Реферат

На тему

“Стеганограф”

Виконав студент групи КМ-502

Філь Володимир

Стеганографія — (з грец. στεγανός — прихований + γράφω — пишу) — тайнопис, при якому повідомлення, закодоване таким чином, що не виглядає як повідомлення — на відміну від криптографії. Таким чином непосвячена людина принципово не може розшифрувати повідомлення — бо не знає про факт його існування.

Якщо криптографія приховує зміст повідомлення, то стеганографія приховує сам факт існування повідомлення.

Методи

Розглядаючи програмні засоби захисту, доцільно спинитись на стеганографічних методах. Слово «стеганографія» означає приховане письмо, яке не дає можливості сторонній особі дізнатися про його існування. Одна з перших згадок про застосування тайнопису датується V століттям до н. е. Сучасним прикладом є випадок роздрукування на ЕОМ контрактів з малопомітними викривленнями обрисів окремих символів тексту — так вносилась шифрована інформація про умови складання контракту.

Комп'ютерна стеганографія базується на двох принципах. По-перше, аудіо- і відеофайли, а також файли з оцифрованими зображеннями можна деякою мірою змінити без втрати функціональності. По-друге, можливості людини розрізняти дрібні зміни кольору або звуку обмежені. Методи стеганографії дають можливість замінити несуттєві частки даних на конфіденційну інформацію. Сімейна цифрова фотографія може містити комерційну інформацію, а файл із записом сонати Гайдна — приватний лист.

Але найчастіше стеганографія використовується для створення цифрових водяних знаків. На відміну від звичайних їх можна нанести і відшукати тільки за допомогою спеціального програмного забезпечення — цифрові водяні знаки записуються як псевдовипадкові послідовності шумових сигналів, згенерованих на основі секретних ключів. Такі знаки можуть забезпечити автентичність або недоторканість документа, ідентифікувати автора або власника, перевірити права дистриб'ютора або користувача, навіть якщо файл був оброблений або спотворений.

Щодо впровадження засобів програмно-технічного захисту в ІС, розрізняють два основні його способи:

- додатковий захист — засоби захисту є доповненням до основних програмних і апаратних засобів комп'ютерної системи;

- вбудований захист — механізми захисту реалізуються у вигляді окремих компонентів ІС або розподілені за іншими компонентами системи.

Перший спосіб є гнучкішим, його механізми можна додавати і вилучати за потребою, але під час його реалізації можуть постати проблеми забезпечення сумісності засобів захисту між собою та з програмно-технічним комплексом ІС. Вмонтований захист вважається більш надійним і оптимальним, але є жорстким, оскільки в нього важко внести зміни. Таким доповненням характеристик способів захисту зумовлюється те, що в реальній системі їх комбінують.

Була розроблена програма для стенографічного шифрування зображень, тобто програма, яка буде одне зображення шифрувати в інше так, що у вихідному зображенні не можна було замітити зашифрованої інформації. В якості мови програмування було вибрано мову Сі. Додатково було використано файл “bmp.h”, розроблений Гарвардським університетом на основі файла від Microsoft, для зручної роботи з зображеннями.

Алгоритм роботи програми

Кодування

Код програми наведено у додатку 1.

Програма отримує 2 графічні файли у форматі bmp з однаковим розширенням, та назву вихідного файла. Відкриває їх. Далі для кожного рядка, для кожного пікселя:

- зчитується колір оригінального пікселя в шістнадцятковому форматі та переводиться в двійковий;

- зчитується колір пікселя, який потрібно приховати в шістнадцятковому форматі та переводиться в двійковий;

- останні два біти кольору оригінального пікселя заміняються першими двома бітами кольору пікселя, який потрібно приховати. Таким чином колір вихідного пікселя буде неможливо відрізнити від кольору оригінального пікселя;

- піксель записується у вихідний файл.

Після проходження усіх пік селів в останньому рядку програма закриває вхідні файли, зберігає і закриває вихідний файл. Код роб

Декодування

Код програми наведено у додатку 2.

Програма отримує графічний файл у форматі bmp, та назву вихідного файла. Відкриває їх. Далі для кожного рядка, для кожного пікселя:

- зчитується колір оригінального пікселя в шістнадцятковому форматі та переводиться в двійковий;

- останні два біти кольору оригінального пікселя стають першими двома бітами вихідного зображення.

Після проходження усіх пікселів в останньому рядку програма закриває вхідний файл, зберігає і закриває вихідний файл.

Приклад використання програми

Є 2 файли, 1.bmp – файл у який буде зашифровано інформацію(рис. 1), 2.bmp – файл, який буде зашифровано(рис.2).



Рис 1. – оригінальний файл

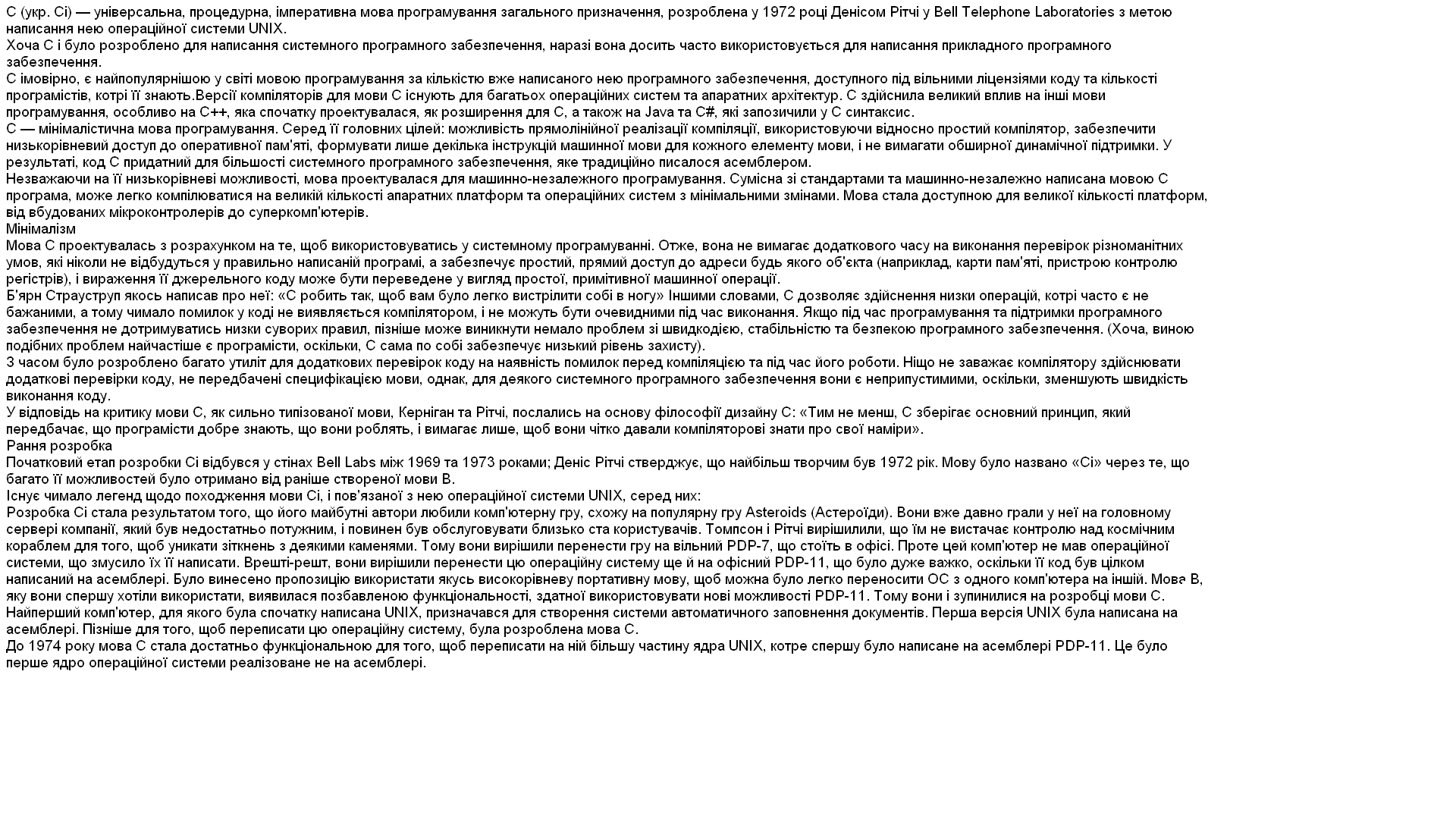


Рис 2. – Файл, що буде зашифровано

Кодування, використання програми: ./code 1.bmp 2.bmp output.bmp

Отримуємо файл output.bmp, показаний на рис.3.



Рис 3. – вихідний файл

Різницю між оригінальним та вихідним файлом не можливо помітити людським оком.

Декодування, використання програми: ./decode output.bmp decoded.bmp

Отримуємо файл decoded.bmp, показаний на рис.4.

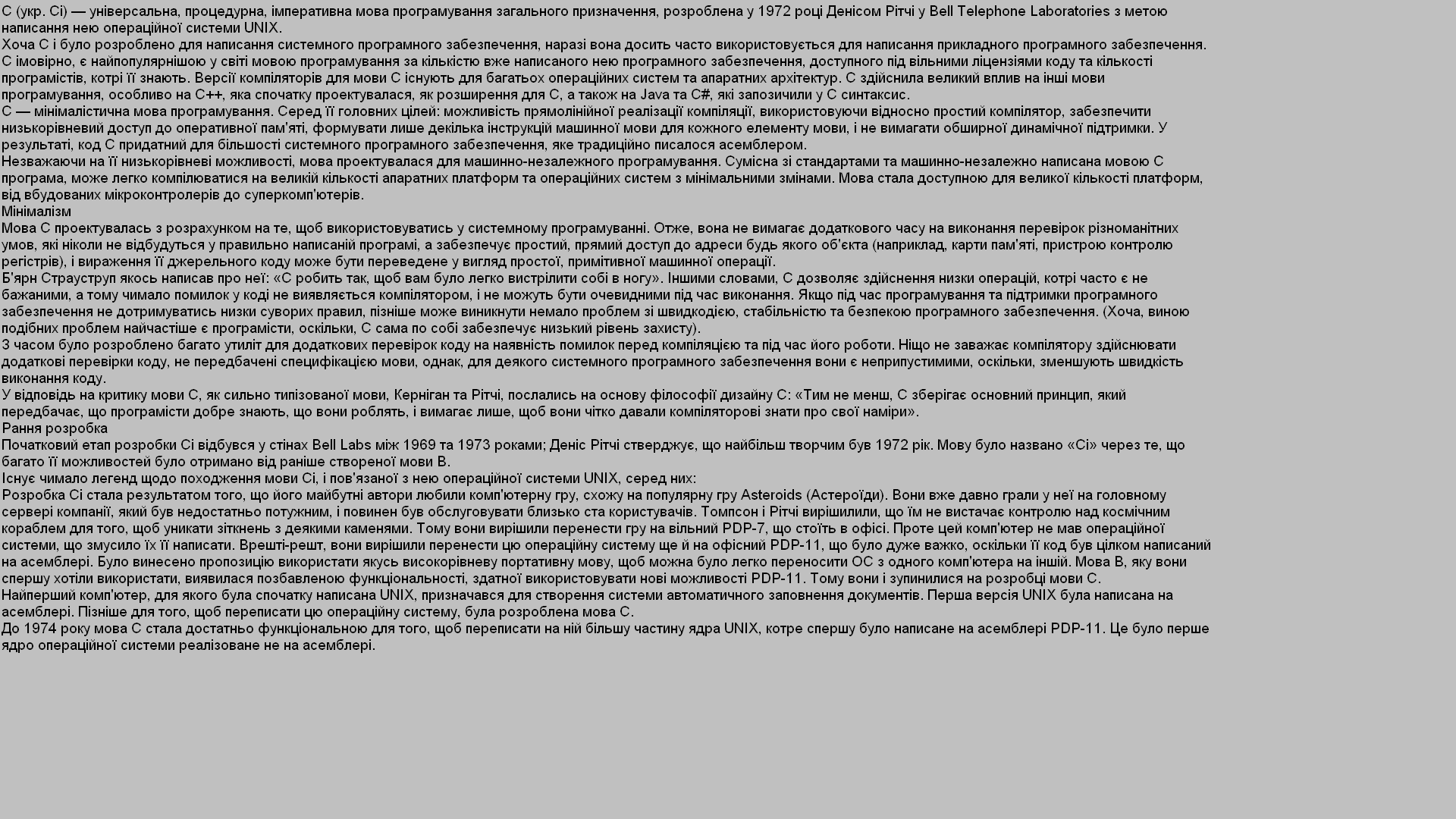


Рис. 4 – Розкодований файл

Сталася втрата білого кольору, але усю інформацію все ще можна прочитати.

Додаток 1. Код програми кодування

#include "bmp.h"

#include <stdio.h>

union toBiteArray;

struct bits;

union toBiteArray

{

BYTE \_byte;

struct

{

unsigned b1:1;

unsigned b2:1;

unsigned b3:1;

unsigned b4:1;

unsigned b5:1;

unsigned b6:1;

unsigned b7:1;

unsigned b8:1;

}bits;};

BYTE split(BYTE valueI, BYTE valueO);

RGBTRIPLE getCryptPix(RGBTRIPLE original, RGBTRIPLE crypte);

int main(int argc, char const \*argv[])

{

if (argc != 4)

{ printf("Використання: ./code infile file\_to\_hide outfile\n");

return 1;}

const char\* inFile = argv[1];

const char\* fileToHide = argv[2];

const char\* outFile = argv[3];

FILE\* inptr = fopen(inFile, "r");

if (inptr == NULL){

printf("Неможливо відкрити %s.\n", inFile);

return 2;}

FILE\* toHidePtr = fopen(fileToHide, "r");

if (toHidePtr == NULL){

printf("Неможливо відкрити %s.\n", fileToHide);

return 3;}

FILE\* outptr = fopen(outFile, "w");

if (outptr == NULL){

fclose(inptr);

fprintf(stderr, "Неможливо створити %s.\n", outFile);

return 3;}

BITMAPFILEHEADER bf;

fread(&bf, sizeof(BITMAPFILEHEADER), 1, inptr);

BITMAPINFOHEADER bi;

fread(&bi, sizeof(BITMAPINFOHEADER), 1, inptr);

int padding = (4 - (bi.biWidth \* sizeof(RGBTRIPLE)) % 4) % 4;

if (bf.bfType != 0x4d42 || bf.bfOffBits != 54 || bi.biSize != 40 ||

bi.biBitCount != 24 || bi.biCompression != 0{

fclose(outptr);

fclose(inptr);

fprintf(stderr, "Непідтримуваний формат файла.\n");

return 4;}

fwrite(&bf, sizeof(BITMAPFILEHEADER), 1, outptr);

fwrite(&bi, sizeof(BITMAPINFOHEADER), 1, outptr);

for(int i = 0; i <= bi.biWidth; i++){

for(int j = 0; j <= bi.biHeight; j++){

RGBTRIPLE tripleOriginal;

RGBTRIPLE tripleToCrypt;

fread(&tripleOriginal, sizeof(RGBTRIPLE), 1, inptr);

fread(&tripleToCrypt, sizeof(RGBTRIPLE), 1, toHidePtr);

tripleToCrypt = getCryptPix(tripleOriginal, tripleToCrypt);

fwrite(&tripleToCrypt, sizeof(RGBTRIPLE), 1, outptr);}}

return 0;

}

RGBTRIPLE getCryptPix(RGBTRIPLE original, RGBTRIPLE crypte){

RGBTRIPLE output;

output = original;

output.rgbtBlue = split(crypte.rgbtBlue, original.rgbtBlue);

output.rgbtGreen = split(crypte.rgbtGreen, original.rgbtGreen);

output.rgbtRed = split(crypte.rgbtRed, original.rgbtRed);

return output;

}

BYTE split(BYTE valueI, BYTE valueO){

union toBiteArray temp;

temp.\_byte = valueI;

if(valueI < 64) {

temp.bits.b1 = 0;

temp.bits.b2 = 0;}

else if(valueI < 128){

temp.bits.b1 = 0;

temp.bits.b2 = 1;}

else if(valueI < 192){

temp.bits.b1 = 1;

temp.bits.b2 = 0;}

else {

temp.bits.b1 = 1;

temp.bits.b2 = 1;}

union toBiteArray out0;

out0.\_byte = valueO;

out0.bits.b1 = temp.bits.b1;

out0.bits.b2 = temp.bits.b2;

return out0.\_byte;}

Додаток 2. Код програми декодування

#include "bmp.h"

#include <stdio.h>

union toBiteArray;

struct bits;

union toBiteArray

{

BYTE \_byte;

struct

{

unsigned b1:1;

unsigned b2:1;

unsigned b3:1;

unsigned b4:1;

unsigned b5:1;

unsigned b6:1;

unsigned b7:1;

unsigned b8:1;

}bits;

};

BYTE split(BYTE valueI);

RGBTRIPLE getCryptPix(RGBTRIPLE original);

int main(int argc, char const \*argv[])

{

if (argc != 3){

printf("Використання: ./decode infile outfile\n");

return 1;}

const char\* inFile = argv[1];

const char\* outFile = argv[2];

FILE\* inptr = fopen(inFile, "r");

if (inptr == NULL){

printf("Неможливо відкрити %s.\n", inFile);

return 2;}

FILE\* outptr = fopen(outFile, "w");

if (outptr == NULL){

fclose(inptr);

fprintf(stderr, "Неможливо створити %s.\n", outFile);

return 3;}

BITMAPFILEHEADER bf;

fread(&bf, sizeof(BITMAPFILEHEADER), 1, inptr);

BITMAPINFOHEADER bi;

fread(&bi, sizeof(BITMAPINFOHEADER), 1, inptr);

int padding = (4 - (bi.biWidth \* sizeof(RGBTRIPLE)) % 4) % 4;

if (bf.bfType != 0x4d42 || bf.bfOffBits != 54 || bi.biSize != 40 ||

bi.biBitCount != 24 || bi.biCompression != 0)

{

fclose(outptr);

fclose(inptr);

fprintf(stderr, "Непідтримуваний формат файла.\n");

return 4;}

fwrite(&bf, sizeof(BITMAPFILEHEADER), 1, outptr);

fwrite(&bi, sizeof(BITMAPINFOHEADER), 1, outptr);

for(int i = 0; i <= bi.biWidth; i++){

for(int j = 0; j <= bi.biHeight; j++){

RGBTRIPLE tripleOriginal;

fread(&tripleOriginal, sizeof(RGBTRIPLE), 1, inptr);

tripleOriginal = getCryptPix(tripleOriginal);

fwrite(&tripleOriginal, sizeof(RGBTRIPLE), 1, outptr);}}

return 0;}

RGBTRIPLE getCryptPix(RGBTRIPLE original){

RGBTRIPLE output;

output.rgbtBlue = split(original.rgbtBlue);

output.rgbtGreen = split(original.rgbtGreen);

output.rgbtRed = split(original.rgbtRed);

return output;}

BYTE split(BYTE valueI){

union toBiteArray temp;

temp.\_byte = valueI;

union toBiteArray out0;

out0.\_byte = 0x00;

out0.bits.b7 = temp.bits.b1;

out0.bits.b8 = temp.bits.b2;

return out0.\_byte;}

int\* toBinaryStr(BYTE value)

{

union toBiteArray temp;

temp.\_byte = value;

int ret[8];

ret[0] = temp.bits.b1;

ret[1] = temp.bits.b2;

ret[2] = temp.bits.b3;

ret[3] = temp.bits.b4;

ret[4] = temp.bits.b5;

ret[5] = temp.bits.b6;

ret[6] = temp.bits.b7;

ret[7] = temp.bits.b8;

return &ret;}