO_txt(r0, r1, r2, r3, 1);
                        FO_txt(r2, r3, r0, r1, 0);
                        FL_dec(0);

                        text[0] = r2 >> 8;
                        text[1] = r2 & 0xff;
                        text[2] = r3 >> 8;
                        text[3] = r3 & 0xff;
                        text[4] = r0 >> 8;
                        text[5] = r0 & 0xff;
                        text[6] = r1 >> 8;
                        text[7] = r1 & 0xff;

                        text += 8;
                }
        }
}
```

Stdafx.cpp

```
// stdafx.cpp : 標準インクルードファイルを含むソースファイル
//              DCMisty.pch 生成されるプリコンパイル済ヘッダー
//              stdafx.obj 生成されるプリコンパイル済タイプ情報
#include "stdafx.h"
// TODO: STDAFX.H に含まれていて、このファイルに記述されていない
// ヘッダーファイルを追加してください。
```

以上、VC++2005 でソフトを作成するために必要なファイルです。

なお、stdafx.h は VC++2005 が自動的に作るファイルです。内容は以下のようなものです。

```
// stdafx.h : 標準のシステムインクルードファイル、
//            または参照回数が多く、かつあまり変更されない
//            プロジェクト専用のインクルードファイルを記述します。
//

#if !defined(AFX_STDAFX_H__FC68BA57_41BA_4DF5_8A81_43C5F748C33F__INCLUDED_)
#define AFX_STDAFX_H__FC68BA57_41BA_4DF5_8A81_43C5F748C33F__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000

#define WIN32_LEAN_AND_MEAN // Windows ヘッダーから殆ど使用されないスタッフを除外します

#include <stdio.h>

// TODO: プログラムで必要なヘッダー参照を追加してください。

//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ は前行の直前に追加の宣言を挿入します。

#endif // !defined(AFX_STDAFX_H__FC68BA57_41BA_4DF5_8A81_43C5F748C33F__INCLUDED_)
```

おわり。　　2012.04.08　　宇山靖政