МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ
«ИЖЕВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
имени М.Т. КАЛАШНИКОВА»

Разработка программного обеспечения для макета SDK4

Описание программы

Листов 31

Выполнил: студент гр. М02-781-1
Симаков В.В.
Проверил: к.т.н.
Петухов К.Ю.

Ижевск 2014

АННОТАЦИЯ

Настоящий документ содержит описание процесса разработки программного обеспечения макета SDK4 и предназначен для студентов или специалистов, намеренных в дальнейшем модернизировать программное обеспечение макета.

Документ описывает выпущенную версию ПО, функциональное назначение, логическую структуру, алгоритмы выполнения.

СОДЕРЖАНИЕ

1. Введение 4

2. Состав оборудования 5

3. Разработка протокола обмена 10

4. Разработка системного ПО макета SDK4 13

5. Разработка прикладного ПО для Windows 28

1. ВВЕДЕНИЕ

На текущий момент большинство различных технологических процессов автоматизировано с применением микроконтроллеров. Но зачастую возникает проблема удаленного управления объектом. Данную проблему позволяет решить наличие Ethernet в составе периферии микроконтроллера, посредствам которого возможно осуществление управления и контроля автоматизируемых процессов на расстояние через сеть Ethernet.

2. СОСТАВ ОБОРУДОВАНИЯ

2.1. Краткое описание макета SDK4

Учебный макет SDK4 базируется на 16-разрядном микроконтроллере AM186ES фирмы AMD с архитектурой x86. Данный макет оснащен достаточно большим набором периферийных устройств: Flash памятью на 512 КБ, ОЗУ на 512 КБ, контроллером Ethernet 10BaseT cs8900a, последовательным каналом RS232, сторожевым таймером.

К программным особенностям макета можно отнести наличие во Flash памяти макета стартового загрузчика, выполняющего при запуске макета инициализацию всего периферийного оборудования и переход на точку входа пользовательского приложения. Причем данный стартовый загрузчик может функционировать в двух режимах:

- Инициализация и старт
- Прием и прошивка пользовательской программы

Для переключения между режимами опрашивается состояние «ножки» EXT1 на разъеме J2: когда на EXT1 высокий уровень (перемычка отсутствует) загрузчик функционирует в первом режиме, иначе – во втором.

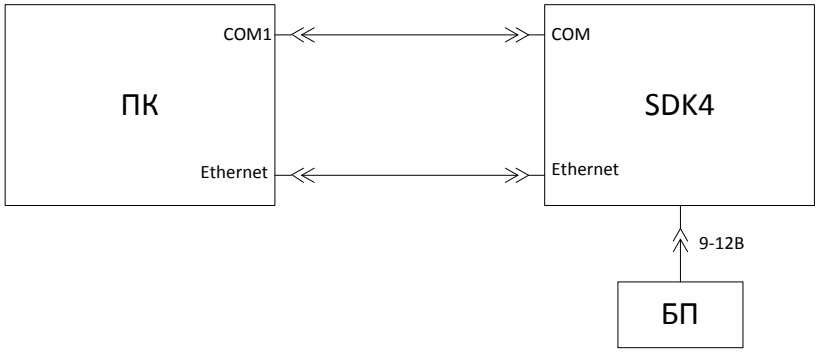
2.2. Требования к ПК

Для организации обмена по Ethernet между ПК и SDK, первый должен иметь в наличии также сетевую карту с поддержкой стандарта 10BaseT. Опционально также требуется наличие COM порта, который необходим для трассирования выполняемой на макете программы в отладочных целях. При отсутствии COM-порта на материнской платы возможно использование платы расширения или конвертера USB-COM.

2.3. Подключение ПК и SDK4

Схема подключения ПК и SDK4 представлена на рисунке. В качестве соединителя COM достаточно использовать двужильный кабель, т.к. кроме линий TxD и RxD другие не используются в программном обеспечении. В качестве Ethernet соединителя используется обычный кроссовер кабель. Подключение всех кабелей рекомендуется осуществлять предварительно отключив обе стороны в целях безопасности и сохранности работоспособности оборудования. Питание SDK подается через внешний источник питания выходным напряжением в диапазоне от 9-12В (следует обратить внимание на полярность, т.к. на разъеме SDK внешний контакт это «плюс», а внутренний штырь «минус»).

Схема соединения ПК и SDK



Рисунок

SDK4 начинает выполнять код загрузчика сразу при подаче питания. Поэтому перед запуском необходимо также предварительно убедиться в отсутствии перемычки на контакте EXT1 разъема J2 для выполнения пользовательского приложения. Для перевода загрузчика в режим загрузки пользовательской прошивки, необходимо обесточить макет, замкнув перемычкой контакт EXT1 и подав вновь напряжение. По окончании процедуры программирования Flash памяти и запуска записанного пользовательского приложения, макет вновь необходимо перезапустить путем повторной процедуры обесточивания-подачи питания.

3. СТОРОННЕЕ ПО

Упростить процесс разработки и отладки программного обеспечения макета SDK4 позволяет набор сторонних утилит.

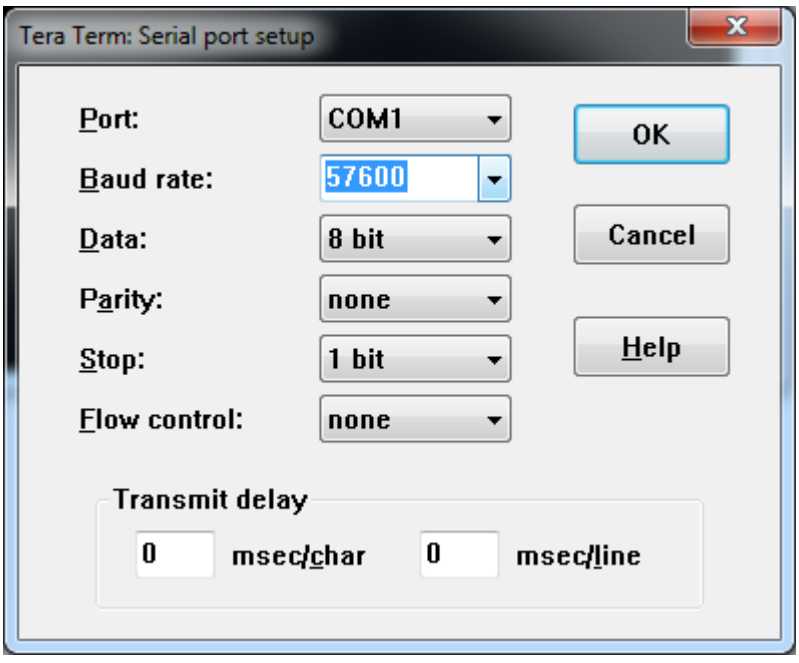
Для отладки используются такой класс программных продуктов, именуемых как мониторы. Данные утилиты позволяет отслеживать траффик по необходимым интерфейсам, что позволяет отладить правильность реализации протоколов обмена, а также выполнять трассировку выполняемой на макете программы.

3.1. COM-монитор

Для трассировки выполняемой на макете программы используется COM-монитор Tera Term. Данная программа отслеживает всю принимаемую информацию по COM-порту. Для этого Tera Term запускается на ПК и конфигурируется согласно рисунку. Данная программа распространяется бесплатно.

Примечание: Tera Term не навязывается как лучшее программное решение COM-монитора, поскольку имеет огромно количество аналогов (PuTTY, COM Port Toolkit) с более широким функционалом.

Настройка COM порта в Tera Term



Рисунок

Скорость 57600 бит/с соответствует скорости, устанавливаемой на макете SDK4 стартовым загрузчиком, поэтому в дальнейшем весь вывод информации по порту в пользовательском

приложении осуществляется только на этой скорости, чтобы избавиться от избыточных переконфигурирований приложения.

При отсутствии аппаратного отладчика использование данной программы облегчает отладку приложения, поскольку каждую выполняемую на макете команду можно снабдить трассировочным сообщением, которое будет отправляться с макета на ПК, далее по логу в терминале Tera Term можно отследить процесс выполнения программы на макете и при наличии ошибки внести соответствующие коррективы.

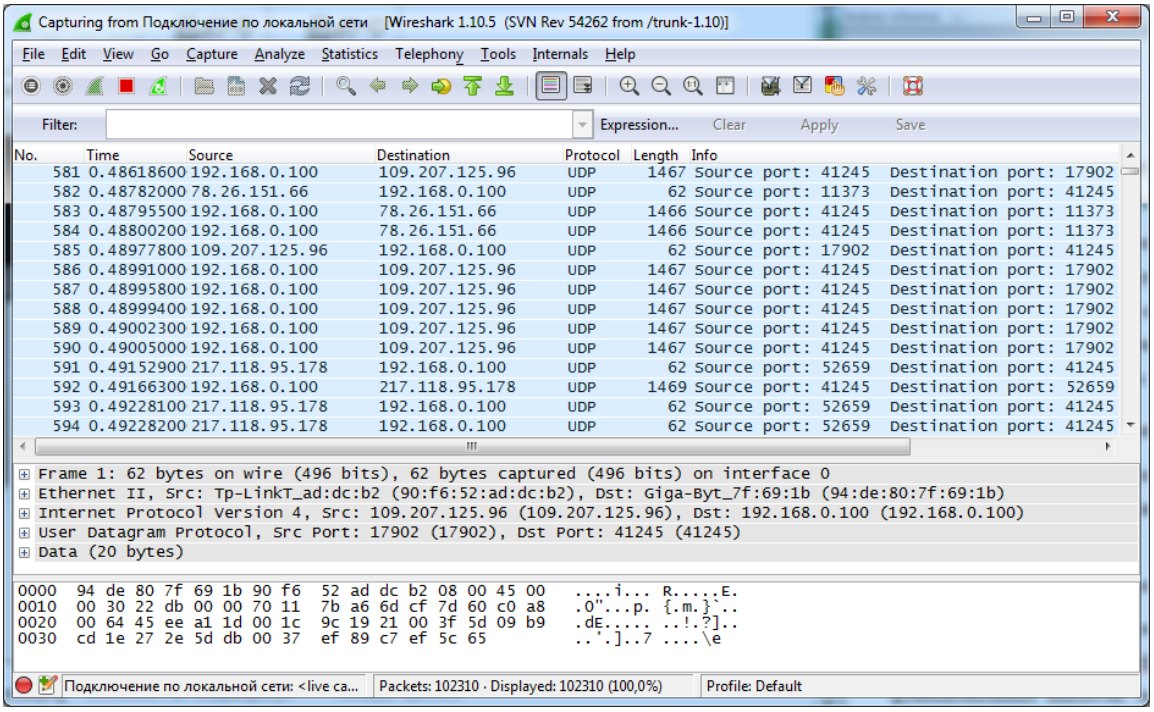
3.2. Ethernet утилиты

Для отладки протокола обмена по Ethernet используется набор сторонних утилит. Данные программы позволяют формировать произвольные Ethernet пакеты и осуществлять прослушку Ethernet канала.

3.2.1. Ethernet-монитор

В качестве программы, осуществляющей прослушку необходимого Ethernet канала, используется утилита WireShark. Данная утилита распространяется бесплатно и не имеет функциональных аналогов, позволяет отображать содержимое принятых пакетов, тип протокола, накладывать фильтры на принимаемые пакеты и много другое. Главное окно программы показано на рисунке .

Основное окно утилиты-монитора Wireshark



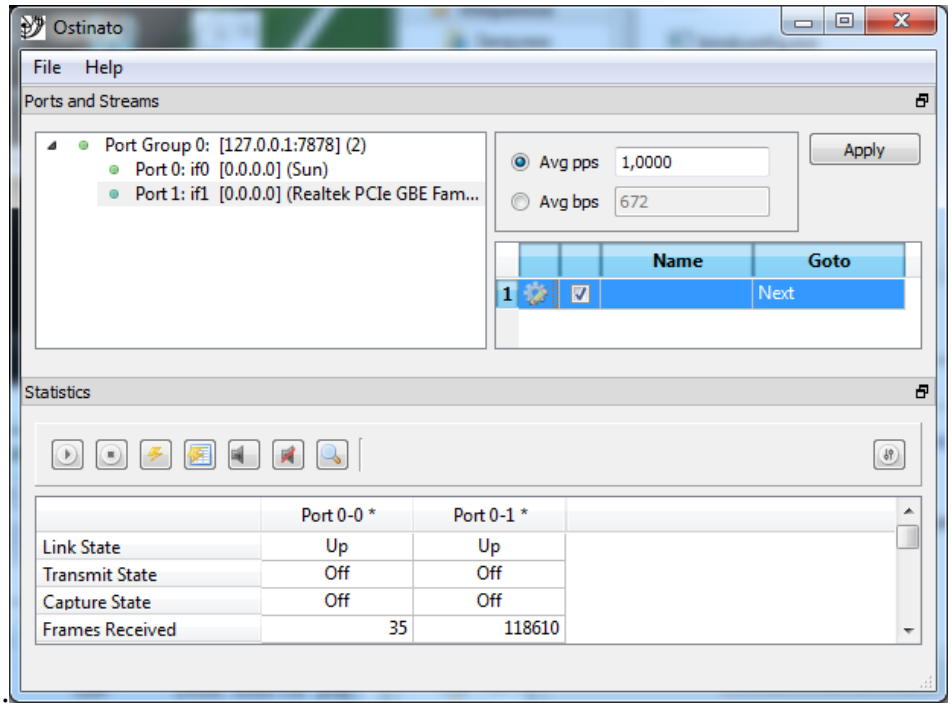
Рисунок

Данный монитор используется для анализа принимаемых пакетов от макета SDK4 и позволяет отладить логику и содержимое данных согласно протоколу обмена.

3.2.2. Формирователь пакетов

В качестве программы, осуществляющей формирование произвольных пакетов по Ethernet и их отправку, используется утилита Ostinato. Данная программа распространяется бесплатно и является уникальной кроссплатформенной программой, позволяющей отправлять в сеть пакеты произвольного содержания, на произвольные адреса. Главное окно программы показано на рисунке

Основное окно утилиты Ostinato



Рисунок

Данная программа позволяет отладить программное обеспечение макета SDK4 в части логики обработки принимаемых данных по Ethernet каналу согласно протоколу обмена.

4. РАЗРАБОТКА ПРОТОКОЛА ОБМЕНА

Протокол обмена устанавливает правила для данных принимаемых и отправляемых между взаимодействующими абонентами.

Обмен построен следующим образом: ПК отправляет командные пакеты, с помощью которых осуществляется управление периферией макета SDK4 (диоды), а также возможность включения/отключения отправки телеметрической информации о состоянии перемычек макета, также его времени. Телеметрия, упакованная в пакет, отправляется макетом с фиксированным периодом.

Далее рассматривается структура командного пакета и пакета с телеметрией.

3.1. Командный пакет

Командный тип пакета отправляется с ПК. Данный тип обмена позволяет управлять встроенной периферией макета SDK4.0, включать/отключать отправку телеметрической информации. Описание командного пакета приведено в таблице .

Таблица – Структура командного Ethernet пакета

Байт	Значение	Описание	Примечание
1-6	DAddr	MAC адрес назначения (SDK4)	
7-12	SAddr	MAC адрес отправителя (ПК)	
13-14	Length	Размер пакета	
15-16	OpCode	Код операции	Типы операций: - Красный диод (0) - Зеленый диод (1) - Желтый диод (2) - Телеметрия (3)
17-20	ParamValue	Значение параметра	Параметр: - 1, вкл - 0, выкл
21-29	Time	Время	Время ПК в строковом формате: «ЧЧММССММММ»
30-60*	Reserved	Резерв	Данные байты используются для дополнения пакета до минимально допустимого

Примечание: серым цветом выделяется заголовок Ethernet пакета, белым – значащие поля согласно протоколу. * - минимально допустимый размер Ethernet пакета составляет 64 байта, из которых 4 байта КС формируются аппаратно.

3.2. Пакет телеметрии

Пакет телеметрии содержит информацию о состоянии макета SDK4. Данная информация отправляется с макета с периодичностью в примерно 2 секунды (неточность данного времени определяется реализацией задержки по тактам команд на счетчике). При включении макета, по умолчанию, отправка телеметрии выключена, для её активации необходимо отправить на макет командный пакет с запросом включения формирования и отправки телеметрии.

Таблица – Структура Ethernet пакета телеметрии

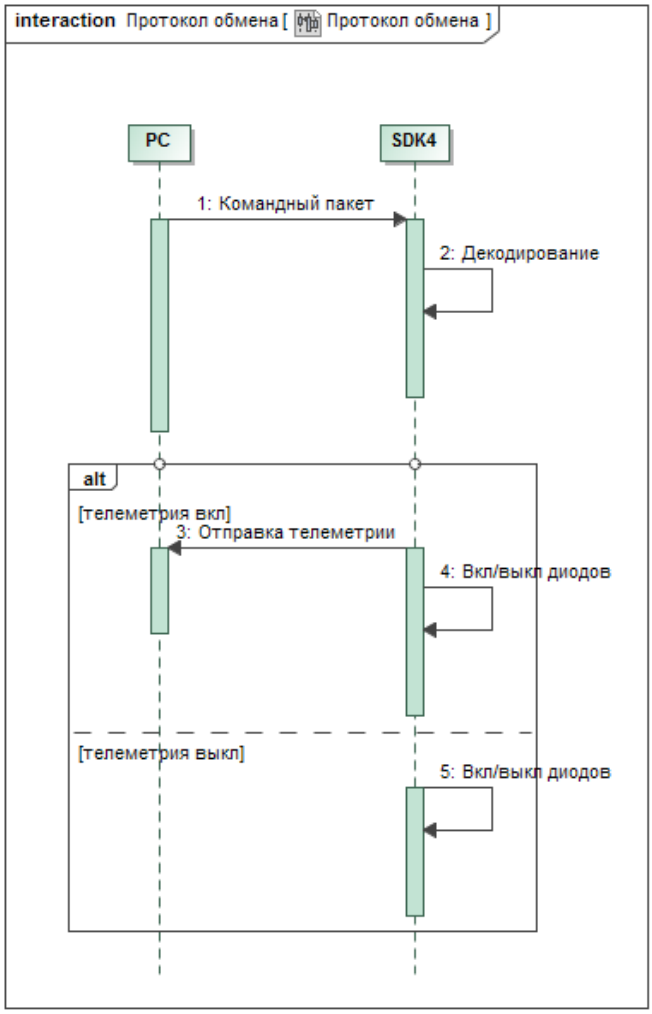
Байт	Значение	Описание	Примечание
1-6	DAddr	MAC адрес назначения (ПК)	
7-12	SAddr	MAC адрес отправителя (SDK4)	
13-14	Length	Размер пакета	
15	RLed	Состояние красного диода	0 – выкл 1 – вкл
16	GLed	Состояние зеленого диода	0 – выкл 1 – вкл
17	YLed	Состояние желтого диода	0 – выкл 1 – вкл
18	JmpModem	Состояние «ножки» Modem	0 – выкл 1 – вкл
19	JmpRS	Состояние «ножки» RS	0 – выкл 1 – вкл
20	JmpLock	Состояние «ножки»Lock	0 – выкл 1 – вкл
21	JmpMIF	Состояние «ножки» MIF	0 – выкл 1 – вкл
22	JmpProg	Состояние «ножки» Prog	0 – выкл 1 – вкл
23	JmpExt1	Состояние «ножки» Ext1	0 – выкл 1 – вкл
24	JmpExt2	Состояние «ножки» Ext2	0 – выкл 1 – вкл
25	Telem	Состояние телеметрии	0 – выкл 1 – вкл
26-29	Time	Время	Время формируемое таймером SDK4 в мс * 10
30-60*	Reserved	Резерв	Данные байты используются для дополнения пакета до минимально допустимого

Примечание: серым цветом выделяется заголовок Ethernet пакета, белым – значащие поля согласно протоколу. * - минимально допустимый размер Ethernet пакета составляет 64 байта, из которых 4 байта КС формируются аппаратно.

4.3. Диаграмма обмена

На рисунке представлены диаграмма последовательности, на которой изображен в общем виде процесс взаимодействия двух сторон – ПК и SDK4.

Диаграмма последовательности обмена ПК и SDK4



Рисунок

Пользователь ПК инициирует обмен, отправляя командный пакет на приемную сторону – SDK4. Далее принятый пакет декодируется программным обеспечением пакета и анализируется командная посылка. В зависимости от принятой команды выполняется соответствующее действие на макете. Далее анализируется поле состояния телеметрии в зависимости от которого принимается решение об отправке пакета телеметрии. Если телеметрия была включена командной посылкой, то в дальнейшем макет начинает периодическую отправку телеметрических пакетов.

5. РАЗРАБОТКА СИСТЕМНОГО ПО МАКЕТА SDK4

5.1. Описание программного обеспечения

Разрабатываемого системное программное обеспечение макета SDK4.0 должно выполнять обмен по Ethernet. Обмен регламентируется вышеописанным протоколом. Макет должен принимать и обрабатывать от ПК командные пакеты. Реакцией на команду должно быть управление светодиодами, установленными на макете. Кроме того, по командному запросу должна быть реализована выдача пакетов телеметрии с фиксированным периодом. Содержание телеметрии также определяется протоколом.

5.2. Описание среды разработки

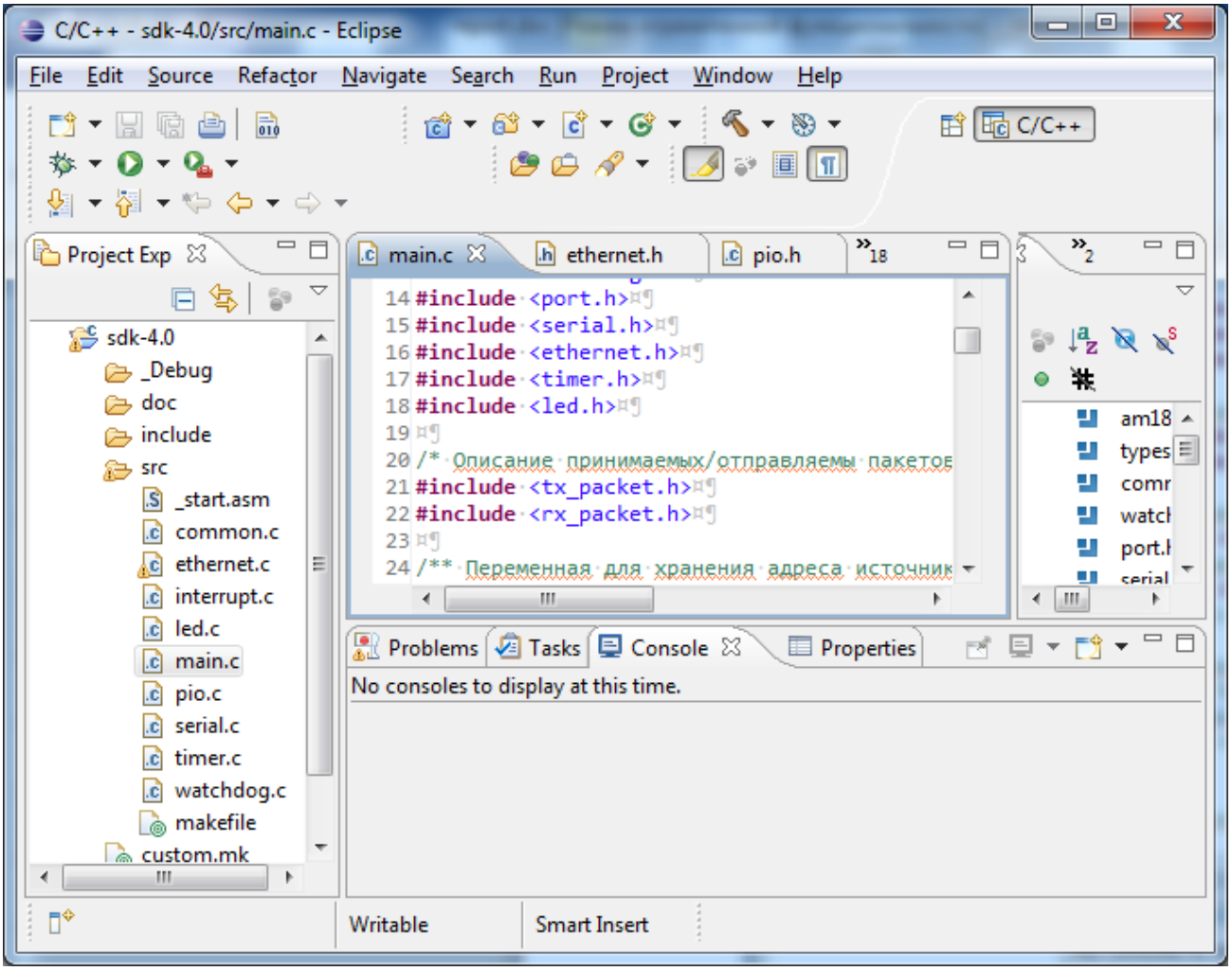
Под интегрированной средой разработки понимается специальная программа, средствами которой осуществляется:

1) Написание исходных кодов программы
2) Компиляция приложения
3) Линковка объектных файлов
4) Отладка приложения

Таким образом, большая часть жизненного цикла разрабатываемого программного обеспечения может быть облегчена с помощью среды разработки. Все этапы по созданию программы могут выполняться и с помощью простого блокнота и консоли, но это в разы снижает производительность по созданию программного продукта.

В качестве интегрированной среды разработки используется Eclipse Indigo. Достоинством данной интегрированной среды разработки является удобный интерфейс, возможность расширения функционала с помощью расширяющих плагинов, с помощью которых можно настроить среду под собственные нужды. Основное окно IDE представлено на рисунке ниже.

Окно интегрированной среды разработки Eclipse Indigo



Рисунок

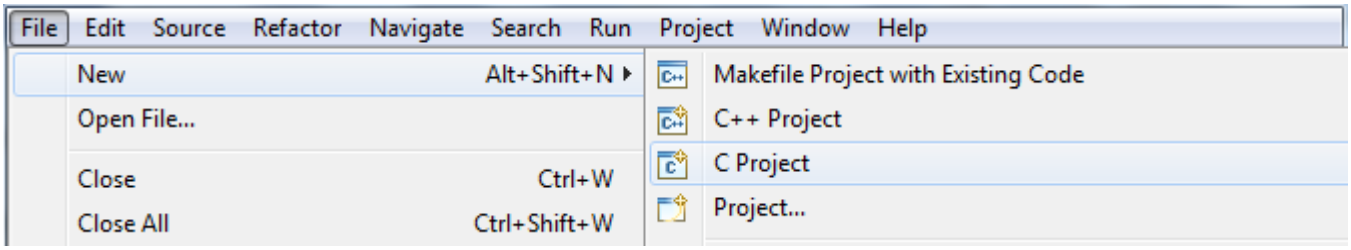
Данная среда разработки не навязывается как самый оптимальный вариант, а лишь рекомендуется к использованию. Для ускорения процесса разработки необходимо потратить определенное время на настройку среды. Данное приложение распространяется бесплатно.

Скачать IDE можно по ссылке [] – данная среда уже включает набор плагинов, необходимых для разработки программ на языках C/C++.

Подготовка IDE выполняется следующим образом:

1. Для того, чтобы создать проект необходимо выполнить цепочку File > New > C Project согласно рисунку.

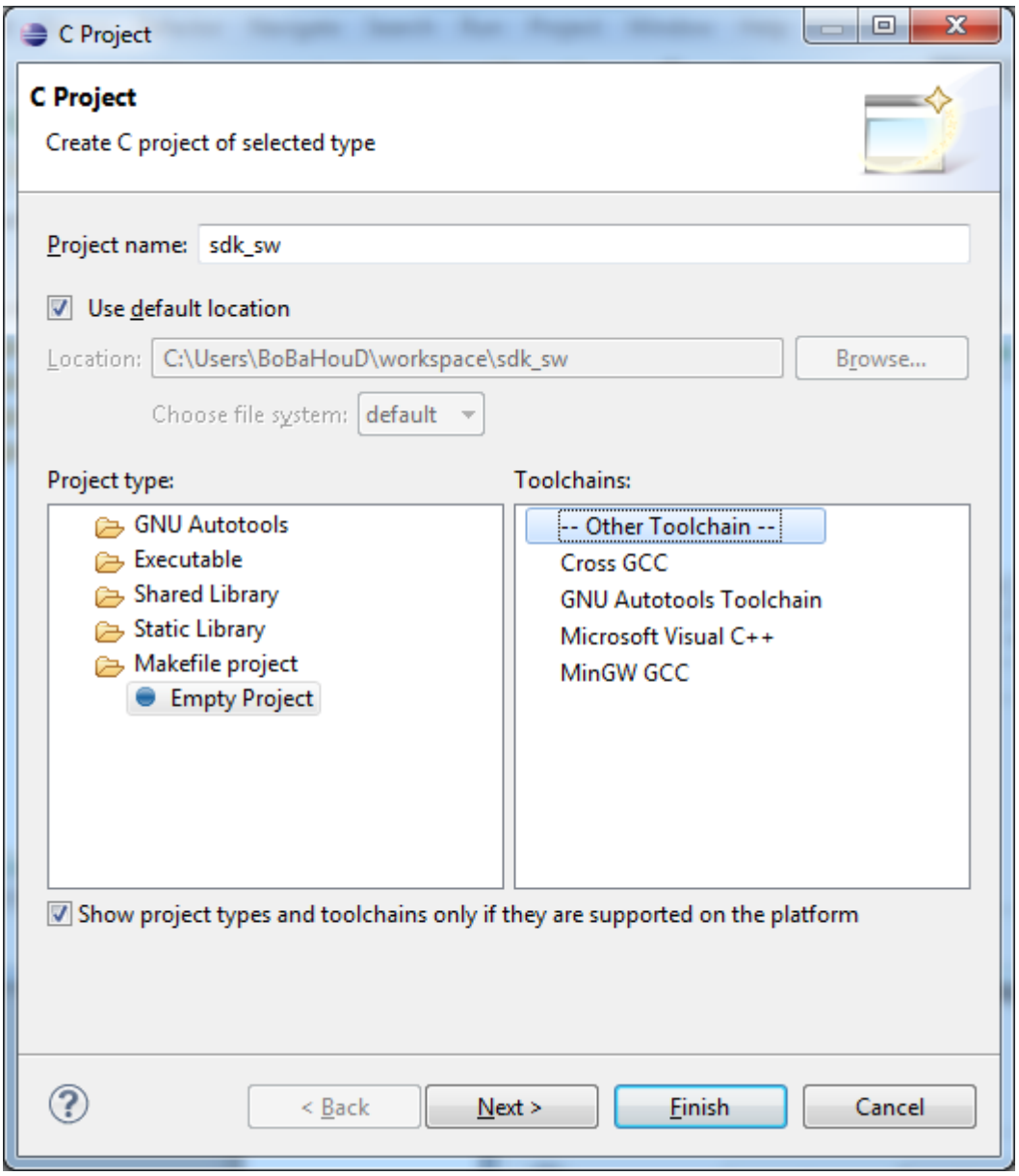
Создание нового C проекта



Рисунок

2. Далее задается имя проекта, проект выбирается как Makefile Empty Project, toolchain не указывается, поскольку утилиты компиляции будут добавлены позже. В конце выбирается пункт Finish.

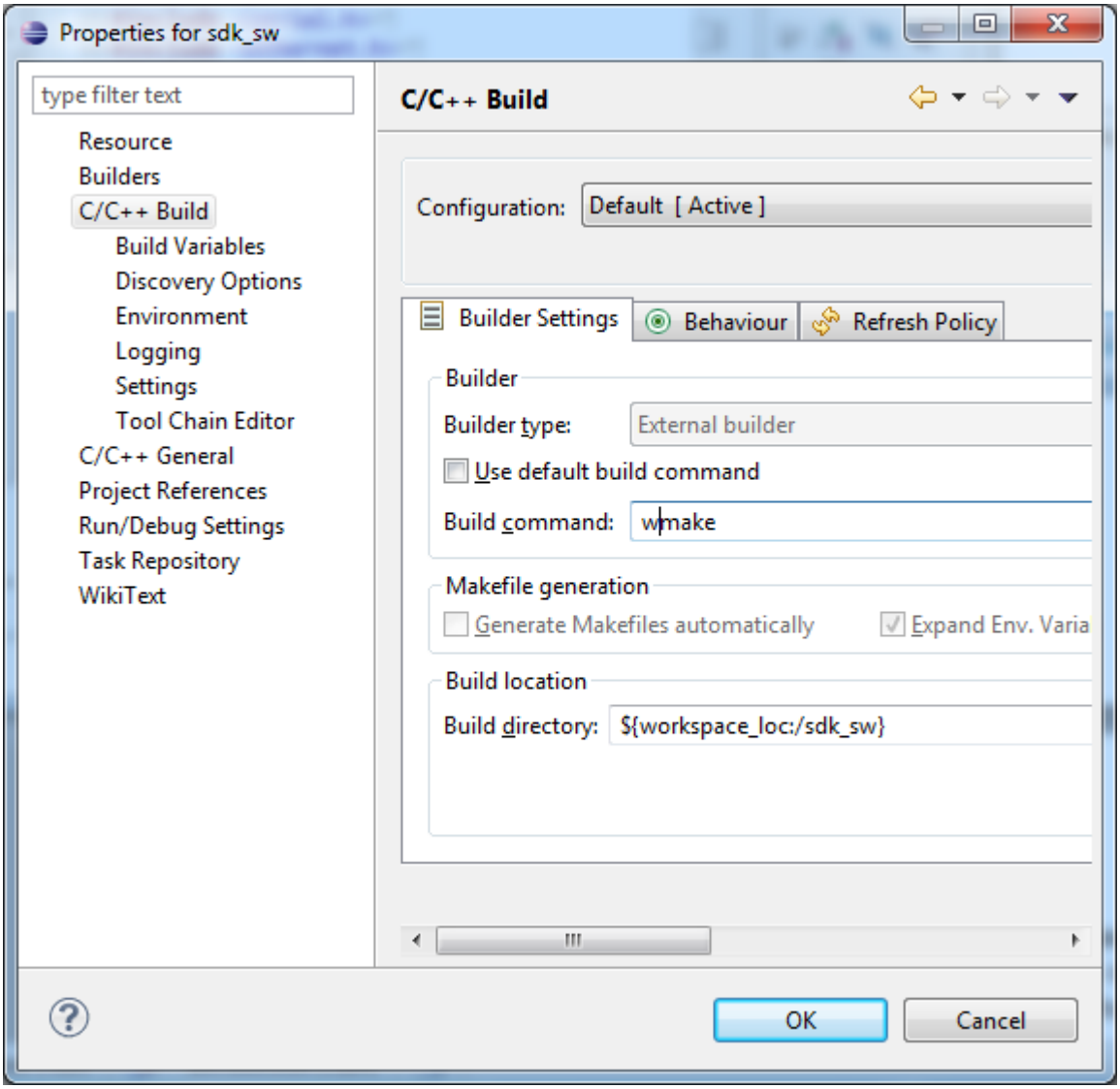
Настройки C Project проекта



Рисунок

3. Далее в Project Explorer необходимо убедиться, что появился созданный проект. Теперь необходимо выполнить настройку проекта. Для этого следует зайти в свойства проекта, вызвав контекстное меню по щелчку правой кнопки мыши или через меню Project > Properties. Необходимо изменить настройки компиляции изменив *Build command* на **wmake** согласно рисунку.

Настройки проекта



Рисунок

После успешного выполнения всех шагов по настройке проекта необходимо выполнить установку утилит компиляции и линковки, называемых *toolchain*.

5.3. Утилиты компиляции

Под утилитами компиляции подразумевается набор таких программ, как компилятор, линковщик, сборщик. Под компилятором подразумевается программа, преобразующая файлы с .c расширением в объектные файлы, выполняемые на целевой платформе. Под линковщиком подразумевается программа, выполняющая связь объектных файлов и создание одного единого исполняемого файла. Все перечисленные утилиты входят в состав Watcom C/C++. Особенностью

данного компилятора является формирование быстрого исполняемого кода при небольшом его размере. Поэтому данный компилятор идеально подходит для компиляции приложений для микроконтроллеров. Данный компилятор предназначен для генерации кода для платформ x86. Используется версия 11.0, поскольку данная версия была актуальна на момент разработки аппаратной части макета SDK4.0. Компилятор Watcom не полностью поддерживает объектно-ориентированные возможности языка C++, что не является существенным недостатком, поскольку разработка программного обеспечения для макета SDK4 будет вестись на языке C.

Далее приводится последовательность действий по установке Watcom.

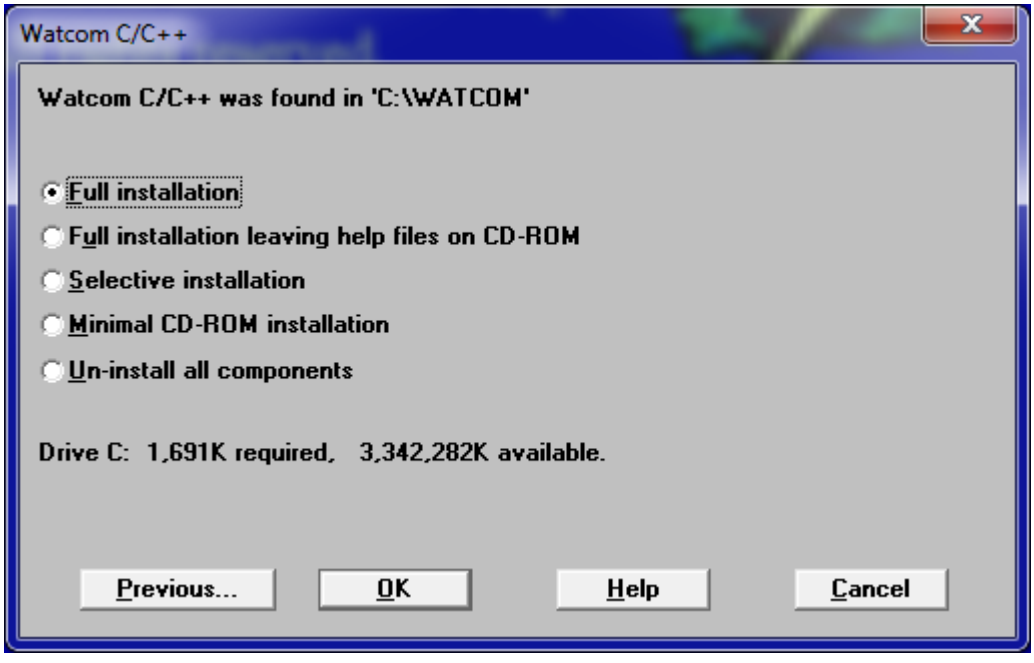
1. При установке рекомендуется оставлять путь установки по умолчанию, согласно рисунку.

Путь установки Watcom

Рисунок

2. В опциях установки необходимо установить пункт полной установки. Далее необходимо дождаться завершения установки компилятора. Опции установки показаны на рисунке.

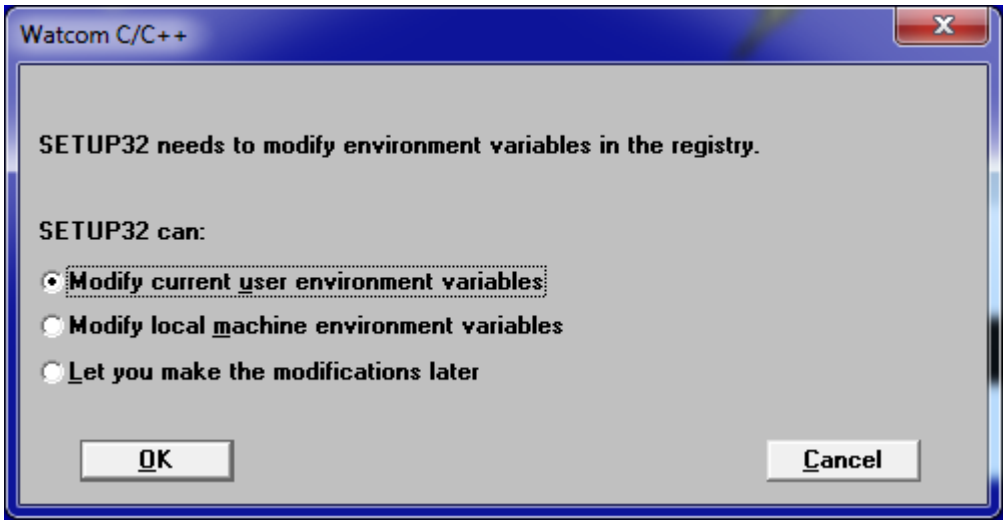
Опции установки Watcom



Рисунок

3. По окончании установки необходимо, чтобы путь до бинарных файлов, был добавлен в переменные среды, для этого необходимо отметить пункт согласно рисунку.

Опции по окончании установки



Рисунок

Когда установка закончена, можно приступать к разработке программного обеспечения для макета SDK4.0.

5.3.1. Компилятор Watcom

Компилятор и линковщик инкапсулируются в одном исполняемом файле *wcc.exe*. В данном разделе необходимо упомянуть о ключах, используемых при линковке/компиляции выходного файла, запускаемого на целевой платформе SDK4.

Используемые директивы компиляции:

Таблица – Директивы компилятора

№ п/п	Директива	Описание	Примечание
1	-1	Использование инструкций для серии процессоров 186 и 188	Процессор AM186ES поддерживает только этот набор инструкций
2	-fpc	Операции с плавающей точкой будут выполняться программно	Аппаратно поддерживается только операции с целыми числами
3	-w4	Установка уровня оповещения ошибок компиляции	4 уровень самый высокий
4	-e25	Максимальный уровень ошибок компиляции	При превышении макс. значения, компиляция прекращается

№ п/п	Директива	Описание	Примечание
5	-oi	Генерация инлайн функций некоторых библиотек	Оптимизация для ускорения кода
6	-d2	Уровень отладочной информации	2 – самый высокий
7	-ms	Модуль памяти small	
8	-bt=dos	Определение целевой платформы как MS-DOS	Утилита загрузки исполняемых файлов на макет, поддерживает только данный тип файлов

Примечание: с подробным описанием всех директив можно в официальной документации, поставляемой вместе с watcom.

Используемые директивы линковки:

Таблица – Директивы линковщика

№ п/п	Директива	Описание	Примечание
1	op q	Скрытие сообщений линковки	Не рекомендуемая опция
2	sys dos	Генерация выходного файла как DOS EXE	

5.3.2. Система сборки wmake

Система сборки целевого приложения почти ничем не отличается от GNU Make за рядом небольших ограничений и особенностей. Для написания собственных makefile’ов рекомендуется ознакомиться с документацией на GNU make.

Для компиляции приложения выстроена следующая иерархия make-файлов: в корне каталога с исходными файлами располагается *makefile* (основные цели сборки), *custom.mk* (константы и утилиты), в каталоге *src* еще один *makefile* согласно таблице:

Таблица – Файлы сборки приложения

№ п/п	Файл	Описание	Примечание
1	*./makefile*	Цели:	По умолчанию сборка

№ п/п	Файл	Описание	Примечание
		all – сборка приложения clean – очистка каталога сборки	выполняется командой wmake, приложение собирается в каталоге _Debug
2	*./custom.mk*	Настройка окружения	Флаги компиляции/линковки, директории сборки, имени компилятора
3	*./src/makefile*	Перечисление объектных файлов и правил компиляции	Добавление нового файла выполняется через переменную OBJS

Примечание: для сборки потребуется утилита mv.exe, установить можно с помощью пакета MSYS (порт UNIX окружения).

5.4. Разработка приложения

При разработке приложения применялся модульный тип разбиения функционала, в соответствии с которым было решено выделять на каждое физическое устройство макета один файл с исходным кодом (не включая заголовочные), в котором реализуется логика управления, настройки устройства. В конечном итоге был определён следующий список устройств и файлы исходного кода:

Таблица - Сводная таблица зависимости функционала устройств с файлами исходного кода

№ п/п	Файл	Устройство	Описание
1	_start.asm	-	Стартовый код, выполняемый при запуске макета, после исполнения которого выполняется переход в точку входа **main**
2	common.c		Функции общего назначения, используемые в большинстве модулей. Включают функции мониторинга перемычек, вывода трассировочных сообщений, смены порядка байт в слове/полуслове.
3	ethernet.c	Контроллер Ethernet cs8900a	Функции управления контроллером Ethernet: инициализация, прием/отправка по прерываниям

№ п/п	Файл	Устройство	Описание
4	interrupt.c	Контроллер прерываний	Функции управления контроллером прерываний: установка векторов прерываний, включение/выключение прерываний
5	led.c	Диоды макета	Функции управления диодами макета: включение/выключение, инициализация
6	main.c	-	Логика программы, основной алгоритм
7	pio.c	Порт ввода/вывода	Функции управления портами ввода/вывода: инициализация на вход/выход, чтение/запись значения.
8	serial.c	асинхронный приемо-передатчик	Функции управления UART без прерываний: инициализация, отправка данных
9	timer.c	Таймер	Функции инициализации таймера, работа по прерыванию с периодом
10	watchdog.c	Сторожевой таймер	Функции сброса сторожевого таймера

5.1.1. Драйвер Ethernet-контроллера

Драйвер включает функции по управлению и настройке Ethernet контроллера cs8900a. Данный контроллер имеет возможность работы как по опросу так и по прерываниям. Поскольку траффик по Ethernet предполагает обмен достаточно большими объемами данных, то разумнее для разгрузки процессора использовать режим работы по прерываниям.

Среди аппаратных особенностей следует отметить, что сброс контроллера можно выполнять как программно, так и аппаратно.

Доступ ко всем внутренним регистрам контроллера Ethernet осуществляется через порт ввода/вывода процессора AM186ES. Где через порт с адресом 0xA осуществляется установка адреса внутреннего адреса регистра cs8900a, а через порт 0xC осуществляется чтение/запись значения в регистр. Таким образом, запись/чтение значения в регистр контроллера Ethernet осуществляется в два шага – выборка, запись/чтение.

Для обеспечения возможности полноценного обмена по Ethernet реализованы следующие команды:

1. Функция программного сброса *ethernet_reset_chip()*

Для выполнения программного сброса контроллера cs8900a необходимо в регистр по адресу 0x114 установить бит сброса в единицу, записав значение 0x40. Согласно документации после

выполнения данного действия необходимо выждать время, перед тем как все внутренние регистры контроллера будут проинициализированы значениями по умолчанию.

После окончания временного таймаута необходимо убедиться, что сброс контроллера выполнен успешно. Для этого в цикле опрашивается значение бита готовности статусного регистра(адрес 0x136). Операция чтение повторяется n-ное число раз. Как только бит успешного сброса взводится(0x80), выполнение команды считается выполненным успешно, иначе, если после многократного чтения регистра статуса бит не взвелся, считается, что сброс контроллера не произошел и дальнейшая работа с ним бессмыслена.

2. Функция инициализации *ethernet_init()*

Инициализация включает непосредственный сброс контроллера с помощью функции *ethernet_reset_chip(),* далее выполняется чтение двух байт из нулевого регистра контроллера. Два байта после сброса должны содержать константное значение, определяющее закодированное имя производителя. Сравнив полученные два байта с ожидаемым значением можно убедиться, что контроллер установлен на макете. Прочитав дополнительные два байта из регистра 0x2 можно определиьт ревизию контроллера в справочных целях.

После успешного определения производителя и ревизии, выполняется настройка управляющих регистров. Выполняется конфигурирование контроллера для работы по прерываниям, которые случаются при ошибке принятого пакета, при непосредственной передаче пакета, и при приеме корректного пакета (с правильным CRC, корректной длиной). С полным описанием всех регистров можно ознакомиться в документации на контроллер.

Также на данном этапе выполняется конфигурирование и инициализация MAC адреса контроллера. Данный адрес составляет 6 байт и может принимать любые значения по усмотрению пользователя.

Далее выполняется запись адреса обработчика прерываний в таблицу векторов, устанавливается приоритет, после чего процедура инициализации заканчивается.

3. Функция передачи данных *ethernet_send_packet()*

Передача пакета осуществляется с подачи соответствующей команды контроллера путем записи чрез порт по адресу 0x4 значения 0xC, что будет расцениваться контроллером как сигнал подготовки к отправке данных. После чего в регистр длины данных записывается общая длина пакета. Далее начинается опрос бита готовности контроллера к передаче данных. Как только бит готовности взводится, записывается 6 байт адреса назначения, 6 байт адреса отправителя (настроенный ранее MAC адрес SDK4), длина данных в пакете, и сами данные. Далее идет опрос

внутренней переменной, которая взводится в обработчике прерывания, который в свою очередь вызывается, как только отправляется пакет. По окончании опроса считается, что процедура отправки выполнена.

4. Функция передачи данных *ethernet_receive_packet()*

Данная операция выполняет чтение пакета из внутренней памяти, после чего взводится бит, определяющий, что чтение произошло и повторное чтение будет определяться ошибочным, поскольку данные будут считаться вычерпанными из буфера. Новые данные помещаются в буфер по прерыванию, и именно там выполняется установка внутренней переменной, что новые данные доступны. Данный код определяется как критическая секция, поскольку и обработчик прерывания и функция получения данных работают с общей переменой определения текущего состояния доступности новых данных, поэтому все операции с общей переменой обрамляются командами запрета и разращения прерываний.

5.1.2. Драйвер контроллера прерываний

Данный драйвер реализует простейшие функции управления и организации прерываний на процессоре AM186ES.

Для обеспечения возможности работы периферийных устройств по прерываниям реализованы следующие функции:

1. Функция регистрации обработчика прерывания *interrupt_vector_catch()*

Функция выполняет запись адреса обработчика прерывания в таблицу векторов прерываний. Вектора с адресами 0x9, 0x15 – 0x19 не доступны, поскольку зарезервированы производителем процессора. Таблица векторов начинается с нулевого физического адреса. Каждый вектор прерывания закреплен за определённым устройством, под адрес обработчика отводится 4 байта.

2. Функция маскирования требуемого прерывания *interrupt_enable()*

Функция маскирует требуемый номер прерывания, с помощью маски возможно как включение, так и отключение необходимого прерывания.

5.1.3. Драйвер портов ввода/вывода

Драйвер предназначен для конфигурирования определённой ножки порта. Данный драйвер реализует всего одну функцию:

1. Функция регистрации обработчика прерывания *pio_configure()*

Функция настраивает один из 32 доступных пинов на вход или выход. При настройке ножки как выходной, возможно включить «подтяжку» входного сигнала как к земле, так и к напряжению питания. В общем доступно 4 режима конфигурирования:

– Предустановленная функция «пина». За некоторыми ножками закрепляется предопределенная функция, например, как ножка приема по *uart*. Данный режим как раз закрепляет за «пином» функцию по умолчанию, определенную производителем процессора.

– Вход с подтяжкой к питанию. Данные в регистре «порта» будут определяться входным сигналом.

– Выход. Запись в регистр данных порта будет определять уровень на его выходе.

– Вход с подтяжкой к «земле».

5.1.4. Драйвер UART

Драйвер реализует функции управления UART. Данный интерфейс используется для отладочных целей путем трассирования этапов выполнения программы. Запись значений в регистры UART осуществляется по адресам пространства ввода/вывода, под которое отводится 64 КБ. Поскольку данный интерфейс используется только для вывода информации, то реализуется только часть функций:

1. Функция установки значения регистра управления *serial_set_control_reg()*

Записывает двухбайтное значение в регистр управления UART.

2. Функция установки скорости *serial_setbdrt()*

Функция записывает соответствующее значение в регистр скорости UART, значение определяется формулой в соответствии с которой его можно формально определить, как:

Значение регистра = частота процессора / 16 / скорость,

где значение скорости может быть задано в диапазоне от 9600 до 115200 бит/с. Но данная формула не всегда формирует правильное значение регистра, поэтому для фиксированного набора скоростей в документации приводятся соответствующие константные значения регистра.

3. Функция установки скорости *serial_init()*

Функция выполняет инициализацию UART в следующей последовательности:

– конфигурирование «пинов» UART’а в режиме предопределённой функции

– вызов функции настройки скорости

– вызов функции настройки регистра управления

Все конфигурационные значения хранятся в программной таблице.

4. Функция вывода символа *serial_putc()*

Функция отправляет по UART символ, переданный ей в качестве параметра. Перед отправкой выполняется многократное чтение регистра статуса UART в цикле, и проверка статуса готовности отправки. Как только бит готовности взводится, выполняется запись однобайтового символа в регистр данных.

5. Функция вывода строки *serial_puts()*

Функция отправляет по UART строку символов. В общем виде представляет собой ничто иное как, вызов функции отправки символа в цикле.

6. Функция вывода цифры *serial_print_num()*

Функция преобразует число в строку и отправляет её по UART. Возможно конвертирование и вывод цифры в различных системах счисления.

5.1.5. Драйвер таймера

Драйвер таймера используется для инициализации только одного из доступных в процессоре. Таймер выполняет роль часов макета, работая в режиме прерывания, которое «испускается» с фиксированным периодом.

Для обеспечения работы таймера реализованы следующие функции:

1. Функция инициализации таймера *timer_init()*

Функция выполняет конфигурирование таймера, настраивает прерывания, устанавливая обработчик прерывания, задается период срабатывания таймера, равный 10 мс.

2. Обработчик прерываний таймера *timer_handle()*

В обработчике выполняется инкрементирование внутренней переменной, значение которой определяет время работы макета.

3. Функция получения значения времени макета *timer_get_tick()*

Функция извлекает актуальное значение времени макета.

5.1.6. Драйвер сторожевого таймера

Сторожевой таймер аппаратно представляется как счетчик, по переполнению которого формируется сигнал сброса макета. Данный таймер необходим во избежание зацикливаний и зависаний программного обеспечения макета в нештатных ситуациях. Необходимо сбрасывать таймер в продолжительных по времени циклах. Логика драйвера сторожевого таймера самая простая из всех устройств и представлена одной функцией:

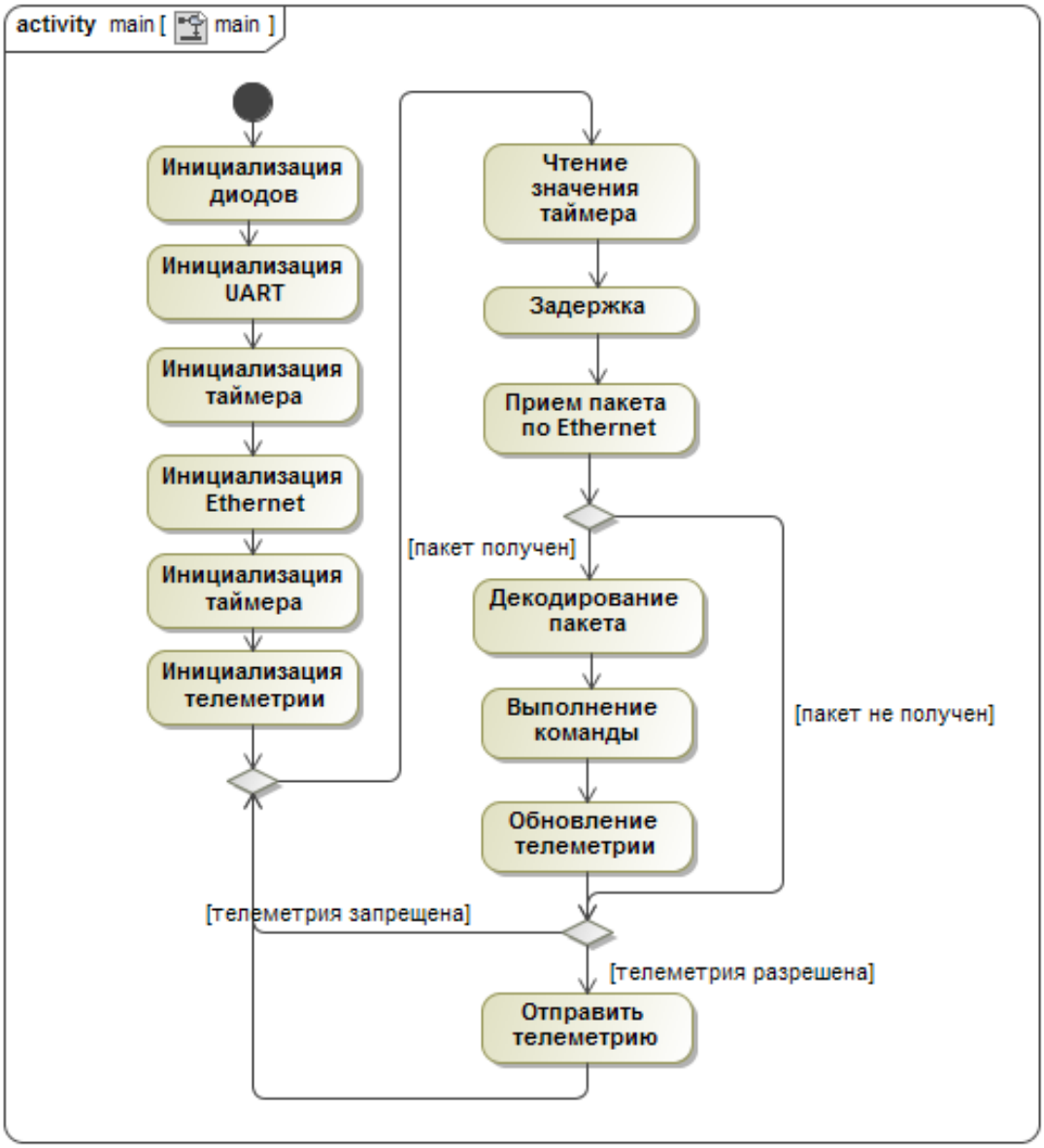
1. Функция сброса сторожевого таймера *WDT_reset()*

Сброс осуществляется последовательной записью константных значений в регистр сторожевого таймера.

5.1.7. Логика программы

При запуске приложения в первую очередь выполняется инициализация периферии. Диаграмма деятельности, поясняющая логику работы основного приложения приводится на рисунке.

Диаграмма деятельности системного программного обеспечения макета SDK4



Рисунок

Инициализация выполняется в следующем порядке:

1. Инициализация диодов: для красного и желтого диодов выполняется конфигурирование на выход соответствующих «пинов» порта ввода/вывода.

2. Далее выполняется инициализация UART: настройка регистра управления, установка скорости.

3. Выполняется инициализация таймера: конфигурируется один таймер, настраивается срабатывание прерываний с периодом в 10 мс.

4. Выполняется сброс сторожевого таймера во избежание сброса макета.

5. После выполняется инициализация пакетов телеметрии в части адресов получателя и назначения.

6. Выполняется конфигурирование перемычек, как «пинов» порта ввода вывода на вход

7. Начинается тело бесконечного цикла

8. Получение значения таймера платы

9. Выполнение программной задержки во избежание слишком частой отправки пакетов телеметрии

10. Вызов функции приема пакета по Ethernet, анализ возвращенного значения, если новый пакет не принят то декодирования пакета не выполняется. Если новый пакет принят , то выполняется декодирование полей командного пакета, выполняется соответствующая команда, устанавливаются поля телеметрии.

11. Если ранее командный пакет разрешил отправку телеметрии, то выполняется отправка пакета по Ethernet, далее программа вновь возвращается в вершину бесконечного цикла в п. 7.

6. РАЗРАБОТКА ПРИКЛАДНОГО ПО ДЛЯ WINDOWS

6.1. Описание программы

Программа должна работать на ПК под управлением Windows NT, формировать командные запросы, принимать и декодировать телеметрию, отправляемую с макета SDK4.0. Формат телеметрии и командных пакетов описывается выше.

6.2. Описание среды разработки

Программное обеспечение под условным названием «Менеджер SDK4.0» разрабатывается на основе Qt Фреймворка.

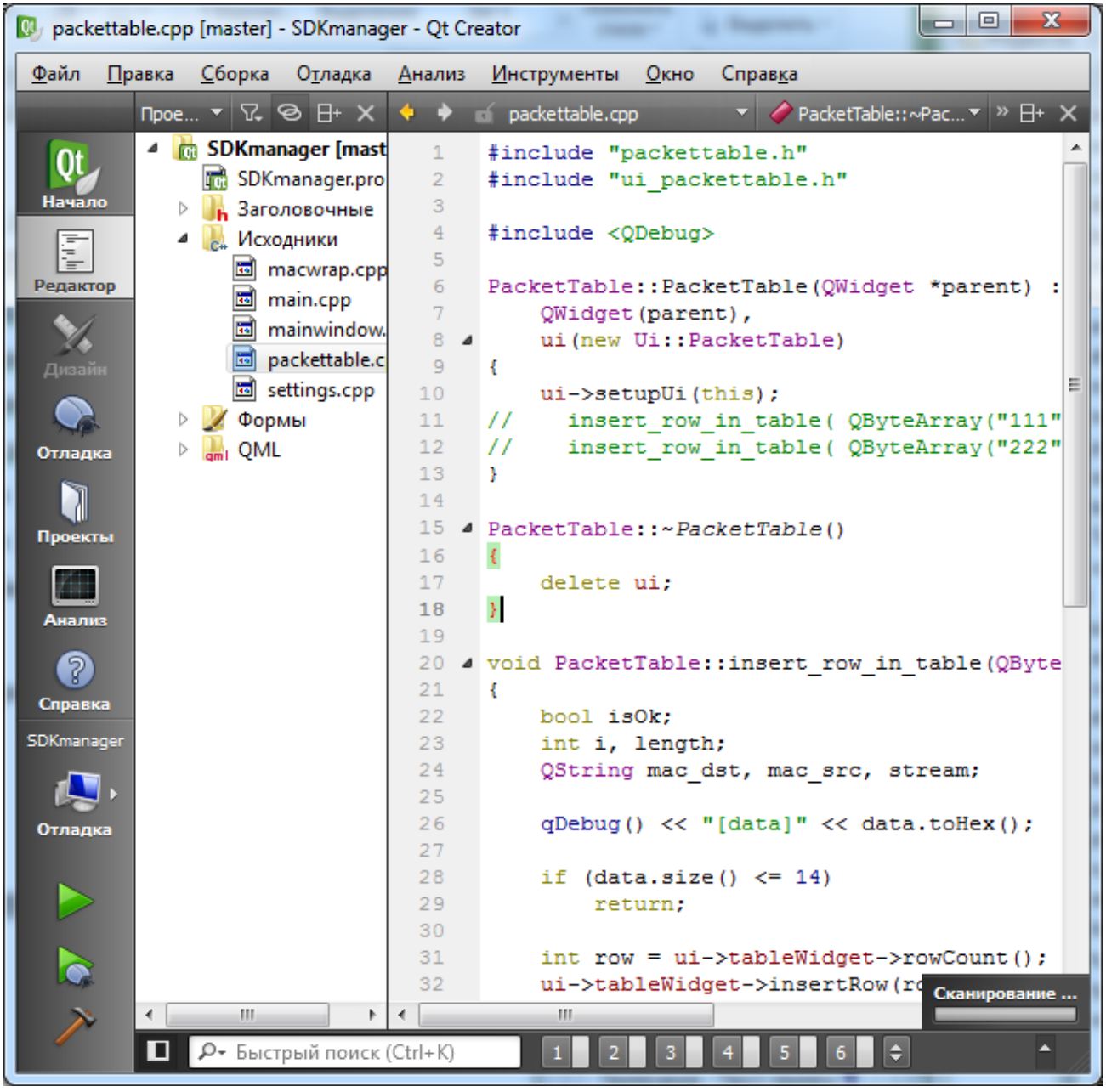
Отличительная особенность Qt от других библиотек — использование Meta Object Compiler (MOC) — предварительной системы обработки исходного кода (в общем-то, Qt — это библиотека не для чистого C++, а для его особого наречия, с которого и «переводит» MOC для последующей компиляции любым стандартным C++ компилятором). MOC позволяет во много раз увеличить мощь библиотек, вводя такие понятия, как слоты и сигналы. Кроме того, это позволяет сделать код более лаконичным. Утилита MOC ищет в заголовочных файлах на C++ описания классов, содержащие макрос Q_OBJECT, и создаёт дополнительный исходный файл на C++, содержащий метаобъектный код.

Qt позволяет создавать собственные плагины и размещать их непосредственно в панели визуального редактора. Также существует возможность расширения привычной функциональности виджетов, связанной с размещением их на экране, отображением, перерисовкой при изменении размеров окна.

Qt комплектуется визуальной средой разработки графического интерфейса «Qt Designer», позволяющей создавать диалоги и формы в режиме WYSIWYG. В поставке Qt есть «Qt Linguist» — графическая утилита, позволяющая упростить локализацию и перевод программы на многие языки; и «Qt Assistant» — справочная система Qt, упрощающая работу с документацией по библиотеке, а также позволяющая создавать кросс-платформенную справку для разрабатываемого на основе Qt ПО. Начиная с версии 4.5.0 в комплект Qt включена среда разработки «Qt Creator», которая включает в себя редактор кода, справку, графические средства «Qt Designer» и возможность отладки приложений. «Qt Creator» может использовать GCC или Microsoft VC++ в качестве компилятора и GDB в качестве отладчика. Для Windows версий библиотека комплектуется компилятором, заголовочными и объектными файлами MinGW.

Основное окно среды разработки представлено на рисунке.

Окно Qt Creator



Рисунок

6.3. Описание технологий

Графический интерфейс пользователя реализован при помощи технологии QML, активно продвигаемой в последнее время.

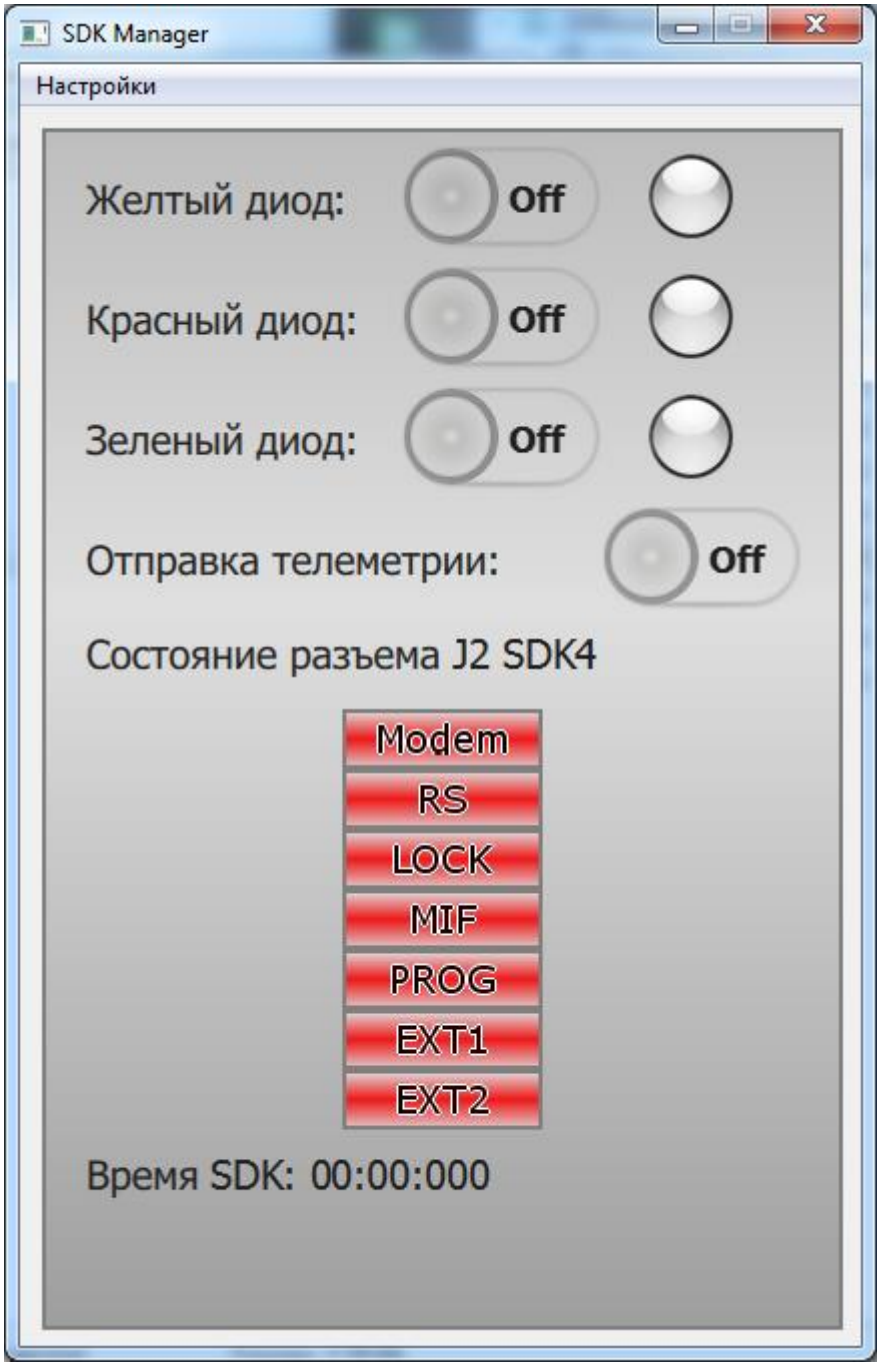
QML (Qt Meta-Object Language) — декларативный язык программирования, основанный на JavaScript, предназначенный для дизайна приложений, делающих основной упор на пользовательский интерфейс. Является частью Qt Quick, среды разработки пользовательского интерфейса, распространяемой вместе с Qt. В основном используется для создания приложений, ориентированных на мобильные устройства с сенсорным управлением.

QML-документ представляет собой дерево элементов. QML элемент, так же, как и элемент Qt, представляет собой совокупность блоков: графических (таких, как rectangle, image) и поведенческих (таких, как state, transition, animation). Эти элементы могут быть объединены, чтобы построить комплексные компоненты, начиная от простых кнопок и ползунков и заканчивая полноценными приложениями, работающими с Internet.

QML элементы могут быть дополнены стандартными JavaScript вставками путем встраивания .js файлов. Также они могут быть расширены C++ компонентами через Qt framework.

6.4. Сторонние библиотеки

6.5. Логика приложения


SDK Manager
Настройки
Желтый диод:
Off
Красный диод:
Off
Зеленый диод:
Off
Отправка телеметрии:
Off
Состояние разъема J2 SDK4
Modem
RS
LOCK
MIF
PROG
EXT1
EXT2
Время SDK: 00:00:000