**Приложение A**

Исходный код приложения

// ------------------------------- main.c -----------------------------------

#include <windows.h>

#pragma pack(1)

#include "main.h"

#include "resource.h"

/\* function prototype declarations \*/

int InitApplication(HINSTANCE, char \*);

HWND InitInstance(HINSTANCE, int);

LRESULT CALLBACK WndProc(HWND, UINT, WPARAM, LPARAM);

int get\_os\_version(void);

int do\_read\_version(HWND, HANDLE, GETVERSIONINPARAMS \*);

int do\_read\_identify(HWND, HANDLE, char, BYTE \*);

int do\_read\_attribute(HWND, HANDLE, char, BYTE \*);

int get\_selected\_disc\_number(HWND);

BOOL fillingCombobox(HWND);

char\* getNormaliseDrive(int);

void clear\_values(HWND hWnd);

void swap\_byte(char \*, int);

/\*

\* @brief Entry point to the program

\*/

int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance,

PSTR szCmdLine, int iCmdShow) {

HWND hwnd;

MSG msg;

if (!hPrevInstance) {

/\* Register window class \*/

if (!InitApplication(hInstance, "MAIN")) {

MessageBox(HWND\_DESKTOP, "Error registering window class",

"Error", MB\_OK | MB\_ICONEXCLAMATION);

return FALSE;

}

}

if ((hwnd = InitInstance(hInstance, iCmdShow)) == NULL) {

MessageBox(HWND\_DESKTOP, "Failed to create window",

"Error", MB\_OK | MB\_ICONEXCLAMATION);

return FALSE;

}

/\* start message loop \*/

while (GetMessage(&msg, NULL, 0, 0)) {

if (!IsDialogMessage(hwnd, &msg)) {

TranslateMessage(&msg);

DispatchMessage(&msg);

}

}

return (int) msg.wParam;

}

/\*

\* @brief Register window class

\*/

int InitApplication(HINSTANCE hInstance, char \*class\_name) {

WNDCLASSEX wndclass;

wndclass.cbSize = sizeof (wndclass);

wndclass.style = CS\_HREDRAW | CS\_VREDRAW;

wndclass.lpfnWndProc = WndProc;

wndclass.cbClsExtra = 0;

wndclass.cbWndExtra = DLGWINDOWEXTRA;

wndclass.hInstance = hInstance;

wndclass.hIcon = LoadIcon(NULL, IDR\_MAINFRAME);

wndclass.hCursor = LoadCursor(NULL, IDC\_ARROW);

wndclass.hbrBackground = (HBRUSH) GetStockObject(DKGRAY\_BRUSH);

wndclass.lpszMenuName = NULL;

wndclass.lpszClassName = class\_name;

wndclass.hIconSm = LoadIcon(hInstance, IDR\_MAINFRAME);

return RegisterClassEx(&wndclass);

}

HWND InitInstance(HINSTANCE hInstance, int iCmdShow) {

HWND hWnd;

/\* Create the Window \*/

hWnd = CreateDialog(hInstance, "MAIN", 0, NULL);

if (hWnd == NULL)

return NULL;

ShowWindow(hWnd, iCmdShow);

UpdateWindow(hWnd);

return hWnd;

}

/\*

\* @brief Main Window Procedure

\*/

LRESULT CALLBACK WndProc(HWND hWnd, UINT msg, WPARAM wParam, LPARAM lParam) {

static GETVERSIONINPARAMS ver\_parm;

static BYTE identify[sizeof (SENDCMDOUTPARAMS) + SMART\_SIZE];

static BYTE attribute[sizeof (SENDCMDOUTPARAMS) + SMART\_SIZE];

HANDLE hDevice;

int v;

static int selectedDiscNumber;

switch (msg) {

case WM\_COMMAND: /\* Message when selecting the menu and buttons, etc. \*/

switch (LOWORD(wParam)) {

case IDOK: /\* Button "Refresh" \*/

selectedDiscNumber = get\_selected\_disc\_number(hWnd);

if ((v = get\_os\_version()) == 2) { /\* Win/NT/2K/XP/7 or old \*/

hDevice = CreateFile(

getNormaliseDrive(selectedDiscNumber),

GENERIC\_READ | GENERIC\_WRITE,

FILE\_SHARE\_READ | FILE\_SHARE\_WRITE,

NULL,

OPEN\_EXISTING, 0, NULL

);

} else {

break; /\* unknown or non support version \*/

}

if (hDevice == INVALID\_HANDLE\_VALUE) {

MessageBox(hWnd, "Can't open",

"Error", MB\_OK | MB\_ICONEXCLAMATION);

break;

}

if (do\_read\_version(hWnd, hDevice, &ver\_parm) != 0) {

CloseHandle(hDevice);

break;

}

if ((ver\_parm.bIDEDeviceMap <= 0) ||

((ver\_parm.fCapabilities & 0x04) == 0)) {

SetWindowText(GetDlgItem(hWnd, IDC\_STATIC2),

"Error");

break;

}

memset(identify, 0, sizeof (identify));

if (do\_read\_identify(hWnd, hDevice, selectedDiscNumber, identify) != 0) {

CloseHandle(hDevice);

break;

}

memset(attribute, 0, sizeof (attribute));

if (do\_read\_attribute(hWnd, hDevice, selectedDiscNumber, attribute) != 0) {

CloseHandle(hDevice);

break;

}

CloseHandle(hDevice);

break;

case IDCANCEL: /\* Exit \*/

PostMessage(hWnd, WM\_DESTROY, (WPARAM) 0, (LPARAM) 0);

break;

case IDABOUT: /\* About \*/

MessageBox(hWnd,

aboutInfo,

"About", MB\_OK | MB\_ICONINFORMATION);

break;

}

break;

case WM\_CREATE:

PostMessage(hWnd, (WM\_APP + 1), 0, 0);

break;

/\* Filling Combobox all supported devices \*/

case (WM\_APP + 1):

fillingCombobox(hWnd);

break;

case WM\_DESTROY:

PostQuitMessage(0);

break;

default:

return (LRESULT) DefWindowProc(hWnd, msg, wParam, lParam);

}

return (LRESULT) 0;

}

/\*

\* @brief Getting OS version

\* @return 1:Win95/98/98SE2/Me 2:WinNT/2K/XP/Vista/7 0:unknown

\*/

int get\_os\_version(void) {

OSVERSIONINFO ver;

memset((char \*) &ver, 0, sizeof (OSVERSIONINFO));

ver.dwOSVersionInfoSize = sizeof (OSVERSIONINFO);

if (GetVersionEx(&ver) == 0) {

return 0; /\* error \*/

}

if (ver.dwPlatformId >= VER\_PLATFORM\_WIN32\_NT) {

return 2; /\* Win/NT or old \*/

}

if (ver.dwPlatformId == VER\_PLATFORM\_WIN32\_WINDOWS) {

return 1; /\* Win95/98/98SE2/Me \*/

}

return 0; /\* unknown or Win32s \*/

}

/\*

\* @brief SMART\_GET\_VERSION

\* @return 0:no error 1:error

\*/

int do\_read\_version(HWND hwnd, HANDLE hdev, GETVERSIONINPARAMS \*vp) {

DWORD read\_byte;

char buffer[64];

memset(vp, 0, sizeof (GETVERSIONINPARAMS));

if (DeviceIoControl(hdev, SMART\_GET\_VERSION,

NULL,

0,

vp,

sizeof (GETVERSIONINPARAMS),

&read\_byte,

NULL) == 0) { /\* If an error occurred \*/

strcpy(buffer, "Program don't support this device!");

SetWindowText(GetDlgItem(hwnd, IDC\_STATIC1), buffer);

free(vp);

return 1;

}

sprintf(buffer, "%d", vp->bVersion);

SetWindowText(GetDlgItem(hwnd, IDC\_STATIC2), buffer);

sprintf(buffer, "%d", vp->bRevision);

SetWindowText(GetDlgItem(hwnd, IDC\_STATIC3), buffer);

sprintf(buffer, "0x%lx", vp->fCapabilities);

SetWindowText(GetDlgItem(hwnd, IDC\_STATIC4), buffer);

sprintf(buffer, "0x%x", vp->bIDEDeviceMap);

SetWindowText(GetDlgItem(hwnd, IDC\_STATIC5), buffer);

sprintf(buffer, "%d", read\_byte);

SetWindowText(GetDlgItem(hwnd, IDC\_STATIC6), buffer);

free(vp);

return 0;

}

/\*

\* @brief Display identify the disk

\* @return 0:no error 1:error

\*/

int do\_read\_identify(HWND hwnd, HANDLE hdev, char drvno, BYTE \*identify) {

PIDENTSECDATA ids;

SENDCMDINPARAMS scip;

DWORD read\_byte;

char buffer[64];

/\* Initialization SENDCMDINPARAMS \*/

scip.irDriveRegs.bFeaturesReg = 0;

scip.irDriveRegs.bSectorCountReg = 1;

scip.irDriveRegs.bSectorNumberReg = 1;

scip.irDriveRegs.bCylLowReg = 0;

scip.irDriveRegs.bCylHighReg = 0;

scip.irDriveRegs.bDriveHeadReg = 0xa0 | ((drvno & 1) << 4);

scip.irDriveRegs.bCommandReg = IDE\_ATA\_ID;

scip.bDriveNumber = drvno;

scip.cBufferSize = 1;

if (DeviceIoControl(hdev, SMART\_RECV\_DRIVE\_DATA,

(PVOID) & scip,

sizeof (SENDCMDINPARAMS) - 1,

(LPVOID) identify,

sizeof (SENDCMDOUTPARAMS) + SMART\_SIZE,

&read\_byte,

NULL) == 0) {

strcpy(buffer, "HDD does not support S.M.A.R.T.");

SetWindowText(GetDlgItem(hwnd, IDC\_STATIC1), buffer);

free(identify);

return 1;

}

ids = (PIDENTSECDATA) ((LPSENDCMDOUTPARAMS) identify)->bBuffer;

sprintf(buffer, "%s%d%s.", "HDD Drive", drvno, ": S.M.A.R.T. supported");

SetWindowText(GetDlgItem(hwnd, IDC\_STATIC1),buffer);

swap\_byte(ids->sModelNumber, sizeof (ids->sModelNumber));

memset(buffer, 0, sizeof (buffer));

strncpy(buffer, ids->sModelNumber, sizeof (ids->sModelNumber));

SetWindowText(GetDlgItem(hwnd, IDC\_STATIC7), buffer);

swap\_byte(ids->sFirmwareRev, sizeof (ids->sFirmwareRev));

memset(buffer, 0, sizeof (buffer));

strncpy(buffer, ids->sFirmwareRev, sizeof (ids->sFirmwareRev));

SetWindowText(GetDlgItem(hwnd, IDC\_STATIC8), buffer);

swap\_byte(ids->sSerialNumber, sizeof (ids->sSerialNumber));

memset(buffer, 0, sizeof (buffer));

strncpy(buffer, ids->sSerialNumber, sizeof (ids->sSerialNumber));

SetWindowText(GetDlgItem(hwnd, IDC\_STATIC9), buffer);

free(ids);

free(identify);

return 0;

}

/\*

\* @brief Read and display SMART attributes

\* @return 0:no error 1:error

\*/

int do\_read\_attribute(HWND hwnd, HANDLE hdev, char drvno, BYTE \*attr) {

SENDCMDINPARAMS scip;

LPDRVATTRIBUTE pda;

HWND hList; /\* ListBox\*/

DWORD read\_byte;

int i;

char buffer[128];

scip.irDriveRegs.bFeaturesReg = SMART\_READ\_ATTRIBUTE\_VALUES;

scip.irDriveRegs.bSectorCountReg = 1;

scip.irDriveRegs.bSectorNumberReg = 1;

scip.irDriveRegs.bCylLowReg = SMART\_CYL\_LOW;

scip.irDriveRegs.bCylHighReg = SMART\_CYL\_HI;

scip.irDriveRegs.bDriveHeadReg = 0xa0 | ((drvno & 1) << 4);

scip.irDriveRegs.bCommandReg = IDE\_EXEC\_SMART\_CMD;

scip.bDriveNumber = drvno;

scip.cBufferSize = 1;

if (DeviceIoControl(hdev, SMART\_RECV\_DRIVE\_DATA,

(PVOID) & scip,

sizeof (SENDCMDINPARAMS) - 1,

(LPVOID) attr,

sizeof (SENDCMDOUTPARAMS) + SMART\_SIZE,

&read\_byte,

NULL) == 0) {

strcpy(buffer, "Failed to get attributes");

SetWindowText(GetDlgItem(hwnd, IDC\_STATIC1), buffer);

return 1;

}

pda = (LPDRVATTRIBUTE)&(((LPSENDCMDOUTPARAMS) attr)->bBuffer[2]);

hList = GetDlgItem(hwnd, IDC\_LIST1);

SendMessage(hList, LB\_RESETCONTENT, 0, 0L);

SendMessage(hList, WM\_SETREDRAW, FALSE, 0);

for (i = 0; i < NUM\_ATTRIBUTE\_STRUCTS; i++) {

if (pda->bAttrID == 0) {

continue;

}

sprintf(buffer, "%2X%10d%7d%7d%7d%31s",

pda->bAttrID, pda->bRawVal[0],

pda->bAttrVal, pda->bWorstVal,

pda->wStatFlag, textAttrib[pda->bAttrID]);

SendMessage(hList, LB\_ADDSTRING, 0, (LPARAM) buffer);

pda++;

}

SendMessage(hList, WM\_SETREDRAW, TRUE, 0);

return 0;

}

/\*

\* @brief Read all supported devices and filling Combobox

\* @return TRUE:no error FALSE:error

\*/

BOOL fillingCombobox(HWND hwnd) {

HANDLE currentHandle;

BOOL resultValue = FALSE;

HWND handleComboBox = GetDlgItem(hwnd, IDC\_COMBO1);

SendMessage(handleComboBox, CB\_RESETCONTENT, 0, 0L);

GETVERSIONINPARAMS VP;

DWORD br;

char buffer[3];

memset((void\*) &VP, 0, sizeof (VP));

int i;

for (i = 0; i < 64; i++) {

if ((get\_os\_version()) == 2) {

currentHandle = CreateFile(

getNormaliseDrive(i),

GENERIC\_READ | GENERIC\_WRITE,

FILE\_SHARE\_READ | FILE\_SHARE\_WRITE,

NULL,

OPEN\_EXISTING, 0, NULL

);

if (currentHandle != INVALID\_HANDLE\_VALUE) {

if (DeviceIoControl(currentHandle,

SMART\_GET\_VERSION,

NULL, 0, &VP,

sizeof (VP), &br, NULL)) { // Get rid of flash drives //

SendMessage(handleComboBox, CB\_ADDSTRING, 0, (LPARAM) itoa(i, buffer, 10));

resultValue = TRUE;

}

}

CloseHandle(currentHandle);

} else {

MessageBox(hwnd, "This version Windows not supported!",

"Error", MB\_OK | MB\_ICONEXCLAMATION);

break; /\* unknown or non support version \*/

}

}

SendMessage(handleComboBox, CB\_SETCURSEL, 0, 1); // Installation 1-st value

return resultValue;

}

/\*

\* @brief Getting selected in the ComboBox disc

\* @return Device number in the system

\*/

int get\_selected\_disc\_number(HWND hWnd) {

char stringVal[10];

HWND handleComboBox = GetDlgItem(hWnd, IDC\_COMBO1);

GetDlgItemText(hWnd,IDC\_COMBO1,stringVal,9);

return atoi(stringVal);

}

/\*

\* @brief Getting string, pointing to the HDD device

\* @return "\\\\.\\PhysicalDrive" + disk number on system

\*/

char\* getNormaliseDrive(int nDrive) {

const char \*sBegin = "\\\\.\\PhysicalDrive";

char \*sEnd, \*sFull, buffer[3];

sEnd = itoa(nDrive, buffer, 10);

sFull = malloc(strlen(sBegin) + strlen(sEnd) + 1);

strcpy(sFull, sBegin);

strcat(sFull, sEnd);

return sFull;

}

void swap\_byte(char \*dat, int length) {

int i;

char tmp;

for (i = 0; i < length; i += 2) {

tmp = dat[i];

dat[i] = dat[i + 1];

dat[i + 1] = tmp;

}

}

//---------------------------------------------------------------------------

// ------------------------------- main.h -----------------------------------

#ifndef \_\_MAIN\_H\_\_

#define \_\_MAIN\_H\_\_

/\* Dialog box for macros \*/

#define IDC\_STATIC -1

#define SMART\_SIZE (512 - 1)

#define NUM\_ATTRIBUTE\_STRUCTS 30

#define SMART\_GET\_VERSION 0x00074080

#define SMART\_SEND\_DRIVE\_CMD 0x0007c084

#define SMART\_RECV\_DRIVE\_DATA 0x0007c088

#define IDE\_ATA\_ID 0xec

#define IDE\_ATAPI\_ID 0xa1

#define IDE\_EXEC\_SMART\_CMD 0xb0

#define SMART\_CYL\_LOW 0x4f

#define SMART\_CYL\_HI 0xc2

#define SMART\_READ\_ATTRIBUTE\_VALUES 0xd0

#define SMART\_READ\_ATTRIBUTE\_THRESHOLDS 0xd1

typedef struct \_GETVERSIONINPARAMS {

UCHAR bVersion;

UCHAR bRevision;

UCHAR bReserved;

UCHAR bIDEDeviceMap;

ULONG fCapabilities;

ULONG dwReserved[4];

} GETVERSIONINPARAMS, \*LPGETVERSIONINPARAMS;

typedef struct \_IDEREGS {

UCHAR bFeaturesReg;

UCHAR bSectorCountReg;

UCHAR bSectorNumberReg;

UCHAR bCylLowReg;

UCHAR bCylHighReg;

UCHAR bDriveHeadReg;

UCHAR bCommandReg;

UCHAR bReserved;

} IDEREGS, \*LPIDERE;

typedef struct \_DRIVERSTATUS {

UCHAR bDriverError;

UCHAR bIDEError;

UCHAR bReserved[2];

ULONG dwReserved[2];

} DRIVERSTATUS, \*LPDRIVERSTATUS;

typedef struct \_SENDCMDINPARAMS {

ULONG cBufferSize;

IDEREGS irDriveRegs;

UCHAR bDriveNumber;

UCHAR bReserved[3];

ULONG dwReserved[4];

UCHAR bBuffer[1];

} SENDCMDINPARAMS, \*LPSENDCMDINPARAMS;

typedef struct \_SENDCMDOUTPARAMS {

ULONG cBufferSize;

DRIVERSTATUS DriverStatus;

UCHAR bBuffer[1];

} SENDCMDOUTPARAMS, \*LPSENDCMDOUTPARAMS;

typedef struct \_IDENTSECDATA {

USHORT wGenConfig;

USHORT wNumCyls;

USHORT wReserved;

USHORT wNumHeads;

USHORT wBytesPerTrack;

USHORT wBytesPerSector;

USHORT wSectorsPerTrack;

USHORT wVendorUnique[3];

CHAR sSerialNumber[20];

USHORT wBufferType;

USHORT wBufferSize;

USHORT wECCSize;

CHAR sFirmwareRev[8];

CHAR sModelNumber[40];

BYTE bReserved[162];

} IDENTSECDATA, \*PIDENTSECDATA;

typedef struct \_DRVATTIBUTE {

BYTE bAttrID;

WORD wStatFlag;

BYTE bAttrVal;

BYTE bWorstVal;

WORD bRawVal[3];

BYTE bReserve;

} DRVATTRIBUTE, \*LPDRVATTRIBUTE;

const char \*aboutInfo = "Practical application to course project on the topic:"

"\n\"Monitoring the SMART status of the hard disk registers\"."

"\n\nAttention: program don't support reading"

"\ninformation from disks connected via SCSI interface,"

"\nas well as RAID arrays."

"\n\nPerformed: student group POIT-41;"

"\nBaranov V.";

const char \*textAttrib[] = {

"<unknown attribute>",

"Raw Read Error Rate",

"Throughput Performance",

"Spin-Up Time",

"Start/Stop Count",

"Reallocated Sectors Count",

"Read Channel Margin",

"Seek Error Rate",

"Seek Time Performance",

"Power-On Hours (POH)",

"Spin-Up Retry Count",

"Recalibration Retries",

"Device Power Cycle Count",

"Soft Read Error Rate",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"End-to-End error",

"<unknown attribute>",

"<unknown attribute>",

"Reported UNC Errors",

"Command Timeout",

"<unknown attribute>",

"Airflow Temperature (WDC)",

"G-sense error rate",

"Power-off retract count",

"Load/Unload Cycle",

"HDA temperature",

"Hardware ECC Recovered",

"Reallocation Event Count",

"Current Pending Sector Count",

"Uncorrectable Sector Count",

"UltraDMA CRC Error Count",

"Write / Multi-Zone Error Rate",

"Soft read error rate",

"Data Address Mark errors",

"Run out cancel",

"Soft ECC correction",

"Thermal asperity rate (TAR)",

"Flying height",

"Spin high current",

"Spin buzz",

"Offline seek performance",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"Disk Shift",

"G-Sense Error Rate",

"Loaded Hours",

"Load/Unload Retry Count",

"Load Friction",

"Load Cycle Count",

"Load 'In'-time",

"Torque Amplification Count",

"Power-Off Retract Cycle",

"<unknown attribute>",

"GMR Head Amplitude",

"Temperature",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"Head flying hours",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"<unknown attribute>",

"Read error retry rate"

};

#endif // \_\_MAIN\_H\_\_

//---------------------------------------------------------------------------

// ------------------------------- main.rc ----------------------------------

#include "resource.h"

#define APSTUDIO\_READONLY\_SYMBOLS

#include "afxres.h"

#undef APSTUDIO\_READONLY\_SYMBOLS

#if !defined(AFX\_RESOURCE\_DLL) || defined(AFX\_TARG\_RUS)

#ifdef \_WIN32

LANGUAGE LANG\_RUSSIAN, SUBLANG\_DEFAULT

#pragma code\_page( [ DEFAULT | CodePageNum ] )

#endif

IDR\_MAINFRAME ICON "AppIcon.ico"

#ifdef APSTUDIO\_INVOKED

1 TEXTINCLUDE DISCARDABLE

BEGIN

"resource.h\0"

END

2 TEXTINCLUDE DISCARDABLE

BEGIN

"#include ""afxres.h""\r\n"

"\0"

END

3 TEXTINCLUDE DISCARDABLE

BEGIN

"\r\n"

"\0"

END

#endif

MAIN DIALOG DISCARDABLE 0, 0, 390, 248

STYLE WS\_OVERLAPPED | WS\_CAPTION | WS\_SYSMENU

CAPTION "HDD S.M.A.R.T. Monitor"

CLASS "MAIN"

FONT 9, "FixedSys"

BEGIN

GROUPBOX "",IDC\_STATIC,4,213,381,30

COMBOBOX IDC\_COMBO1,40,224,80,60, WS\_VSCROLL | CBS\_DROPDOWNLIST |

CBS\_HASSTRINGS

DEFPUSHBUTTON "Refresh",IDOK,135,223,77,14

PUSHBUTTON "About",IDABOUT,217,223,77,14

PUSHBUTTON "Exit",IDCANCEL,300,223,77,14

CTEXT "Disk: ",IDC\_STATIC,10,224,25,12,SS\_CENTERIMAGE

CTEXT "",IDC\_STATIC1,4,2,381,14,SS\_CENTERIMAGE | WS\_BORDER

LISTBOX IDC\_LIST1,130,38,255,175,NOT LBS\_NOTIFY |

LBS\_NOINTEGRALHEIGHT | LBS\_DISABLENOSCROLL | LBS\_NOSEL |

WS\_VSCROLL | WS\_TABSTOP

GROUPBOX "Get VERSION",IDC\_STATIC,4,22,122,79

LTEXT "Version",IDC\_STATIC,10,35,58,8,SS\_CENTERIMAGE

LTEXT "Revision",IDC\_STATIC,10,48,58,8,SS\_CENTERIMAGE

LTEXT "Capabilities",IDC\_STATIC,10,61,58,8,SS\_CENTERIMAGE

LTEXT "DeviceMap",IDC\_STATIC,10,74,58,8,SS\_CENTERIMAGE

LTEXT "BytesReturned",IDC\_STATIC,10,87,58,8,SS\_CENTERIMAGE

CTEXT "",IDC\_STATIC2,70,35,51,8,SS\_CENTERIMAGE | WS\_BORDER

CTEXT "",IDC\_STATIC3,70,48,51,8,SS\_CENTERIMAGE | WS\_BORDER

CTEXT "",IDC\_STATIC4,70,61,51,8,SS\_CENTERIMAGE | WS\_BORDER

CTEXT "",IDC\_STATIC5,70,74,51,8,SS\_CENTERIMAGE | WS\_BORDER

CTEXT "",IDC\_STATIC6,70,87,51,8,SS\_CENTERIMAGE | WS\_BORDER

GROUPBOX "IDENTIFY DEVICE DATA",IDC\_STATIC,4,109,122,104

LTEXT "Mode number:",IDC\_STATIC,10,123,110,8,SS\_CENTERIMAGE

LTEXT "Firmware revision:",IDC\_STATIC,10,153,110,8,

SS\_CENTERIMAGE

LTEXT "Serial number:",IDC\_STATIC,10,184,110,8,SS\_CENTERIMAGE

LTEXT "",IDC\_STATIC7,10,135,110,10,SS\_CENTERIMAGE | WS\_BORDER

LTEXT "",IDC\_STATIC8,10,165,110,10,SS\_CENTERIMAGE | WS\_BORDER

LTEXT "",IDC\_STATIC9,10,196,110,10,SS\_CENTERIMAGE | WS\_BORDER

CTEXT "id",IDC\_STATIC,130,26,14,10,SS\_CENTERIMAGE | WS\_BORDER

CTEXT "raw values",IDC\_STATIC,149,26,42,10,SS\_CENTERIMAGE |

WS\_BORDER

CTEXT "current - worst",IDC\_STATIC,196,26,63,10,SS\_CENTERIMAGE |

WS\_BORDER

CTEXT "attribute name",IDC\_STATIC,264,26,121,10,SS\_CENTERIMAGE |

WS\_BORDER

END

#ifdef APSTUDIO\_INVOKED

GUIDELINES DESIGNINFO DISCARDABLE

BEGIN

"MAIN", DIALOG

BEGIN

RIGHTMARGIN, 380

BOTTOMMARGIN, 270

END

END

#endif

#endif

#ifndef APSTUDIO\_INVOKED

#endif

//---------------------------------------------------------------------------

// ------------------------------- resource.h -------------------------------

#define IDC\_LIST1 1000

#define IDC\_STATIC1 1001

#define IDC\_STATIC2 1002

#define IDC\_STATIC3 1003

#define IDC\_STATIC4 1004

#define IDC\_STATIC5 1005

#define IDC\_STATIC6 1006

#define IDC\_STATIC7 1007

#define IDC\_STATIC8 1008

#define IDC\_STATIC9 1009

#define IDC\_COMBO1 1010

#define IDABOUT 1011

#define IDR\_MAINFRAME 1012

#ifdef APSTUDIO\_INVOKED

#ifndef APSTUDIO\_READONLY\_SYMBOLS

#define \_APS\_NEXT\_RESOURCE\_VALUE 102

#define \_APS\_NEXT\_COMMAND\_VALUE 40001

#define \_APS\_NEXT\_CONTROL\_VALUE 1010

#define \_APS\_NEXT\_SYMED\_VALUE 101

#endif

#endif