СММ

Современная коммуникационная периферия.
USB, Ethernet

Коммуникационные задачи для различных приложений

- **Сбор данных:** Обмен сигналами управления и данными
  температура, давление, телеметрия, счетчики расхода и т.д.
- **Комплексное управление:** Системы с обратной связью и диагностикой
  поддержание температуры, системы индикации и т.д
- **Идентификация объектов:** Периодический или постоянный запрос
  системы доступа, отслеживание контейнеров, RFID и т.д.
- **Офисная и домашняя автоматизация:** доступ к различному типу контента
  сканнеры бар-кодов, считыватели карт, VoIP, UPS, маршрутизаторы данных, музыка/радио, игровые контроллеры

Требования, влияющие на выбор решения

- Возможность доступа после установки (upgrade)
  - Требуется ли Upgrade? Удаленный или локальный ?
- Требования к характеристикам
  - Быстродействие, скорость обмена, тип трафика (постоянный или «взрывной»)
- Механические и температурные требования
  - Температурный диапазон, ЭМС, шумы, вибрации
- Стоимость обслуживания
  - Соотношение характеристики/качество
  - Наличие стандартных кабелей
- Требования к ответной части
  - Существуют ли готовые решения ?
- Целостность данных
  - Безопасность, секретность, авто-повтор

Альтернативы решений: компьютер, модуль, микроконтроллер

- Компьютер
  - Высокая стоимость. Избыточность
  - Потенциальные отказы механических деталей (диски, вентиляторы)
- Готовые адаптеры и модули
  - Не требуется разработка ПО
  - Применимо при большом количестве портов
  - Удобно для запуска прототипов
  - Фиксированная функциональность. Нет возможности улучшения/расширения
  - Обычно ограничивается преобразователями последовательных протоколов
- Микроконтроллерное решение
  - Низкая стоимость при одном или небольшом количестве портов
  - Минимальная разработка ПО
  - Минимальные аппаратные затраты
  - Может быть добавлено к существующим изделиям
  - Применяемые стеки ПО масштабируются к любому семейству контроллеров
  - Могут быть совмещены несколько типов протоколов связи, и добавлены другие возможности (графика, логика,)

Альтернативы протоколов

TCP/IP через Ethernet

- Самая широко распространенная сеть
  - Наиболее известная
  - Присутствует везде: офис, дом, пром. предприятие
  - Огромная готовая инфраструктура
- Совместимость
  - Открытые стандарты
  - Стандартизованные протоколы
  - Стандартное готовое ПО
  - Браузеры, E-mail клиенты и т.д.
  - Большое сообщество поддержки
- Маленькие задержки
  - Близко к реальному времени доставки пакетов
- Масштабируемость и расширяемость
  - Недороое оборудование
  - Автоматическая конфигурация
- Бесшовное соединение с сетью Интернет
  - Единые стандарты, единые протоколы
  - Кросс- платформенность
- Легкая миграция к беспроводному WiFi
- Защита даннных через SSL

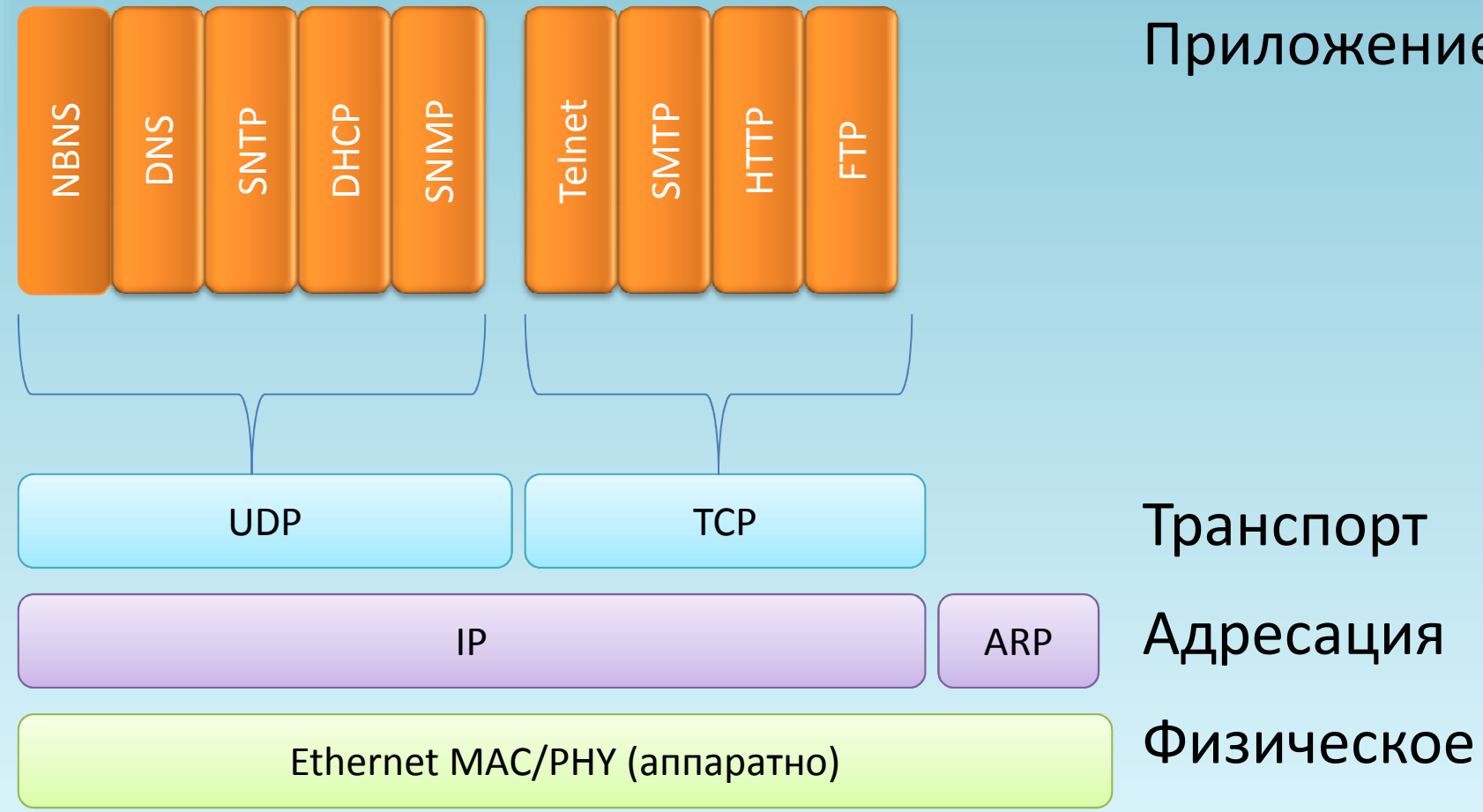
TCP/IP: Сервер или клиент ?

- Сервер
  - Слушает соединения от других устройств
  - Всегда включен. Всегда ждет
  - Имеет известный фиксированный адрес
- Клиент
  - Устанавливает соединение с другими устройствами
  - Активен по запросу.
  - Динамический адрес или местоположение
- В обоих случаях обмен данными в обе стороны !!!
  - Разница в том, кто устанавливает соединение

TCP/IP: Адресация

- MAC адрес
  - Связан с конкретным аппаратным экземпляром изделия
  - 6 байт. Например, 00:04:A3:00:12:34
    (Старшие 3 байта являются кодом поставщика оборудования)
  - Глобально уникален. Выделяется IEEE
  - Используется для адресации внутри локальной сети
- IP адрес
  - Назначается программно в сети. Вручную или выделяется DHCP сервером
  - 4 байта. Например, 192.168.1.100
  - Распределяется региональным администратором
    (Например, Интернет сервис –провайдером)
  - Некоторые адреса закреплены для частных сетей, отделенных маршрутизатором или Firewall-ом
    (192.168.*, 10.*, 169.254.*, 172.16.*)

TCP/IP: Стек протоколов



TCP/IP: TCP или UDP ?

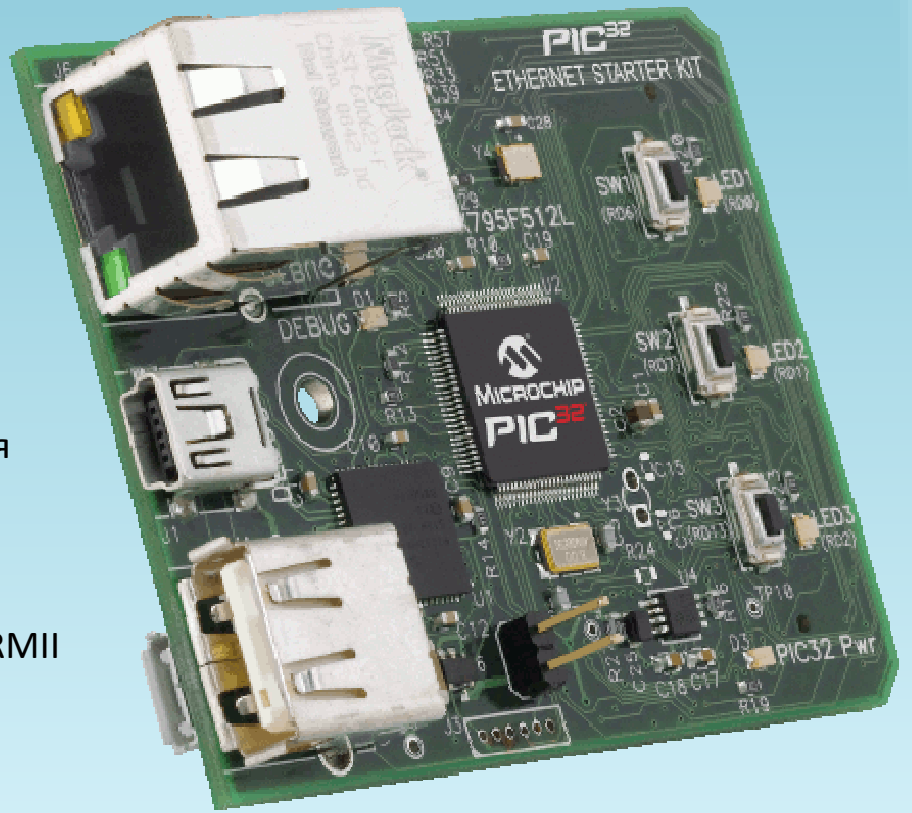
- UDP – передача датаграмм без установки соединения
  - Нет потерь на установку соединения
  - Доставка не гарантирована. Пакеты могут теряться или приходить разрушенными
  - Пакеты могут посылаться одному узлу или нескольким одновременно
  - Удобны для вспомогательных задач сети
- TCP – обмен данными в рамках установленного соединения
  - Установка соединения между двумя устройствами
  - Подтверждение правильного приема всех данных
  - Потерянные или разрушенные пакеты автоматически повторяются
  - Дублирующие пакеты игнорируются
  - Порядок получения пакетов гарантируется

Лабораторная работа: Шаг 1

- Создание TCP сервера
  - Аппаратная реализация: PIC32 Ethernet Starter Kit
  - Реализация TCP/IP стека: Microchip Application Library
  - Задача: Принимать входящее соединение, обрабатывать полученные в соединении данные, выдавать ответы.

Шаг 1: PIC32 Ethernet Starter Kit

- Микроконтроллер PIC32MX795 со встроенной поддержкой MAC уровня Ethernet
  - Полу- и полно- дуплексные операции
  - Конфигурация фильтрации пакетов
  - Конфигурация прерываний
  - Ручное и автоматическое управление потоком
  - Вычисление контрольной суммы
  - Каналы DMA для приема и передачи
  - RX фильтр
  - RMII интерфейс для физического уровня
- Контроллер физического уровня Ethernet
  - Контроллер DP83848
  - Интерфейс к микроконтроллеру через RMII
- USB
  - Возможность подключения в режимах HOST, DEVICE и OTG
- Встроенный отладчик/программатор
- Ввод/Вывод
  - 3 светодиода
  - 3 кнопки



Шаг 1: TCP/IP стек Microchip

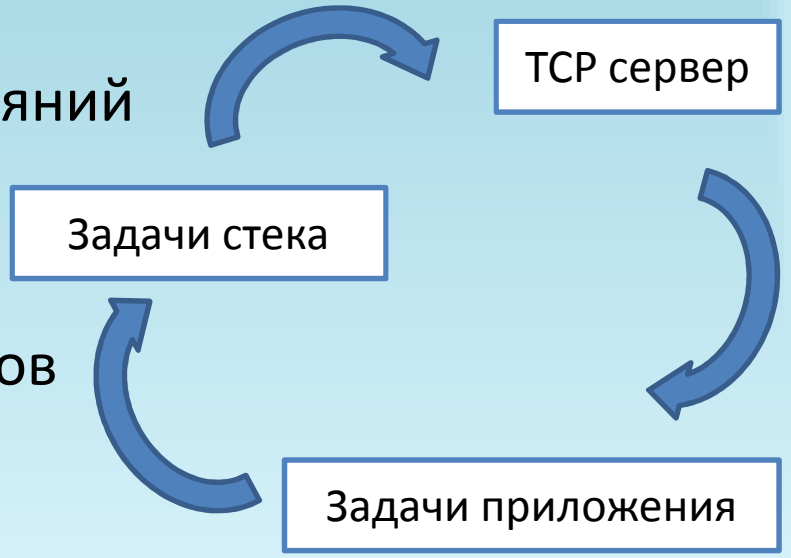
- Поставляется в исходных кодах на Си
- Бесплатное лицензионное соглашение
- Поддерживает семейства PIC18, PIC24, dsPIC DSC, PIC32
- Независим от используемой RTOS
- Модульный подход: подключаются только те модули, которые необходимы
- Поддержка одновременно нескольких соединений и транспортов
- Множество готовых примеров
- Утилита конфигурации

Шаг 1: Конфигурация стека

- Выбор протоколов
  - Необходимый набор протоколов стека для установки TCP соединения:
    - IP (Основной протокол передачи)
    - TCP (Для установки TCP соединения)
    - UDP (Для поддержки вспомогательных протоколов)
    - ARP (Для привязки MAC адреса к IP адресу)
    - DHCP сервер (Для выдачи IP адреса компьютеру)
    - ICMP сервер (Для ответа на PING)
- Назначение адресов
  - Назначение MAC адреса платы
  - Назначение IP адреса и маски подсети для создаваемого сервера
  - Назначение IP адреса шлюза
- Задание количества сокетов
  - Для работы сервера назначает только один TCP сокет
  - Максимальное количество UDP сокетов: 8 (по умолчанию)
- Использование утилиты конфигурации

Шаг1: Кооперативная многозадачность

- Последовательное переключение задач
  - Ни одна задача не должна блокировать процессор
  - Долгие задачи должны быть разбиты на части
  - Использование машины состояний
- Преимущества
  - Малая избыточность кода
  - Гибкое распределение ресурсов
  - Совместимость с RTOS



Шаг 1: Кооперативная многозадачность

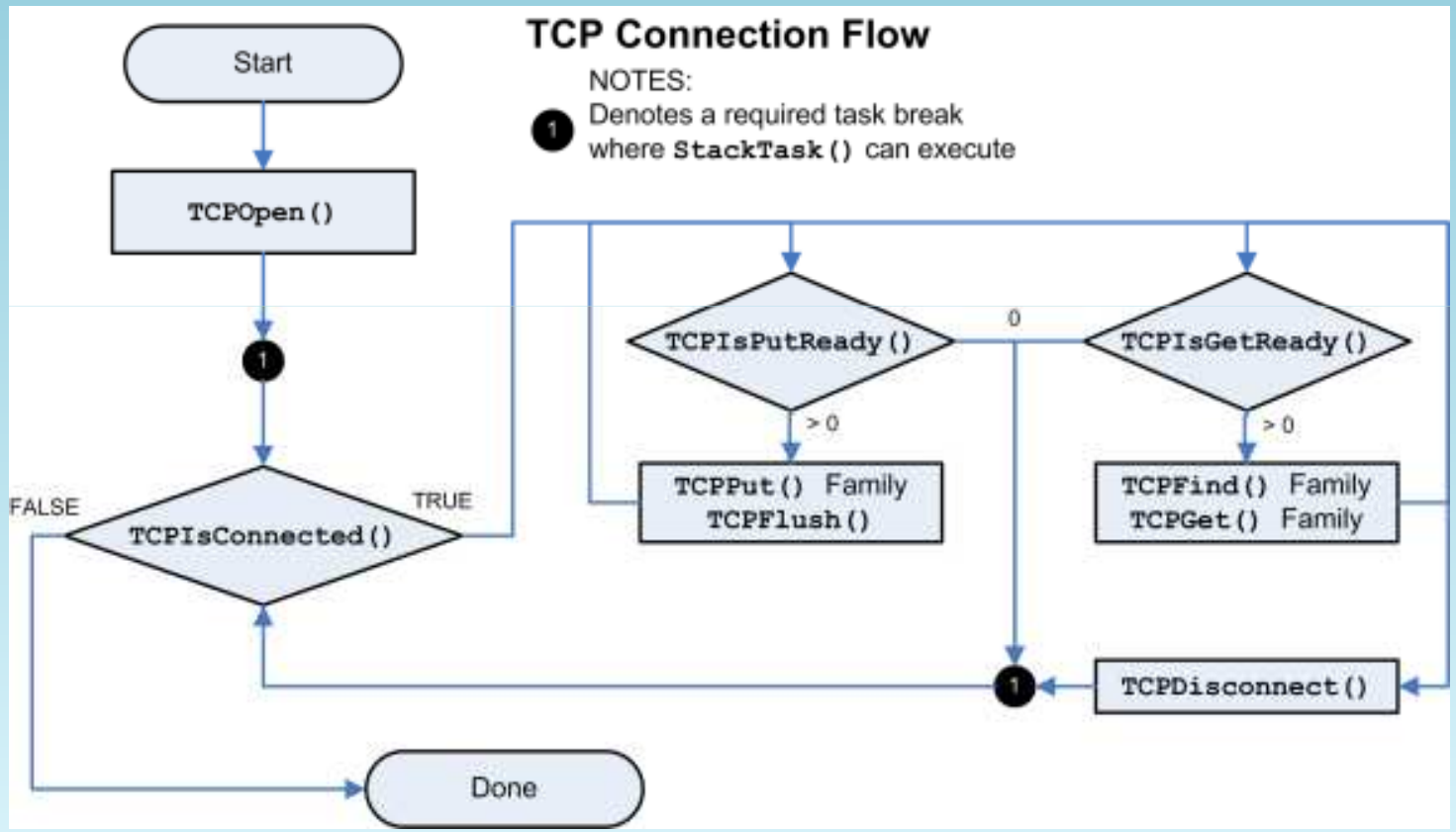
```
int main(void)
{
    InitializeBoard();  // Вся
    AdditionalInit();   // необходимая
    StackInit();        // инициализация

    while(1)            // бесконечный цикл
    {
        StackTask();          // задача стека
        StackApplications(); // приложения стека
        UserApp();            // приложения пользователя
     }
}
```

Шаг 1: Кооперативная многозадачность

- Потенциальные проблемы
  - Долгие задачи: Необходимо разбивать и вводить машину состояний
  - Программные задержки: Использовать модуль Tick из стека
  - UART и другой ввод-вывод: работать по прерываниям
- Как часто ?
  - Чем чаще – тем лучше
  - Общее время цикла:
    - Цель: не более 1-2 мс
    - Приемлемо: 10-20 мс
    - Допустимо в крайнем случае: 100 и более мс

Шаг 1: Управление соединением для TCP сервера



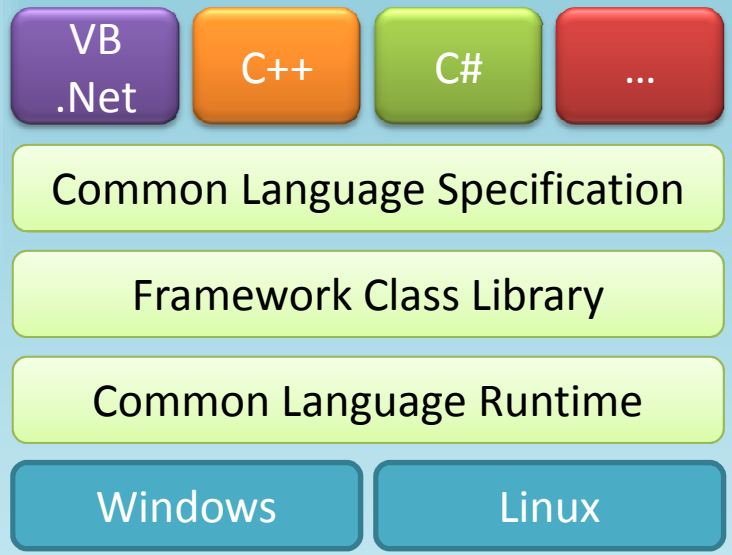
Лабораторная работа: Шаг 2

- Создание TCP клиента
  - Использование среды .Net
  - Использование Visual C++ Express IDE

Шаг 2: Современные требования к разработке программ для компьютера

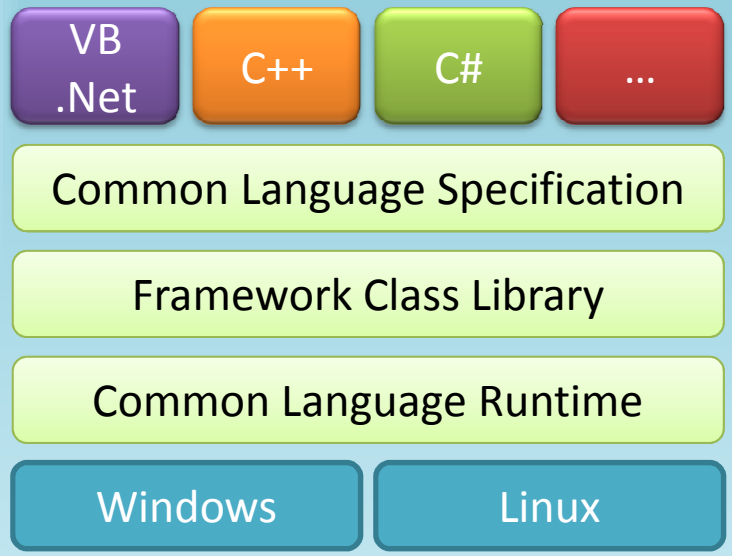
- Упрощенная разработка пользовательского графического интерфейса
- Независимость от платформы
- Возможность разработки на нескольких языках программирования
- Возможность создания как локальных, так и распределенных приложений
- Новые возможности по безопасности и надежности
  - Назначение прав доступа и выполнения различных действий для пользователей и групп

Шаг 2: Среда .Net



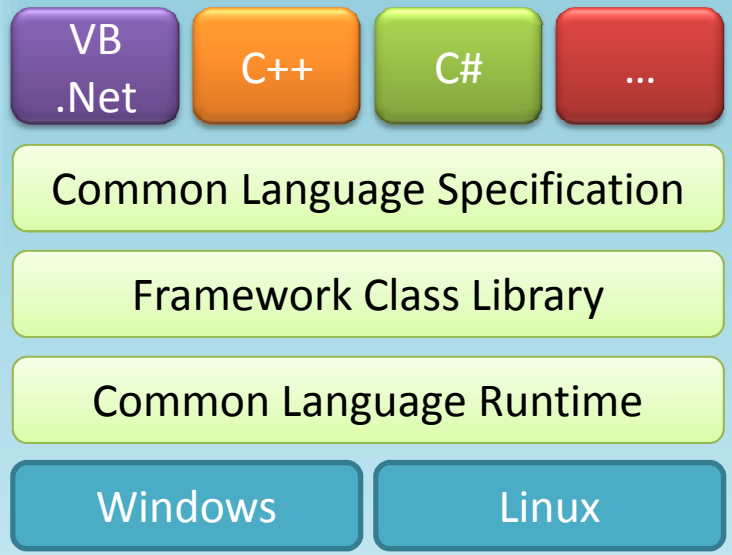
- Абстрактный уровень над имеющейся ОС
- Приложения компилируются в специальный промежуточный язык (IL) и выполняются в «управляемой» («managed»)среде
- Основано на базе стандарта общего языка (Common Language Infrastructure – CLI)
- 2 варианта реализации
  - Microsoft .Net
  - Nowell’s Mono Project

Шаг 2: Common Language Runtime (CLR)



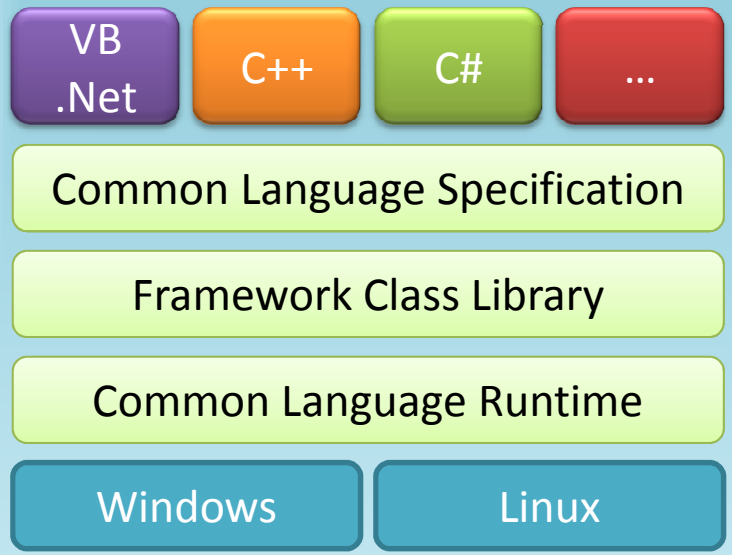
- Виртуальная машина
  - Загружает .Net сборки (assemblies)
  - Проверяет, что IL код безопасен для выполнения, включая проверку типов аргументов
  - Непосредственно перед выполнением компилирует IL код в машинный код
  - Собирает «мусор»

Шаг 2: Framework Class Library (FCL)



- Библиотека из тысяч готовых классов, выполняющих множество системых функций
- «Обертка» для «обычных» Win32 API вызовов
- Примеры классов
  - System::IO
    - Классы для работы с файлами и другими потоками ввода-вывода
  - System::Windows::Forms
    - Классы для создания элементов графического пользовательского интерфейса
  - System::Net::Sockets
    - Классы для работы с TCP/IP сокетами

Шаг 2: Common Language Specification (CLS)



- Спецификация CLS определяет основную функциональность которую должен обеспечивать любой .Net язык программирования для обеспечения совместимости
  - Класс, написанный на одном языке должен иметь возможность наследовать класс, написанный на другом языке
  - Единая система типов данных
  - Запрет использования указателей
  - Поддержка межязыковой отладки

Шаг 2: Единая система типов данных

Base Class	Visual Basic 2008	C#	C++/CLI
System::Byte	Byte	byte	unsigned char
System::SByte	SByte	sbyte	char
System::Int16	Short	short	short
System::Int32	Integer	int	int
System::Int64	Long	long	long long
System::Single	Single	float	float
System::Double	Double	double	double
System::String	String	string	String^
System::Decimal	Decimal	decimal	Decimal

Шаг 2: C++/CLI- язык C++ для .Net

- C++/CLI – это доработка стандартного языка C++ для работы со средой .Net
  - Новые типы данных
  - Новые конструкции языка
  - Новые операторы
- Позволяет использовать и «обычные» имеющиеся бибилотеки, C++ код, dll, Win32 API
- Позволяет работать с библиотекой классов .Net, что сильно упрощает работу по созданию пользовательского интерфейса, работу с файлами и сетевыми ресурсами

USB

- USB предназначен для расширения функциональности компьютера за счет подключения разнообразных устройств
  - Автоматическое распознавание и конфигурация (Plug&Play)
  - Легкое расширение с использованием HUB-ов
  - Питание устройств от сети
  - Данные защищены контрольной суммой (CRC). Разрушенные данные автоматически повторяются
  - 4 возможные скорости (в Мбод):
    - Low – 1.5
    - Full – 12
    - High – 480
    - Super – 5000 (только USB 3.0)

USB: Типы устройств

- Peripheral (также называется «Функция»)
  - Обеспечивает конкретную функциональность для Host
- Hub
  - Транслирует трафик (в обоих направлениях), управляет питанием
- Compound Device
  - Содержит Hub + одну или более функций
- Composite Device
  - Имеет несколько интерфейсов, работающих одновременно
  - Host содержит драйвер для каждой функции
  - Например: видеокамера имеет два активных интерфейса – аудио и видео

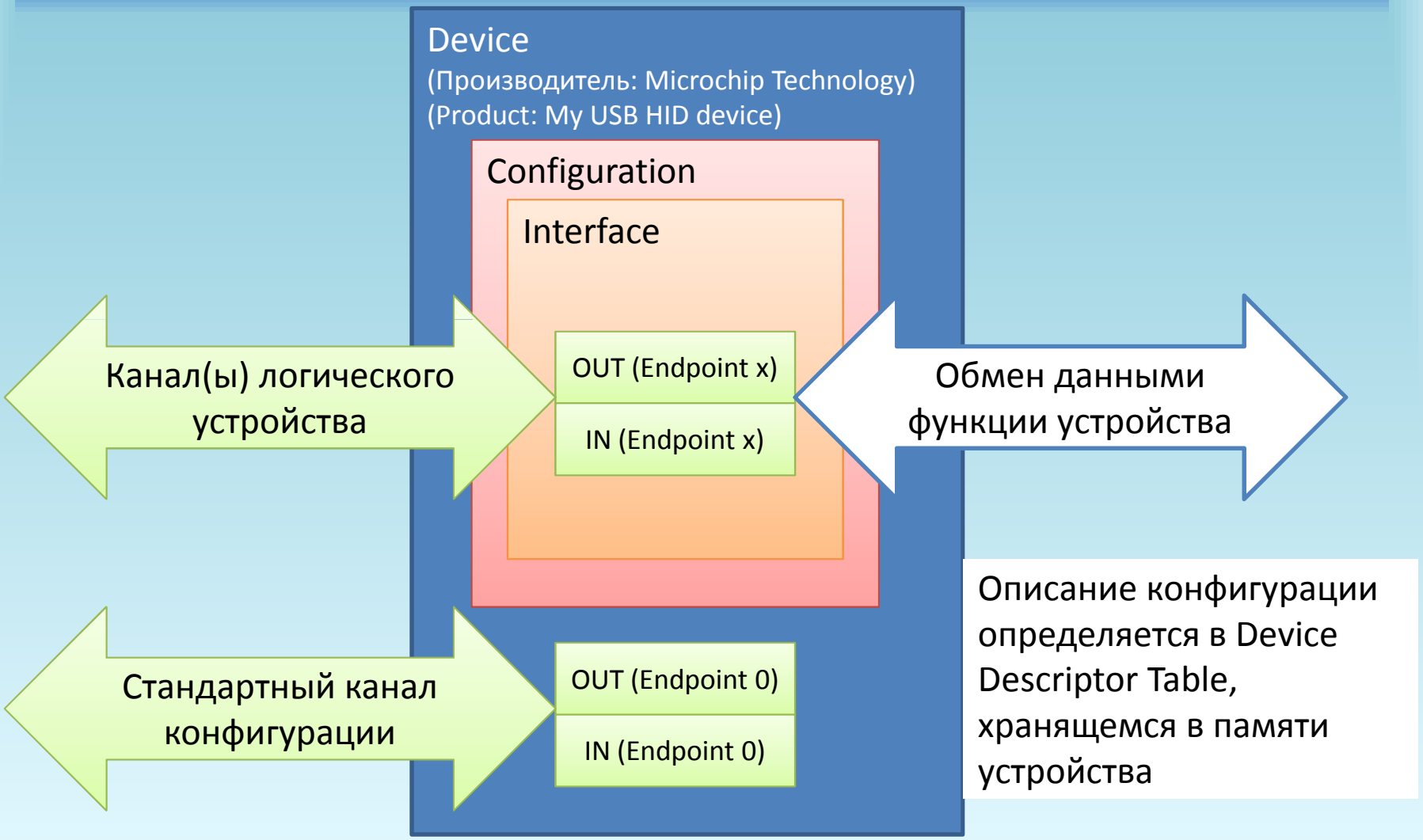
USB: Основы

- USB – это шина с опросом по принципу один «Master» + несколько «Slave»
- Физическая топология:
  - 1 Host controller & Root Hub
  - До 5 Hub
  - До 126 устройств
- Логическая топология:
  - Одноранговая звезда
  - Host работает со всеми логическим устройствами так, как если бы они были подключены непосредственно к Root Hub

USB: Конечные точки (Endpoints)

- Конечная точка – это буфер памяти в USB устройстве, через которое происходит обмен данными с Host
- Конечная точка может быть входной (данные в сторону Host) или выходной (данные в сторону Device)
- Одна пара конечных точек образуют один коммуникационный канал (pipe)
- Максимум 16 каналов (16 входных и 16 выходных точек) на одно устройство
- EP0 – Коммуникационный канал по умолчанию. Используется для конфигурации. Присутствует во всех устройствах

USB: Логическое устройство



USB: Обмен данными

- Обмен данными происходит по инициативе Host.
  - При передаче данных в сторону Host они хранятся в буфере конечной точки устройства пока Host их не запросит
- Host разбивает передачу на фреймы времени
  - Время фрейма определяется типом USB подключения (LowSpeed, FullSpeed, HighSpeed)
- В каждом фрейме передается несколько транзакций
  - Одна транзакция осуществляет обмен (прием или передачу) пакетом с одной конечной точкой
  - Величина пакета определяется размером буфера конечной точки
- Одна передача (Transfer) данных разбивается на несколько транзакций, если длина данных больше буфера конечной точки.
- Режим передачи конечной точки определяет как и когда передаются транзакции во фреймах

USB: Режимы передачи

- Control
  - Для передачи управляющих сигналов и установочных пакетов.
  - Используется для конфигурации устройства
- Interrupt
  - Передача данных гарантирована и определена по времени
  - Максимально одна транзакция на фрейм для одной конечной точки
  - Максимально достижимая скорость 64 Кбайт/с для Full Speed
- Bulk
  - Передача данных гарантирована, но не определена по времени.
  - Передача осуществляется, если на данный момент осталось свободное время после передач другого типа в данном фрейме
  - Максимально достижимая скорость 1216 Кбайт/с для Full Speed
- Isochronous
  - Передача данных с максимальным приоритетом, но не гарантирована . (Кадры могут теряться или портиться)
  - Передача занимает все свободное время во фрейме
  - Максимально достижимая скорость 1023 Кбайт/с для Full Speed

USB: Классы устройств

- Класс устройства определяет механизм взаимодействия между Host и Device
- Класс устройства, определяет:
  - Количество конечных точек
  - Размер буферов конечных точек
  - Режимы передачи конечных точек
  - Протокол обмена данными
- Host загружает соответствующий драйвер устройства в зависимости от класса устройства и работает с устройством через этот драйвер
- Классы устройства могут быть пользовательскими и стандартными
  - Примеры стандартных классов:
    - HID – Human Interface Device
    - MSD – Mass Storage Device
    - CDC – Communication Device Class
  - Стандартные классы используют стандартные драйверы (не требуется установка специальных драйверов)

USB: Сравнение классов устройств

Features	HID	CDC	MCHPUSB	WinUSB	LibUSB
Driver support built into Windows	Yes	Need .inf	No	Need .inf	No
64-bit PC Support	Yes	Yes	Yes	Yes	Yes
XP Ready	Yes	Yes	Yes	Yes	Yes
7 Ready	Yes	Yes	Yes	Yes	Yes
Transfer Types for user's data					
Control	Yes	No	Yes	Yes	Yes
Interrupt	Yes	No	Yes	Yes	Yes
Isochronous	No	No	Yes	No	Yes
Bulk	No	Yes	Yes	Yes	Yes
Max Speed	64 KB/s	~80 kB/s	~1.0 MB/s	~1.0 MB/s	~1.0 MB/s

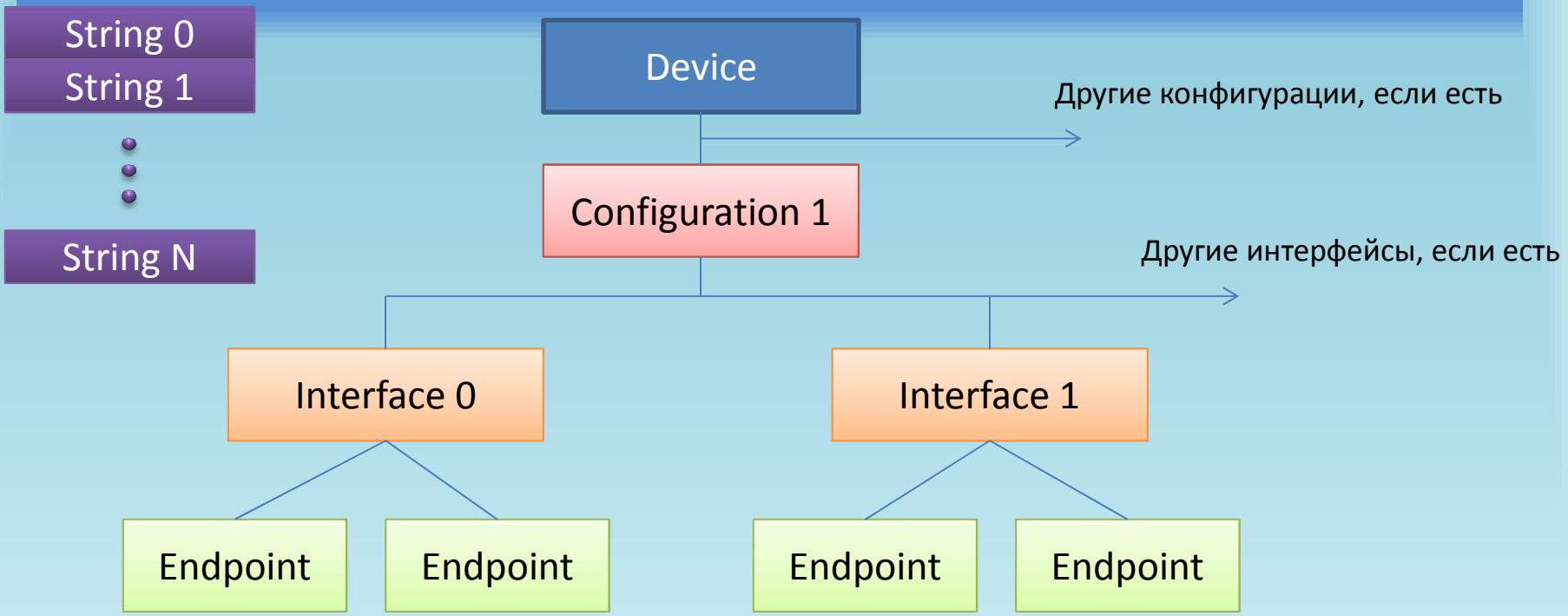
Лабораторная работа: Шаг 3

- Создание USB устройства
  - Аппаратная реализация: PIC32 Ethernet Starter Kit
  - Реализация USB стека: Microchip Application Library
  - Подключение USB: Full Speed
  - Класс устройства - HID
  - Задача: Принимать команды от Host, обрабатывать полученные данные, выдавать ответы.

Шаг 3: USB стек Microchip

- Поддерживает USB 2.0 low speed и full speed
- Поддерживает семейства PIC18, PIC24, dsPIC DSC, PIC32
- Работа поллингом или по прерыванием
- Поддержка классов Device:
  - Audio, CCID, CDC, HID, MSD, PHDC, Custom
- Поддержка Host:
  - PIC24F/E, dsPIC33E, PIC32
  - Клиентские драйверы для CDC, Charger, Custom, HID, MSD, Printer
- Поддержка OTG

Шаг 3: Дескрипторы



Дескрипторы обычно хранятся в энергонезависимой памяти

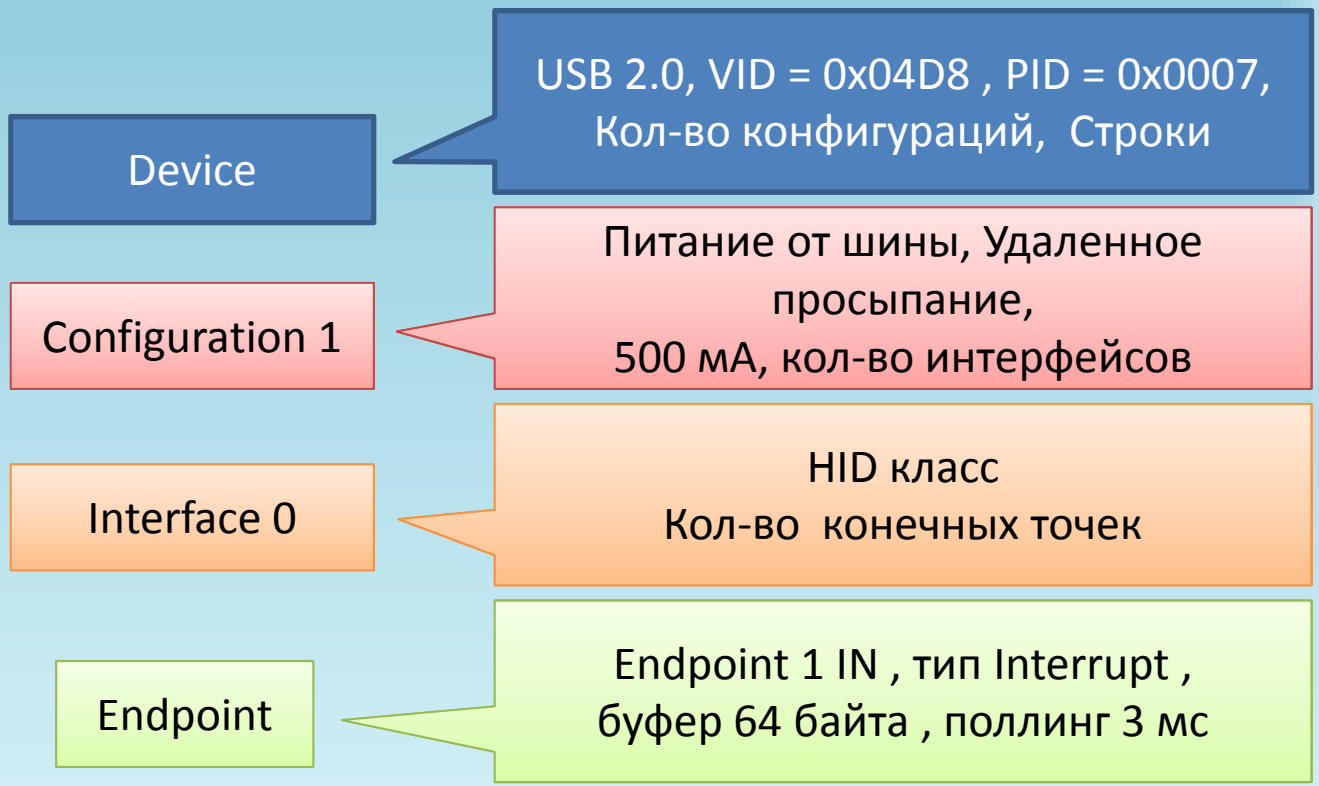
Шаг 3: Пример дескрипторов

String 0
Производитель

Microchip

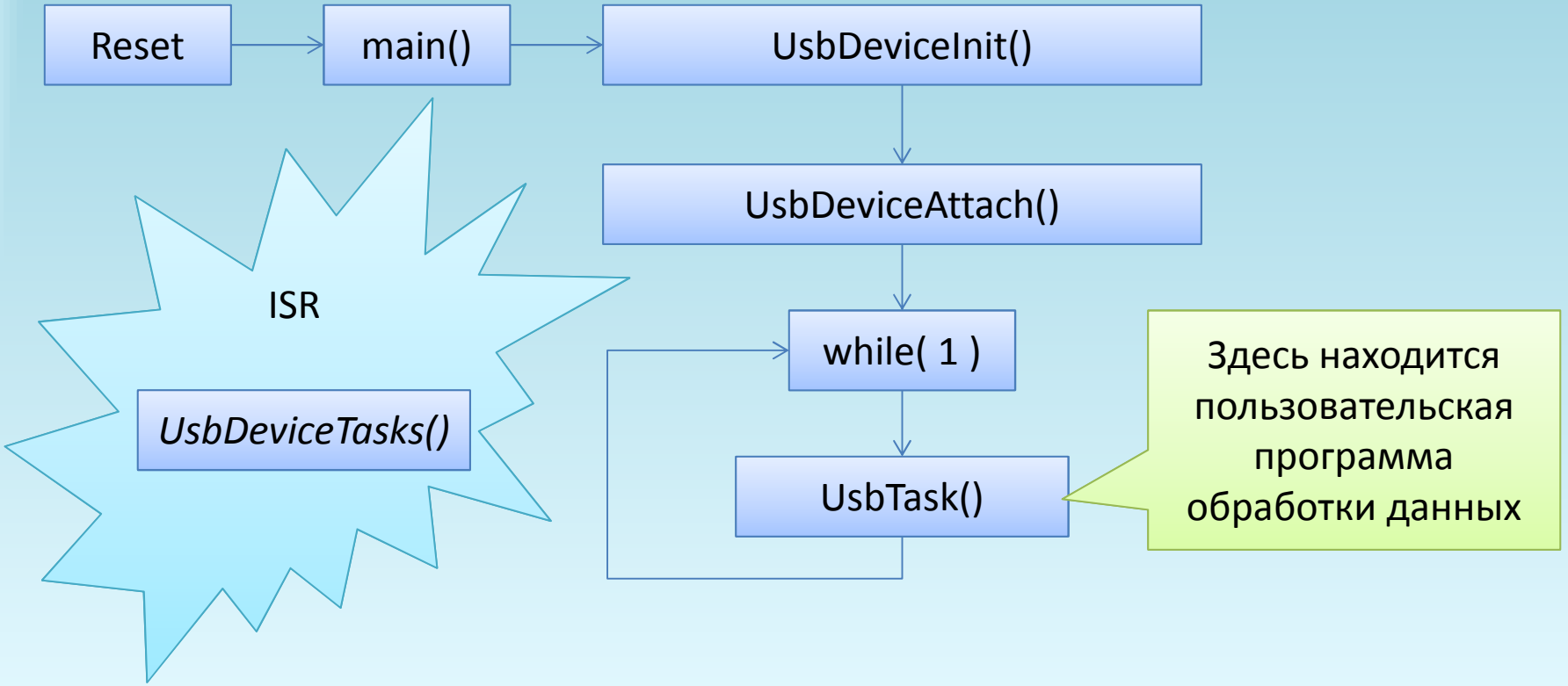
String 1
Продукт

HID device demo



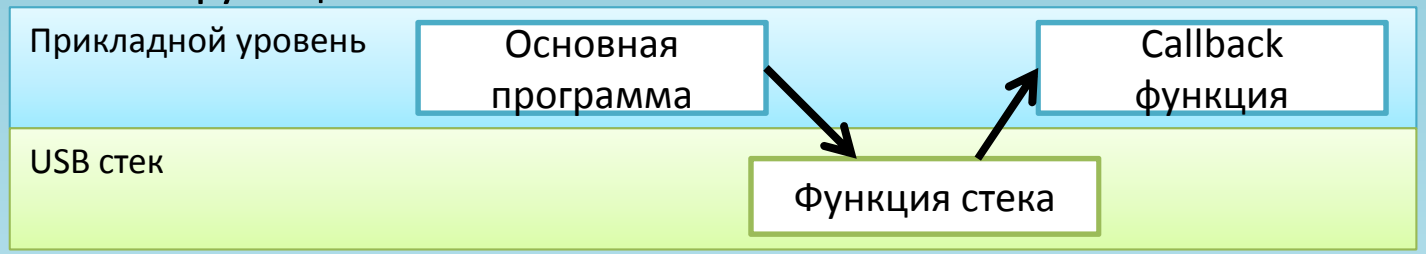
Шаг 3: Выполнение программы

- При конфигурации стека для работы по прерыванием задача USBDeviceTasks() выполняется в контексте прерывания



Шаг 3: Функция Callback

- USB стек, в ответ на различные возникающие события, вызывает Callback функцию



- Вы можете, модифицируя эту функцию управлять реакцией на USB события
- Прототип функции:

  BOOL USER_USB_CALLBACK_EVENT_HANDLER
  (USB_EVENT event, void * pdata, WORD size)
- Параметры:
  - USB_EVENT event – тип события
  - void *pdata – указатель на данные события
  - WORD size – длина данных события

Шаг 3: USB события

- EVENT_NONE - нет события
- EVENT_TRANSFER - USB передача завершена
- EVENT_SOF - принят запрос USB на начало передачи
- EVENT_RESUME - переход устройства в режим resume
- EVENT_SUSPEND - переход устройства в режим suspend
- EVENT_RESET - переход устройства в режим bus reset
- EVENT_STALL - возникла ситуация stall
- EVENT_SETUP - принят пакет setup
- EVENT_CONFIGURED - уведомление, что принята команда SET_CONFIGURATION()
- EVENT_SET_DESCRIPTOR - принят запрос SET_DESCRIPTOR()
- EVENT_EP0_REQUEST - принят запрос конечной точки 0, которую стек еще не знает как обрабатывать. См. документацию на драйвер данного класса о том, что делать при приеме такого запроса
- EVENT_BUS_ERROR - Ошибка на линии USB

Шаг 3: Класс HID Device Class

- Для работы устройств с реакцией на действия человека
  - Небольшая скорость и объем данных
  - Хорошая скорость реакции
- Использует только передачи типа Interrupt
- Транзакции до 64 байт
- Максимум 1 транзакция на фрейм
- До 1000 фреймов в секунду (64000 байт/с)
- Данные передаются «отчетами» («Reports»)
- Дескриптор отчета сообщает host-у формат данных, приходящих от устройства
- Host посылает запрос дескриптора отчета при подключении устройства

Шаг 3: «Стандартный» дескриптор отчета

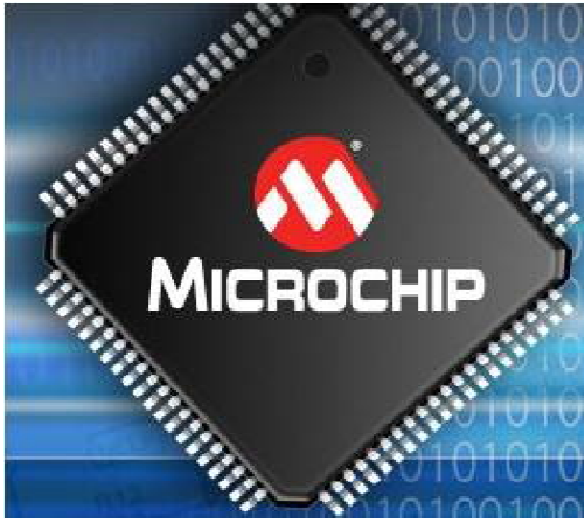
```
//Class specific descriptor - HID
ROM struct{BYTE report[HID_RPT01_SIZE];}hid_rpt01={
{
    0x06, 0x00, 0xFF,          // Usage Page = 0xFFFF (Vendor Defined)
    0x09, 0x01,                // Usage (Vendor Usage 1)
    0xA1, 0x01,                // Collection (Application)
    0x19, 0x01,                //   Usage Minimum (0)
    0x29, 0x40,                //   Usage Maximum (64)
    0x15, 0x00,                //   Logical Minimum (0)
    0x26, 0xFF, 0x00,          //   Logical Maximum (255)
    0x75, 0x08,                //   Report Size 8 bits per report.
    0x95, 0x40,                //   Report Count 64 bytes per report.
    0x81, 0x02,                //   Input (Data, Var, Abs)
    0x19, 0x01,                //   Usage Minimum (Vendor Usage = 0)
    0x29, 0x40,                //   Usage Maximum (Vendor Usage = 64)
    0x91, 0x02,                //   Output (Data, Var, Ads)
    0xC0}                      // End Collection
};
```

Лабораторная работа: Шаг 4

- Использование HID Class DLL

Шаг 4: HID Class DLL

- .Net сборка “HID Class.dll”
- Берет на себя всю организацию обмена данными Win32 с HID устройством
- Пространство имен HIDClass::MCHPHIDClass
- Четыре основные функции:
  - void HIDClassInit(VID, PID, len, timeout);
    - timeout - опционально
  - bool HIDWriteReport(buffer, len);
    - Для работы функции в устройстве должен быть реализован «стандартный» дескриптор отчета
  - bool HIDReadReport(buffer)
    - Функция неблокирующая.
    - Для работы функции в устройстве должен быть реализован «стандартный» дескриптор отчета
  - bool HIDIsConnected();
    - Запрос ОС. Никакого обмена с устройством не происходит.



Microchip's Russia MASTERS

Спасибо за внимание !