**Федеральное государственное автономное образовательное учреждение**

**Высшего профессионального образования**

Санкт-Петербургский политехнический университет

Институт информационных технологий и управления

Кафедра компьютерных систем и программных технологий

**Отчёт по лабораторной работе №10**

«Утилита отображения аппаратного состава ПК (Windows)»

Работу выполнил студент гр. № 53501/3 Цыганов А.А.

Работу принял преподаватель \_\_\_\_\_\_\_\_\_\_\_\_\_\_ Душутина Е.В.

Санкт-Петербург

2016

# Цель работы:

Цель работы – изучение возможностей Windows API на примере создания утилиты.

**Постановка задачи:** Необходимо разработать утилиту, которая показывала анализ аппаратного состава компьютера (периферийные устройства, драйвера, порты). Приложение должно состоять из оконного интерфейса. Список периферийных устройств будет выводится на компонент listview. Динамическая библиотека для вызова Win API функций.

# Ход работы:

Для работы была использована среда разработки Visual Studio 2015 Enterprise в ОС Windows 10 Pro.

# Способы получения аппаратного состава компьютера средствами Windows

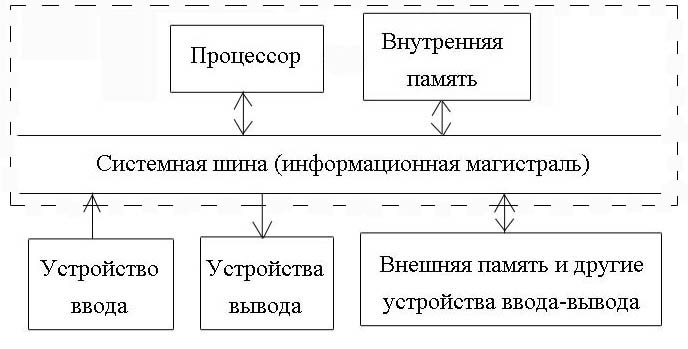
Аппаратное обеспечение компьютера — это все электронные и механические устройства компьютера. Периферийное устройство — аппаратура, которая позволяет вводить информацию в компьютер или выводить её из него. Периферийные устройства являются не обязательными для работы системы и могут быть отключены от компьютера. Однако большинство компьютеров используются вместе с теми или иными периферийными устройствами. Общая схема аппаратного обеспечения компьютера:



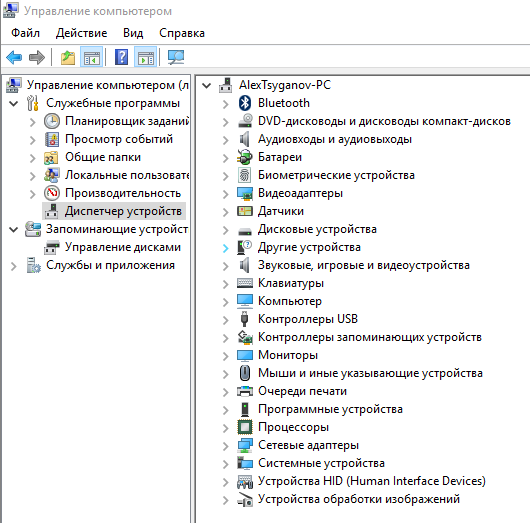
Структура аппаратного обеспечения персонального компьютера:

* системная плата (материнская плата) - на ней размещены:
  + процессор (центральный процессор) и система охлаждения,
  + внутренняя память,
  + системная шина,
  + слоты.
* платы периферии (могут быть встроены в системную плату) - на них размещены контроллеры устройств ввода-вывода и разъемы:
  + контроллеры дисководов,
  + видеокарта (видеоконтроллер, графическая плата),
  + звуковая карта (звуковой контроллер, звуковая плата),
  + сетевая карта (сетевой адаптер, сетевая плата),
  + контроллеры других устройств,
* устройства ввода и вывода:
  + дисководы:
    - дисковод на жёстком диске,
    - привод гибких дисков (дисковод для дискет),
    - привод оптических дисков (привод CD-ROM, CD-RW, DVD-ROM, DVD-RW),
    - стример
  + Клавиатура
  + Мышь, трекбол или тачпад
  + Джойстик
  + Сканер
  + Микрофон
* блок питания

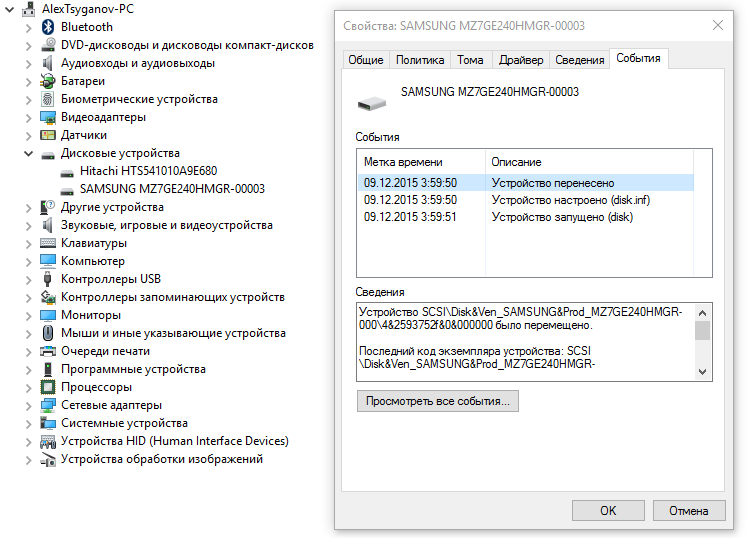
Общая схема компьютера показана ниже:



В Windows для получения аппаратного состава компьютера можно использовать диспетчер устройств. Ниже приведен список периферийных устройств компьютера, полученный в диспетчере задач:



Периферийные устройства разделены на соответствующие группы. Так же можно получить подробную информацию о каждом устройстве. Ниже приведен пример, сведения (драйвер, порт, тип устройства, изготовитель и т.п.) о дисковом накопителе Samsung:



# Разработка утилиты

Для того, чтобы получить аппаратный состав компьютера в своей программе можно воспользоваться соответствующими функциями Win API:

**SetupDiGetClassDevs**( GUID   \*ClassGuid, // Default NULL

PCTSTR Enumerator, // Default NULL

HWND   hwndParent, // Default NULL

DWORD  Flags) // Default DIGCF\_ALLCLASSES | DIGCF\_PROFILE

– функция возвращает указатель DeviceInfoSet на набор периферийных устройств локального компьютера.

## Параметры функции (описание из официальной документации MSDNA)

ClassGuid [входной параметр, обязательный]

Указатель на GUID для класса устройств или интерфейса устройства. Этот указатель является необязательным и может быть NULL.

Enumerator [входной параметр, обязательный]

Указатель на строку PCTSTR:

* Идентификатор (id) подключи и Plug and Play (PnP). Этот идентификатор может быть либо глобально уникальный идентификатор-значения (GUID) или символическое имя. Например, "PCI" может использоваться для указания значения PCI PnP. Другие примеры символьных имен для значений PnP: "USB", "PCMCIA" и "SCSI" и другие.
* PnP устройства экземпляр ID. При указании PnP экземпляра устройства, DIGCF\_DEVICEINTERFACE интерфейсное устройство должно быть установлено в параметре Flags. Этот указатель является необязательным и может быть NULL. Если значение перечисления не используется для выбора устройств, установите Enumerator в NULL.

hwndParent [входной параметр, обязательный]

Дескриптор окна верхнего уровня, которые будут использоваться для пользовательского интерфейса, который связан с установкой экземпляра устройства для получения информации об наборе устройств. Это указатель обязательным и может быть NULL.

Flags [входной параметр]

Переменная типа DWORD, который определяет параметры управления, фильтрации информации об устройствах. Этот параметр может быть задан побитовое или null или из нескольких флагов:

DIGCF\_ALLCLASSES

Возвращает список всех установленных классов или интерфейсов устройств.

DIGCF\_DEVICEINTERFACE

Возвращает список устройств, которые поддерживают интерфейсы для указанных классов устройств. Этот флаг должен быть установлен в параметре Flags, если Enumerator параметр указывает на конкретный экземпляр устройства.

DIGCF\_DEFAULT

Возвращает устройства, которые относятся к основным интерфейсным устройствам системы, если он установлен, для указанного класса интерфейсов устройства.

DIGCF\_PRESENT

Возвращает устройства, которые в настоящее время присутствуют в системе.

DIGCF\_PROFILE

Возвращает устройства, которые являются частью текущего профиля аппаратных средств.

## Функция возвращает:

Если операция прошла успешно, SetupDiGetClassDevs возвращает дескриптор информационного набора устройств, который содержит все установленные устройства, по заданным параметрам. Если операция не выполняется, то функция возвращает INVALID\_HANDLE\_VALUE. Чтобы получить дополнительную информацию об ошибке, можно воспользоваться функцией GetLastError.

Алгоритм получения периферийных устройств на основе приведенных функций Win API:

1. Получение структуры HDEVINFO m\_hDevInfo = SetupDiGetClassDevs(NULL, NULL, NULL, DIGCF\_ALLCLASSES | DIGCF\_PROFILE); Список аппаратурного состава всех периферийных устройств.
2. Цикл перебора для выборки каждого устройства:

while (SetupDiEnumDeviceInfo(m\_hDevInfo, m\_MemberIndex, &m\_spDevInfoData))

1. Получение подробной информации о выборанном устройстве: название, описание, производить, драйвер, расположение:

* GetDeviceRegistryProperty(SPDRP\_CLASS, (PBYTE)szBuf);
* GetDeviceRegistryProperty(SPDRP\_FRIENDLYNAME, (PBYTE)szFriendlyName);
* GetDeviceRegistryProperty(SPDRP\_DRIVER, (PBYTE)szDriver);
* GetDeviceRegistryProperty(SPDRP\_MFG, (PBYTE)szMfg);
* GetDeviceRegistryProperty(SPDRP\_PHYSICAL\_DEVICE\_OBJECT\_NAME, (PBYTE)szPhysical);
* SetupDiGetClassDescription(&m\_spDevInfoData.ClassGuid, szDesc, MAX\_PATH, NULL);

1. Данные полученные из каждого устройства формируют структуру:

typedef struct DeviceData

{

int id;

wchar\_t \*name, \*descr, \*friendly\_Name, \*driver, \*mfg, \*physical\_Device;

} DeviceData;

Спецификация:

|  |  |  |
| --- | --- | --- |
| Функция | Место в коде | Описание |
| HDEVINFO SetupDiGetClassDevs( NULL, NULL, NULL, DIGCF\_ALLCLASSES | DIGCF\_PROFILE) | AlexWinDevicesLib.cpp:27 | Получение списка всех периферийных устройств. |
| SetupDiEnumDeviceInfo (m\_hDevInfo, m\_MemberIndex, &m\_spDevInfoData) | AlexWinDevicesLib.cpp:53 | Метод для выборки данных устройства в spDevInfoData по индексу m\_MemberIndex из списка m\_hDevInfo. |
| GetDeviceRegistryProperty(SPDRP\_CLASS, (PBYTE)szBuf); | AlexWinDevicesLib.cpp:58 | Название устройства в переменную szBuf |
| SetupDiGetDeviceRegistryProperty( m\_hDevInfo, &m\_spDevInfoData, Property, 0L, PropertyBuffer, 2048, 0); | AlexWinDevicesLib.cpp:38 | Получение подробной информации устройства (m\_spDevInfoData) выбранного параметра (Property) в переменную PropertyBuffer. |
| void AlexWinDevices::getAllDevices( cb\_enumWinDevices cb) | AlexWinDevicesLib.cpp:47 | Метод класса AlexWinDevices для выборки периферийных устройств. Параметр cb\_enumWinDevices функция callback |

Получение списка периферийных реализовано в динамической библиотеке, которая состоит из 2х файлов AlexWinDevixesLib.h AlexWinDevixesLib.m, внутри которых реализован класс AlexWinDevices. Внутри класса реализован алгоритм с вызовами Win API функций для получения аппаратного состава компьютера.

В клиентском коде необходимо подключить библиотеку и создать экземпляр класса и вызвать метод getAllDevices(), который через функцию callback возвращает структуру DeviceData c данными о каждом устройстве.

Ниже приведен фрагмент кода реализации класса AlexWinDevices:

AlexWinDevicesLib.h

typedef struct DeviceData // структура с информацией (название, описание, производить, драйвер и т.п.) об устройстве

{

int id;

wchar\_t \*name, \*descr, \*friendly\_Name, \*driver, \*mfg, \*physical\_Device;

}DeviceData;

typedef void(\*cb\_enumWinDevices) (DeviceData data); // функция callback

class DLL\_API AlexWinDevices

{

private:

HDEVINFO m\_hDevInfo;

SP\_DEVINFO\_DATA m\_spDevInfoData;

int m\_MemberIndex;

BOOL GetDeviceRegistryProperty(DWORD Property, PBYTE PropertyBuffer);

std::vector<DeviceData> devices;

public:

AlexWinDevices();

void getAllDevices(cb\_enumWinDevices cb); // метод для получения списка аппаратного состава компьютера

};

AlexWinDevicesLib.m

AlexWinDevices::AlexWinDevices()

{

m\_hDevInfo = SetupDiGetClassDevs(NULL, NULL, NULL, DIGCF\_ALLCLASSES | DIGCF\_PROFILE);

}

/\*\*

Retrieve a specified Plug and Play device property.

@param Property A value indicating the property to be retrieved such as SPDRP\_CLASSGUID or SPDRP\_FRIENDLYNAME. See

http://msdn.microsoft.com/en-us/library/ms792967.aspx for full details.

@param PropertyBuffer A pointer to a buffer that receives the property that is being retrieved.

\*/

BOOL AlexWinDevices::GetDeviceRegistryProperty(DWORD Property, PBYTE PropertyBuffer)

{

BOOL bGotRegProp = SetupDiGetDeviceRegistryProperty(m\_hDevInfo, &m\_spDevInfoData,

Property,

0L,

PropertyBuffer,

2048,

0);

return bGotRegProp;

}

void AlexWinDevices::getAllDevices(cb\_enumWinDevices cb)

{

devices.clear(); // отчистить ранее полученные устройства

m\_MemberIndex = 0; // обнулить порядковый индекс устройства

m\_spDevInfoData.cbSize = sizeof(SP\_DEVINFO\_DATA);

while (SetupDiEnumDeviceInfo(m\_hDevInfo, m\_MemberIndex, &m\_spDevInfoData)) // получения указателя на найденное устройство

{

m\_MemberIndex++;

wchar\_t szBuf[MAX\_PATH] = { 0 };

if (GetDeviceRegistryProperty(SPDRP\_CLASS, (PBYTE)szBuf)) // функция возвращает название устройства, если нет название, то переход к следующему устройству

{

DeviceData data; // объявление структуры данных об устройстве для передачи в клиентский код

data.id = m\_MemberIndex;

data.name = szBuf;

wchar\_t szFriendlyName[MAX\_PATH] = { 0 };

AlexWinDevices::GetDeviceRegistryProperty(SPDRP\_FRIENDLYNAME, (PBYTE)szFriendlyName);

data.friendly\_Name = szFriendlyName;

wchar\_t szDriver[MAX\_PATH] = { 0 };

AlexWinDevices::GetDeviceRegistryProperty(SPDRP\_DRIVER, (PBYTE)szDriver);

data.driver = szDriver;

wchar\_t szMfg[MAX\_PATH] = { 0 };

GetDeviceRegistryProperty(SPDRP\_MFG, (PBYTE)szMfg);

data.mfg = szMfg;

wchar\_t szPhysical[MAX\_PATH] = { 0 };

GetDeviceRegistryProperty(SPDRP\_PHYSICAL\_DEVICE\_OBJECT\_NAME, (PBYTE)szPhysical);

data.physical\_Device = szPhysical;

wchar\_t szDesc[MAX\_PATH] = { 0 };

SetupDiGetClassDescription(&m\_spDevInfoData.ClassGuid, szDesc, MAX\_PATH, NULL); // функция возвращает локализированное описание устройства

data.descr = szDesc;

int ImageIndex = 0;

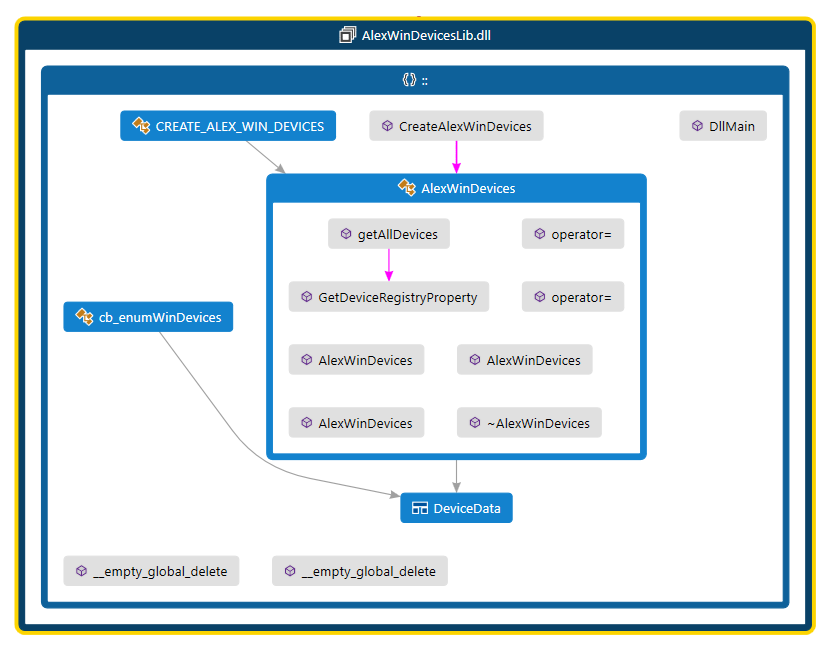
if (cb)

cb(data); // вызов функции callback и передача данных об устройстве

}

}

Архитектура динамической библиотеки AlexWinDevicesLib.dll, построенная в Visual Studio 2015. На которой показан зависимость вызываемых функций и связи между ними.



Для представления полученных данных об аппаратном составе компьютера, реализован оконный windows интерфейс. Для создания диалогового окна используются функции Win API. При создании проекта, Visual Studio формирует необходимые конфигурационные файлы и создает шаблон диалогового окна.

Ниже приведен фрагмент кода, в котором создается главное диалоговое окно приложения и функции обработки событий, происходящих над диалоговым окном:

HINSTANCE hInst;

HINSTANCE mMAINDIALOG;

HWND listView;

LVCOLUMN LvCol; // Make Coluom struct for ListView

LVITEM LvItem; // ListView Item struct

BOOL CALLBACK cb\_mainDialog(HWND, UINT, WPARAM, LPARAM);

AlexWinDevices getAlexWinDevicesFromDLL();

void cb\_enumWinDevices2(DeviceData data)

{

wchar\_t buf[256], buf2[256], buf3[256];

swprintf(buf2, sizeof(buf2), L" Driver:%ls; %ls", data.driver, data.physical\_Device);

SendMessage(listView, LB\_INSERTSTRING, 0, (LPARAM)buf2); // вывод строки на компонент listview

swprintf(buf, sizeof(buf), L"%ls -- %ls -- %ls", data.name, data.friendly\_Name, data.descr);

SendMessage(listView, LB\_INSERTSTRING, 0, (LPARAM)buf); // вывод строки на компонент listview

}

// функция входа в программу

INT APIENTRY WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance,

LPSTR lpCmdLine, int nCmdShow)

{

DialogBox(mMAINDIALOG, MAKEINTRESOURCE(IDD\_DIALOG1), 0, cb\_mainDialog); // создание диалогового окна на основе шаблона IDD\_DIALOG1

return 0;

}

//

// ФУНКЦИЯ: cb\_mainDialog(HWND, UINT, WPARAM, LPARAM) – обрабатывает события

//

// НАЗНАЧЕНИЕ: обрабатывает сообщения в главном окне.

//

// WM\_COMMAND — обработать меню приложения

// WM\_PAINT — отрисовать главное окно

// WM\_DESTROY — отправить сообщение о выходе и вернуться

//

//

BOOL CALLBACK cb\_mainDialog(HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam)

{

switch (message)

{

case WM\_COMMAND:

{

int wmId = LOWORD(wParam);

switch (wmId)

{

case IDH\_CANCEL:

case IDCLOSE:

DestroyWindow(hWnd); // закрытие диалогового окна

break;

case MB\_OK:

case IDC\_ABOUT:

break;

default:

return DefWindowProc(hWnd, message, wParam, lParam);

}

}

break;

case WM\_INITDIALOG:

{

listView = GetDlgItem(hWnd, IDC\_LIST1); // получение указателя на компонент listview

AlexWinDevices lib = getAlexWinDevicesFromDLL(); // создание экземпляра класса AlexWinDevices из динамической библиотеки.

lib.getAllDevices(cb\_enumWinDevices2); // вызов метода для получения периферийных устройств

}

break;

case WM\_DESTROY:

break;

default:

return DefWindowProc(hWnd, message, wParam, lParam);

}

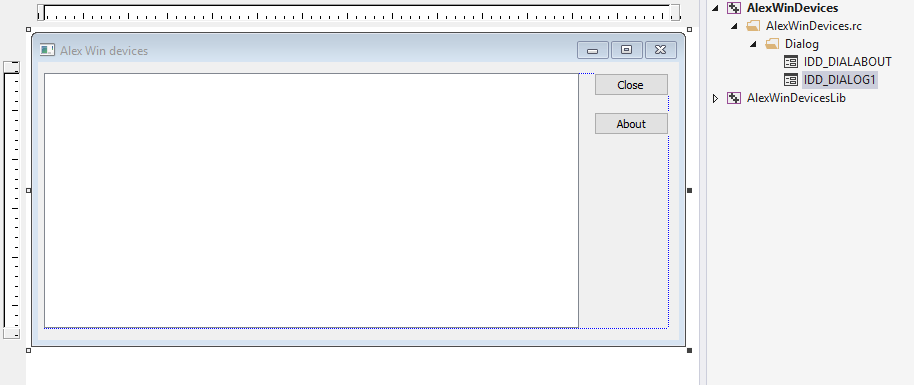
return 0;

}

Спецификация:

|  |  |  |
| --- | --- | --- |
| Функция | Место в коде | Описание |
| INT APIENTRY WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR lpCmdLine, int nCmdShow) | main.cpp:27 | Точка входа в приложение. |
| BOOL CALLBACK cb\_mainDialog(HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam) | main.cpp:44 | Перехват событий, происходящих на главном диалоговом окне. После создания окна, создается экземпляр класса AlexWinDevices из динамической библиотеки и выборка периферийных устройств. |
| void cb\_enumWinDevices2(DeviceData data) | main.cpp:16 | Callback функция из динамической библиотеки. Через переменную DeviceData получение данных о найденном устройстве. |
| SendMessage(listView, LB\_INSERTSTRING, 0, (LPARAM)buf2); | main.cpp:21 | Добавление строки buf2(с данными устройства) на компонент listview. |
| HWND GetDlgItem(hWnd, IDC\_LIST1); | main.cpp:69 | Возвращает идентификатор на компонент listveiw |

Для отображения списка устройстве используется компонент listview. Для вывода информации на этот компонент можно использовать Win API функцию SendMessage(), параметры которой указываются, идентификатор listview, LB\_INSERTSTRING (действие) , сообщение (набор символов). Ниже приведен интерфейс главного экрана в приложении, построенный в Visual Studio 2015:



Сборка проекта в Visual Studio 2015 происходит по этапам, компилируется динамическая библиотека и затем создается диалоговое окно:

1>------ Сборка начата: проект: AlexWinDevicesLib, Конфигурация: Debug Win32 -

1> stdafx.cpp

1> AlexWinDevicesLib.cpp

1> Создание кода...

1> Создается библиотека C:\Users\Alexandr\Documents\Visual Studio 2015\Projects\Системное программирование\AlexWinDevices\Debug\AlexWinDevicesLib.lib и объект C:\Users\Alexandr\Documents\Visual Studio 2015\Projects\Системное программирование\AlexWinDevices\Debug\AlexWinDevicesLib.exp

1> AlexWinDevicesLib.vcxproj -> C:\Users\Alexandr\Documents\Visual Studio 2015\Projects\Системное программирование\AlexWinDevices\Debug\AlexWinDevicesLib.dll

2>------ Сборка начата: проект: AlexWinDevices, Конфигурация: Debug Win32 ----

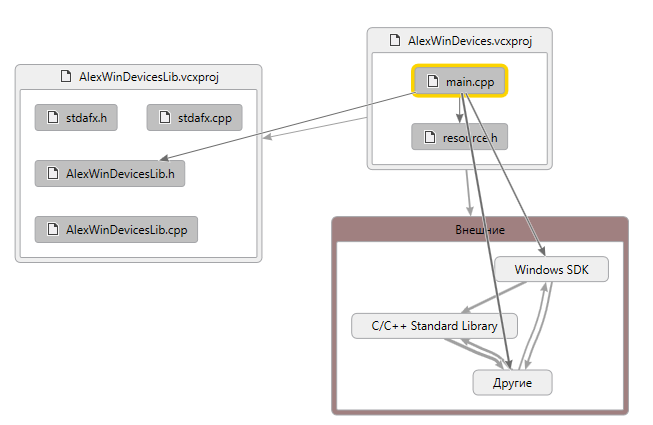
2> main.cpp

2> AlexWinDevices.vcxproj -> C:\Users\Alexandr\Documents\Visual Studio 2015\Projects\Системное программирование\AlexWinDevices\Debug\AlexWinDevices.exe

2> AlexWinDevices.vcxproj -> C:\Users\Alexandr\Documents\Visual Studio 2015\Projects\Системное программирование\AlexWinDevices\Debug\AlexWinDevices.pdb (Partial PDB)

========== Сборка: успешно: 2, с ошибками: 0, без изменений: 0, пропущено: 0

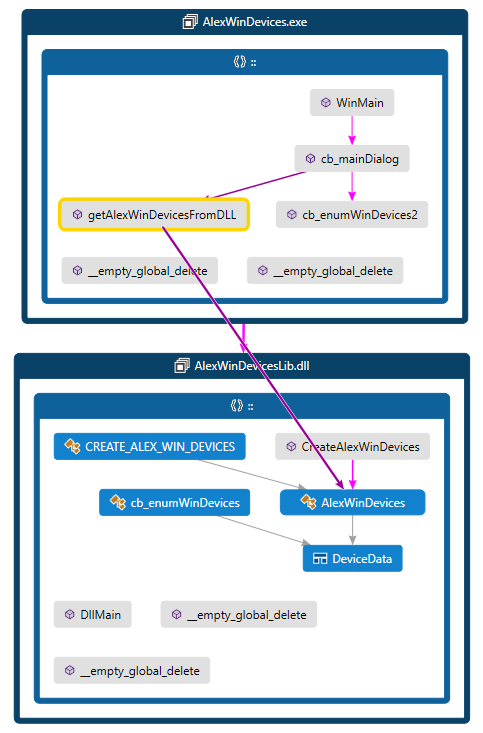
Ниже показана диаграмма включаемых файлов, при сборке проекта. Стрелками показана их зависимость.



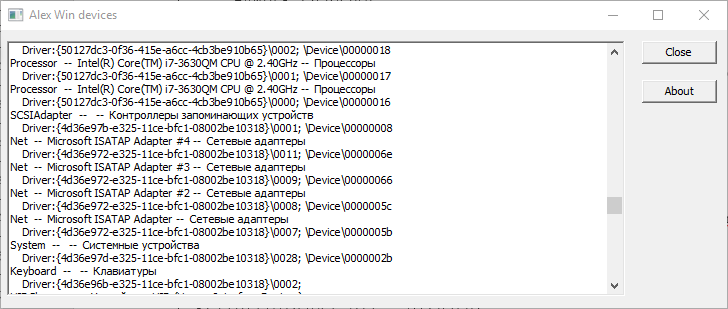
Стек вызовов выглядит следующим образом, в момент вызова функции getAllDevices()



Архитектура решения из динамической библиотеки и оконного приложения, построенная средствами Visual Studio 2015, выглядит следующим образом. На которой показаны связи между вызываемыми функциями.

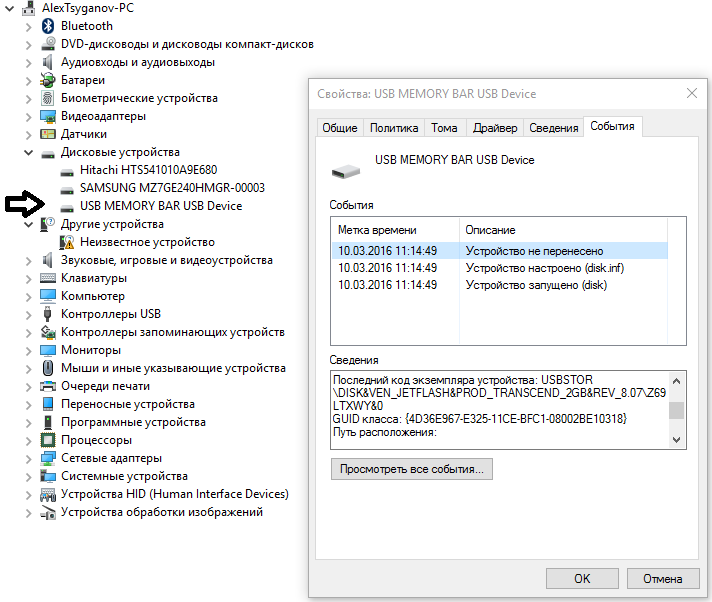


После успешной сборки, запускаем AlexWinDevices.exe, отображается главное окно приложение, в котором выведена информация периферийных устройств компьютера:

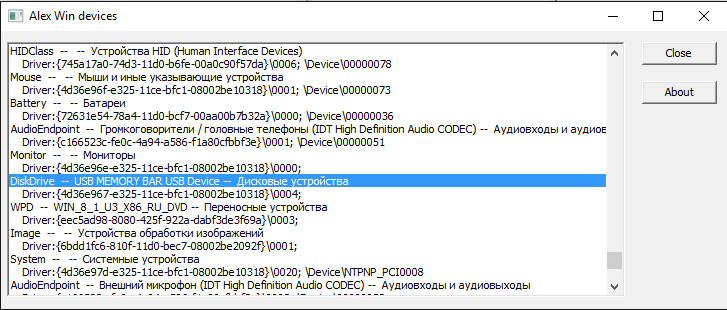


Проведем тест приложения. К компьютеру подключили usb устройство. Проверим какие данные покажет диспетчер устройств Windows:

* **Дисковое устройство: USB MEMORY BAR USB Device**



Теперь запустим приложение и найдем вставленное usb дисковое устройство.



После проведения несколько тестов, можно утверждать, что полученное приложение правильно отображает информацию о текущем аппаратном составе компьютера. Так же приложение совместимо с ранними версиями Windows. Отдельно работающая динамическая библиотека может использоваться и другими приложениями для получения периферийного состава компьютера.

Используемые источники

1. https://msdn.microsoft.com/ru-ru/library/windows/hardware/ff551069(v=vs.85).aspx
2. https://support.microsoft.com/ru-ru/kb/259695
3. https://ru.wikipedia.org/wiki/%D0%90%D0%BF%D0%BF%D0%B0%D1%80%D0%B0%D1%82%D0%BD%D0%BE%D0%B5\_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5
4. http://infoegehelp.ru/index.php?option=com\_content&view=article&id=430:apparatnoe-obespechenie&catid=50:ustr-comp&Itemid=100