

1 Лабораторная работа №1. Обзор отладочного комплекта, среды разработки, документации, цифрового осциллографа

Установка, настройка и порядок работы с интегрированной средой разработки MDK Keil μ Vision, изучение средств отладки. Изучение цифрового осциллографа и отладочного комплекта Open32F3-D.

1.1 Часть 1. Обзор документации, изучение цифрового осциллографа

Меры безопасности при работе в лаборатории. Порядок установки и настройки интегрированной среды разработки Keil-MDK-ARM (ИСР Keil). Изучение цифрового осциллографа.

Учебное время 2 часа.

Цель работы: Привитие практических навыков по установке ИСР Keil и использованию осциллографа.

1.1.1 Содержание работы

- 1) Изучить инструкцию по технике безопасности.
- 2) Изучить порядок установки и настройки ИСР Keil для работы с микроконтроллерами (МК) STM32.
- 3) Скачать основную документацию МК STM32F303VCT6 и перевести названия основных разделов RM0316 [1] и DS9118 [2].
- 4) Изучить основные функции и органы управления осциллографа ПрофКиП С8-2021 [3, 4].
- 5) В ручном режиме при настройках, согласно номеру варианта таблицы 1.1 измерить амплитуду и частоту сигнала компенсации пробника. Сохранить сигнал (п. 2.12 [3]) на внешний носитель памяти (USB Flash).
- 6) Изменяя развёртку по времени (по напряжению оставить согласно варианта), измерить время нарастания (п. 2.11.3 [3]) переднего (Rise Time) и время спада заднего (Fall Time) фронта сигнала компенсации пробника. Сохранить процессы нарастания и спада (п. 2.12 [3]) на внешний носитель памяти (USB Flash).

Таблица 1.1 – Варианты заданий лабораторной работы №1 часть 1.

Номер варианта	Чувствительность по напряжению	Скорость развёртки по времени	Номер варианта	Чувствительность по напряжению	Скорость развёртки по времени
1.	100 мВ/дел	100 мкс/дел	14.	200 мВ/дел	500 мкс/дел
2.	200 мВ/дел	50 мкс/дел	15.	500 мВ/дел	250 мкс/дел
3.	500 мВ/дел	500 мкс/дел	16.	2 В/дел	100 мкс/дел
4.	2 В/дел	250 мкс/дел	17.	100 мВ/дел	500 мкс/дел
5.	100 мВ/дел	50 мкс/дел	18.	200 мВ/дел	250 мкс/дел
6.	200 мВ/дел	1 мс/дел	19.	500 мВ/дел	100 мкс/дел
7.	500 мВ/дел	1 мс/дел	20.	2 В/дел	50 мкс/дел
8.	2 В/дел	500 мкс/дел	21.	100 мВ/дел	250 мкс/дел
9.	1 В/дел	1 мс/дел	22.	200 мВ/дел	100 мкс/дел
10.	1 В/дел	500 мкс/дел	23.	500 мВ/дел	50 мкс/дел
11.	1 В/дел	100 мкс/дел	24.	2 В/дел	1 мс/дел
12.	1 В/дел	50 мкс/дел	25.	100 мВ/дел	100 мкс/дел
13.	100 мВ/дел	1 мс/дел	26.	1 В/дел	50 мкс/дел

1.1.2 Инструкция по технике безопасности

Требования безопасности во время занятий:

- 1) Проверить исправность стола и стула (кресла).
- 2) Прежде чем включить электроприбор (осциллограф, компьютер, зарядку и т.п.) в электросеть проверить исправность шнуровой пары: изоляционные втулки штепселей не должны иметь трещин, а шнуры - оголенных от изоляции мест. Розетка должна быть плотно укреплена в стене.
- 3) При обнаружении каких-либо неисправностей в работе оборудования или возникновении каких-либо сомнений по его исправности, доложить преподавателю и до их устранения к учебе не приступать.
- 4) При мигании света, перегорании ламп, неисправности электророзеток, выключателей или их крышек, нарушении изоляции электропроводки сообщить преподавателю или секретарю кафедры.
- 5) При включении и выключении электроприборов братья только за корпус вилки или разъема.
- 6) Запутанный питающий провод любого электроприбора распутывать только при вынутой вилке из штепсельной розетки.

В целях соблюдения электробезопасности студентам запрещается:

- брать в руки оборванные, висящие или лежащие на полу электропровода и наступать на них – они могут находиться под напряжением;

- подходить к электрощитам, открывать двери электрощитов и электрошкафов;
- прикасаться к токоведущим частям электроприборов, клеймам, неизолированным или поврежденным электропроводкам, к арматуре освещения;
- включать в сеть переносные электроприборы без штепсельных розеток;
- пытаться устранить самостоятельно неполадки электрооборудования (освещение, розетки и т. п.).

В целях соблюдения пожаробезопасности студентам запрещается:

- в учебных и служебных помещениях курение и применение открытого огня;
- вешать одежду и другие предметы на выключатели, штепселя или рубильники.
- допускать скопление мусора, одежды, посторонних предметов.

Студент должен знать:

- места расположения медицинской аптечки и средств пожаротушения;
- номера телефонов медицинской службы и пожарной охраны;
- пути эвакуации, а также главные и запасные выходы в случае аварии и пожара.

Студент должен соблюдать требования стандартов, норм и правил по охране труда, а также приказов, распоряжений, постановлений, инструкций по охране труда и пожарной безопасности, регламентированные для охраны учебного процесса.

Меры безопасности при работе с безкорпусной отладочной платой:

- 1) Приступая к работе с открытой печатной платой, где микросхемы и проводники не защищены корпусом, нужно понимать опасность выхода из строя микросхем при касании от статического электричества, которое почти всегда накапливается на теле человека.
- 2) Будьте внимательны и аккуратны! При небрежном отношении к оборудованию возможны механические повреждения.

1.1.3 Отладочная плата

Для начала программирования встроенных приложений на МК недостаточно одного лишь ПК. Необходима отладочная плата и программатор. В нашем курсе используется отладочный комплект Open32F3-D Standard [5] и встроенный в отладочную плату (STM32F3-Discovery) USB программатор ST_Link/V2 (ST-Link Debugger см. рис. 1.1). Для начала самостоятельного изучения, на начальном этапе, будет вполне достаточно отладочной платы STM32F3-Discovery [6].

На плате STM32F3-Discovery (см. рис. 1.1) установлен:

- микроконтроллер STM32F303VCT6;

- 3х-осевой цифровой микроэлектромеханический гироскоп (L3GD20);
- микроэлектромеханическая система в корпусе – акселерометр с компасом, содержащая 3-осевой цифровой линейный акселерометр и 3-осевой цифровой геомагнитный сенсор (LSM303DLHC);
- 8 управляемых светодиодов, две пользовательские кнопки;
- встроенный USB отладчик/программатор ST-LINK/V2.

Питание от интерфейса USB или от внешнего источника 3 или 5 В;



Рисунок 1.1 – Отладочная плата STM32F3-Discovery.

1.1.4 Среда разработки

Для микроконтроллеров на ядре ARM существует достаточное количество сред разработки (см. рис. 1.2). При выполнении цикла аудиторных лабораторных работ на МК фирмы STMicroelectronics будем использовать интегрированную среду разработки (Integrated Development Environment - IDE) немецкой компании Keil для платформы ARM, Keil MDK ARM (Microcontroller Development Kit) [7]. Это оценочная версия ПО, которая предоставляется разработчиком бесплатно, но имеет ограничение на размер компонуемого кода в 32 килобайта. Компания Keil является официальным партнером ARM, а сама Keil-MDK

является совместной разработкой Keil и ARM. Так ядро IDE (компилятор, линковщик, ассемблер и ряд утилит) собственная разработка ARM, а от Keil – оболочка (µVision IDE) и отладчик.

После окончания установки Keil-MDK-ARM необходимо поставить пакет программного обеспечения для поддержки МК STM32F303VCT6 в этой среде [8]. Пакет представляет собой файл с расширением *.pack (например Keil.STM32F3xx_DFP.2.2.2.pack).

Provider	Product	Cores	Framework		Purpose		Compiler	Debugger	Win	Linux	OS X
			Proprietary	Eclipse	General	Specific					
ac6	System Workbench (SW4STM32)	All		✓	✓		gcc	OpenOCD	✓	✓	✓
Atollic	Atollic TrueSTUDIO	All		✓	✓		gcc	gdb	✓	✓	
iSystem	iSYS-WinIDEAOpen	All	✓		✓		gcc	gdb	✓		
Arm KEIL	MDK5-Cortex-M	M0, M0+	✓		✓		ARM/llvm	uVision	✓		
Arduino	Arduino IDE	All	✓			✓	gcc		✓	✓	✓
Arm MBED	ARM mbed	All	✓			✓	ARM		✓	✓	✓

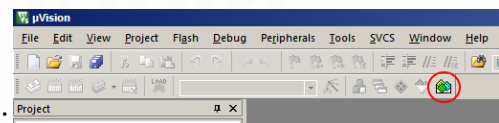
Provider	Product	Cores	Framework		Compiler	Debugger	Win	Linux	OS X	Safety edition	Free edition or use, limitation
			Proprietary	Eclipse							
IAR Systems	EWARM	All	✓		IAR	IAR	✓			✓	Code limit: 32KB
Arm KEIL	MDK5-Cortex-M	All	✓		ARM / llvm	uVision	✓			✓	Code limit: 32KB
Emprog	ThunderBench	All		✓	gcc	OpenOCD	✓				Time limit: 30-day
iSystem	iSYS-WinIDEA	All	✓		gcc, others	gdb	✓				WinIDEAOpen
Raisonance	Raisonance Ride7	All	✓		gcc	gdb	✓				No
Rowley	CrossWorks	All	✓		gcc	Rowley	✓	✓	✓		Time limit: 30-day
Segger	Embedded Studio	All	✓		gcc / llvm	Segger	✓	✓	✓		Build/Run warning
SysProgs	VisualGDB Embedded	All	✓		gcc / llvm	OpenOCD	✓	✓			Time limit: 30-day
Tasking	TaskingVX	All		✓	Tasking	Tasking	✓	✓	✓		On request
Cosmic	IDEA	All	✓		Cosmic	Cosmic	✓				Code limit: 32KB
Green hills	MultiIDE	All	✓		Green hills	Green hills	✓	✓			

Рисунок 1.2 – Интегрированные среды разработки под ядро ARM.

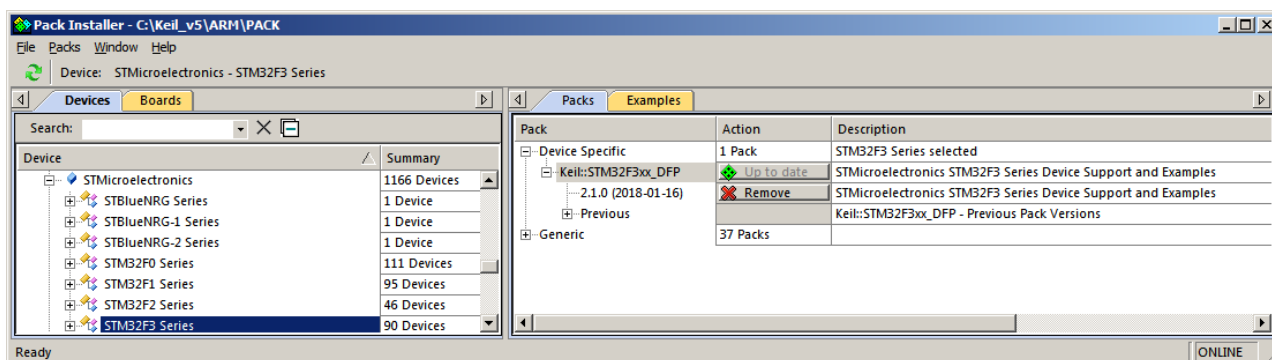
Настройка ИСР Keil под МК STM32F303VCT6.

Для установки скачанного пакета запускаем ИСР Keil-MDK () и нажимаем значок

менеджера пакетов  (...\\Keil_v5\\UV4\\PackInstaller.exe):



В появившемся окне менеджера пакетов (*Pack installer*) импортируем необходимый для работы файл (*File >> import >> Keil.STM32F3xx_DFP.x.x.x.pack*). После завершения установки проверяем, что файлы были подключены. Для этого, в этом же окне в левой панели во вкладке *Devices* выбираем *STMicroelectronics >> STM32F3 series*. В правой панели во вкладке *Packs* в раскрывающемся списке *Device Specific* должен быть пакет *Keil::STM32Fxx_DFP* отмеченный как *up to date*:



Далее для работы, нам понадобится утилита STM32CubeProg [9] – программный интерфейс для программирования микроконтроллеров STM32 от производителя, на случай если нам, понадобится: считать прошивку с МК; обновить прошивку программатора; осуществлять вывод в окно отладки; автоматически установить драйвера.

1.1.5 Документация и библиотеки

Освоение курса подразумевает умение работать с программными библиотеками под STM32. Основные библиотеки:

1) Библиотека CMSIS [10] (Cortex Microcontroller Software Interface Standard) - стандарт программного интерфейса микроконтроллеров с ядром Cortex, разработанный компанией ARM. Входит в состав ИСР Keil-MDK.

2) Библиотека SPL [11, 12] (Standard Peripherals Firmware Library) – стандартная библиотека периферии, набор низкоуровневых драйверов. Каждый драйвер предоставляет пользователю набор функций для работы с периферийным блоком.

3) Библиотека HAL LL [13, 14] (Hardware Abstraction Layer and Low-Layer drivers) – слой абстракции встроенного программного обеспечения, гарантирующий максимальную переносимость кода между семействами STM32.

При работе с микроконтроллером STM32, приходится часто заглядывать в документацию, чтобы найти какие-либо сведения. Основной перечень документации на микроконтроллер STM32F303VC доступен на сайте компании ST Microelectronics по ссылке: <https://www.st.com/en/microcontrollers-microprocessors/stm32f303vc.html#documentation>.

Перечень разделён на разделы. Выделим необходимые документы из раздела Documentation (техническая документация):

- Product Specifications (спецификация изделия). Здесь один файл спецификации DS9118 [2] в котором подробно описаны аппаратные и программные особенности микроконтроллера STM32F303VC. Дополняет справочное руководство RM0316 (ниже).

- Reference Manuals (справочники). Здесь одно справочное руководство *RM0316* [1] по семейству STM32F303xB/C/D/E (более 1000 страниц) с подробнейшей информацией по этой серии. Здесь описана архитектура, регистры, их биты, функции и периферия микроконтроллера.

- Programming Manuals (руководства программиста). Здесь одно руководство *PM0214* [15] для программирования систем с ядром Cortex-M4, в котором описывается программирование на уровне машинных команд и ассемблера, регистры общего и специального назначения.

Основной перечень инструментальных средств и программного обеспечения, доступных для МК STM32F303VC, расположен здесь: <https://www.st.com/en/microcontrollers-microprocessors/stm32f303vc.html#tools-software>. В этом разделе можно найти и скачать все необходимые программные пакеты и библиотеки для комфортной работы с микроконтроллером STM32F303VC.

Для работы с отладочной платой STM32F3-Discovery, необходимо описание UM1570 [16] и схема [17].

Итак, основной перечень необходимых документов для дальнейшей работы в том числе и самостоятельной состоит всего из четырёх документов: *RM0316* [1], *DS9118* [2], *PM0214* [15], MB1035-F303C-D01 [17].

1.1.6 Осциллографирование электрических сигналов

Осциллограф – измерительный прибор, предназначенный для визуального наблюдения и исследования электрических сигналов.

Исследуемый сигнал отображается на экране в виде светящихся линий или фигур, называемых осциллограммами. Осциллограмма представляет собой функциональную зависимость двух или трех величин $y=F(x)$ или $y=F(x, z)$, каждая из которых является, в свою очередь, функцией времени: $y(t)$, $x(t)$, $z(t)$. Вертикальная ось (Y) представляет значения напряжения, горизонтальная ось (X) – время. Интенсивность или яркость выведенной на экран прибора картинка называется осью Z (см. рис. 1.3).

Осциллограф измеряет форму сигнала [4]. Форма колебания напряжения выражается в виде диаграммы зависимости величины напряжения (по вертикали) от времени (по горизонтали) которые показаны на рисунке 1.3. Современные цифровые осциллографы обладают функциями, значительно облегчающими измерения формы сигналов. В качестве органов управления эти приборы используют клавиши лицевой панели и/или экранные меню,

через которые можно выбрать режимы полностью автоматических измерений, включающих в себя измерение: амплитуды, периода, частоты, времени нарастания/спада импульса, длительности импульса и др. Результаты автоматических измерений отображаются на экране в текстовом формате. Обычно такие показания более точны, чем интерпретация графического изображения.

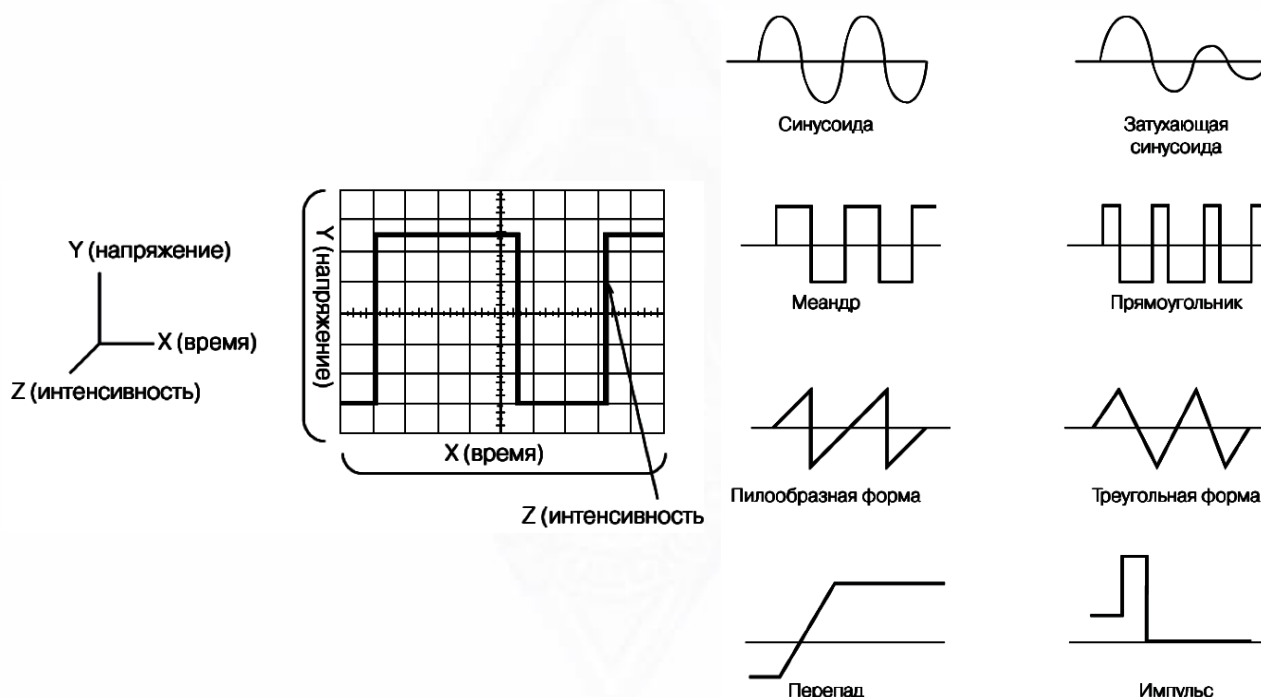


Рисунок 1.3 – Компоненты X, Y и Z отображаемого сигнала, основные формы сигналов.

Обычный цифровой осциллограф (ЦО) позиционируется как цифровой запоминающий осциллограф (DSO - digital storage oscilloscope). Дисплей такого прибора относится к экрану растрового типа. ЦО позволяют захватывать и просматривать события, случающиеся однократно, например переходные процессы. Поскольку информация о сигнале существует в цифровом формате в виде последовательности сохранённых бинарных значений, эти значения можно легко анализировать, архивировать, распечатывать, либо обрабатывать каким-либо иным способом, как в самом осциллографе, так и во вне. В отличие от аналоговых моделей, цифровые запоминающие осциллографы обеспечивают постоянное сохранение в памяти захваченной информации, разностороннюю обработку параметров и их анализ (см. рис. 1.4). Однако такие приборы не отображают градации яркости развёртки сигнала в реальном времени, поэтому DSO неспособны наглядно представлять изменяющиеся “живые” сигналы.



Рисунок 1.4 – Маршрут последовательной обработки входных сигналов ЦО.

Как и в аналоговых осциллографах, первым (входным) функциональным узлом ЦО является усилитель вертикального отклонения. Органы управления вертикальным отклонением позволяют регулировать амплитуду и положение развёртки сигнала. Далее, аналого-цифровой преобразователь (АЦП) в системе горизонтального отклонения осуществляет выборку сигнала в дискретных точках определенного временного интервала и преобразует напряжение исследуемого сигнала в этих точках в цифровые значения, называемые элементами выборки. Весь этот процесс называется оцифровкой сигнала.

Схема синхронизации системы горизонтального отклонения устанавливает частоту, с которой АЦП делает выборки. Эта величина называется частота выборки и измеряется в выборках в секунду (выб/с). Выборки, полученные от АЦП, сохраняются в оперативной памяти прибора в качестве элементов описания формы сигналов. Некоторое количество выборок могут составить одну точку развёртки сигнала. Взятые вместе точки развёртки сигнала составляют одну развёртку сигнала. Используемое количество точек, необходимое для создания развёртки сигнала называется длина записи. Система запуска осциллографа определяет момент пуска и останова процесса записи.

Сигнальный тракт цифровых осциллографов включает в себя микропроцессор, через который проходит измеряемый сигнал. Микропроцессор обрабатывает сигнал, управляет выводом данных на дисплей, органами управления передней панели прибора и т.п. Затем сигнал поступает в память дисплея, а из нее – выводится на экран.

Обычный осциллограф состоит из 4-х различных систем: системы вертикального отклонения (VERTICAL); горизонтального отклонения (HORIZONTAL); системы запуска (TRIGGER); системы отображения информации (дисплей) (см. рис. 1.5). Правильное понимание того, как работает каждая из этих систем, позволяет эффективно использовать прибор для решения различных измерительных задач. Необходимо помнить, что каждая система вносит свой вклад в способность осциллографа точно воспроизводить форму сигнала.

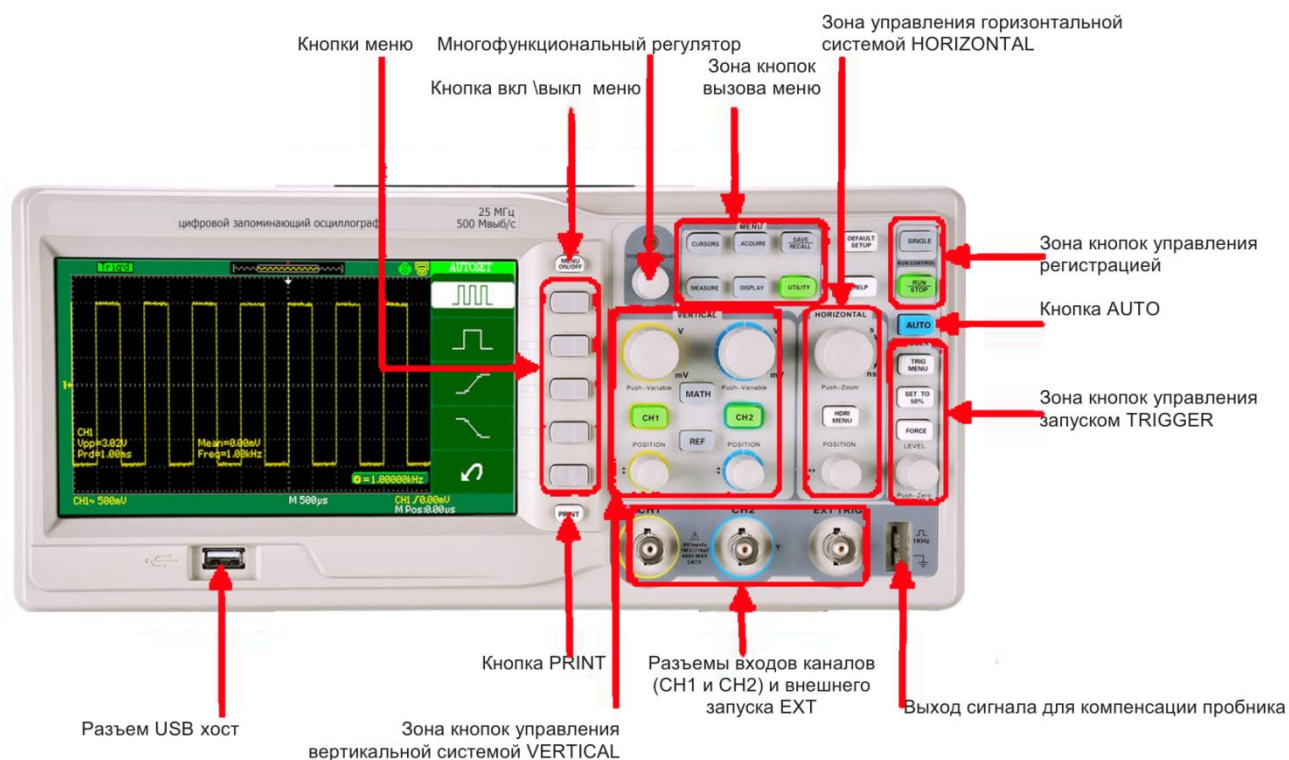


Рисунок 1.5 – Органы управления осциллографом ПрофКиП С8-2021, расположенные на передней панели.

Для того чтобы приступить к измерениям входного сигнала, на осциллографе необходимо выполнить три основные настройки:

- 1) Вертикальное отклонение: установить уровень ослабления или усиления сигнала. Установите ручкой управления чувствительностью (см. рис. 1.6) необходимое число вольт на деление (В/дел) для регулировки вертикального размера отображаемой развёртки сигнала. Если установлено значение 5 В/дел, это значит, что на каждое деление вертикальной шкалы приходится по 5 вольт. Если шкала экрана осциллографа имеет восемь вертикальных делений, то при такой настройке на экране уместится развёртка сигнала амплитудой 40 В. При настройке 0,5 В/дел, весь экран займет развёртка сигнала амплитудой 4 В и т.д. Максимальное напряжение, которое можно отобразить на экране, равно значению настройки В/дел, умноженному на количество делений вертикальной шкалы. Необходимо учитывать, что используемые пробники с коэффициентами 1х и 10х также влияют на коэффициент масштабирования. В этом случае необходимо умножать значение В/дел на коэффициент ослабления пробника.

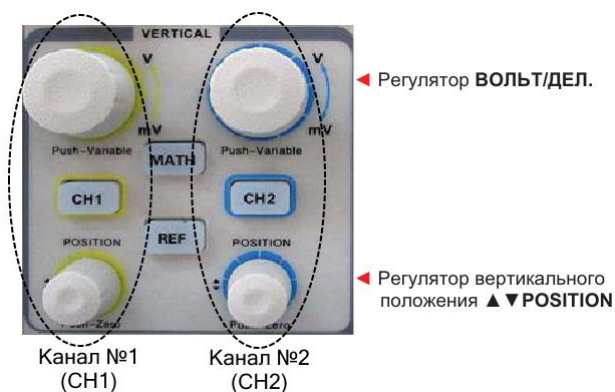


Рисунок 1.6 – Органы управления вертикальной системы развёртки.

2) Горизонтальное отклонение: установить скорость развёртки. Установите необходимое число секунд на деление (сек/дел) для регулировки масштаба развёртки сигнала по горизонтальной оси (см. рис. 1.7). С помощью регуляторов можно изменять горизонтальный масштаб и положение осциллограмм. Стрелка в верхней части масштабной сетки указывает положение момента запуска, а соответствующее числовое значение положения момента запуска в единицах времени отображается в нижней части экрана.

3) Запуск: установить параметры запуска осциллографа. Установите уровень запуска для стабилизации развёртки периодического сигнала или выберите режим однократного запуска (см. рис. 1.8). Система запуска определяет момент запуска – момент начала отсчёта времени для регистрируемых данных и отображаемой осциллограммы сигнала. Правильная настройка системы запуска позволяет вместо нестабильного изображения или просто пустого экрана получить осциллограмму, верно воспроизводящую форму сигнала.



Рисунок 1.7 – Органы управления горизонтальной системы развёртки.



Рисунок 1.8 – Органы управления системы запуска.

Функция запуска осциллографа синхронизирует начало горизонтальной развёртки с соответствующей точкой сигнала, что является чрезвычайно важным для точного определения характеристик этого сигнала. Органы управления запуском позволяют стабилизировать изображение периодических сигналов и захватывать однократные и непериодические

импульсы. Запуск превращает периодический сигнал в статическое изображение на экране осциллографа посредством непрерывного отображения одного и того же участка входного сигнала. Если каждая развёртка начинается с различных участков исследуемого импульса, то изображение будет нестабильным, с наложениями сигнала, и будет "плыть" влево или вправо, как это проиллюстрировано на рисунке 1.9.

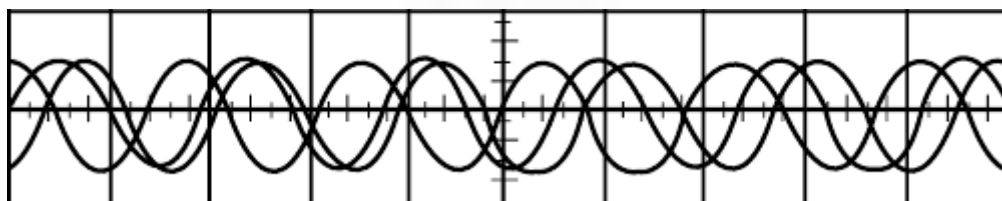


Рисунок 1.9 – Несинхронизированная развертка синусоидального сигнала.

Помимо запуска по уровню (регулятор Level рис. 1.8), применяемому в аналоговых и цифровых осциллографах, большинство цифровых моделей имеют массу специализированных режимов запуска, отсутствующих в аналоговых моделях. В этих режимах запуск осуществляется по определенным условиям или свойствам входного сигнала, что значительно облегчает их выявление, например, обнаружение импульсов, имеющих длительность меньше установленной.

Управление горизонтальным положением точки запуска присутствует только в цифровых осциллографах. Эта функция указывает положение точки запуска по горизонтали при регистрации сигнала как показано на рисунке 1.10. Изменяя это положение, оператор получает возможность увидеть форму сигнала перед моментом запуска. Эта функция известна как «упреждающий запуск». Таким образом, можно определить длину окна обзора исследуемого сигнала как после точки запуска, так и до нее. Способность цифровых осциллографов отображать развертку сигнала до запуска обусловлена тем, что они постоянно обрабатывают входной сигнал, вне зависимости от того, произошла ли команда запуска или нет. Непрерывный поток данных постоянно поступает на прибор, а запуск просто указывает осциллографу на необходимость сохранения части этих данных в памяти.



Рисунок 1.10 – Установка события запуска.

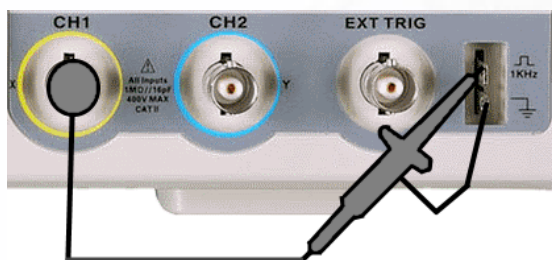
Режимы запуска определяют, каким образом осциллограф отобразит развертку сигнала на основе характеристик этого сигнала. Основными режимами запуска являются «ждущий» (Normal) и «автоматический» (Auto). При «ждущем» режиме осциллограф запускает развертку, только когда параметры входного сигнала отвечают заданным условиям запуска. В противном случае экран остаётся тёмным (аналоговые осциллографы) или застывшим на последней захваченной осциллограмме (цифровые осциллографы). В «Автоматическом» режиме осциллограф запускает развёртку без сигнала запуска. Если в какой-либо момент входного сигнала нет, таймер осциллографа всё равно запускает развёртку. При этом на экране прибора всегда присутствует изображение, даже если сигнал не удовлетворяет условиям запуска. Практический интерес представляют оба режима: «ждущий» режим позволяет просматривать только нужные сигналы, даже на низкой скорости развертки, а «автоматический» режим почти не требует настроек.



Более подробно по настройкам режимов и функциях нужно прочитать в руководстве по эксплуатации осциллографа UDS1000 [3] (аналог ПрофКиП С8-2021).

При работе с интегральными схемами (ИС) необходимо принять меры по защите исследуемой схемы от статического электричества. Интегральные схемы имеют в своем составе полевые транзисторы, которые могут быть повреждены статическим электричеством, накапливаемым на человеческом теле. Например, вы можете полностью вывести из строя дорогостоящую ИС, прикоснувшись к ее выводам после того, как сняли свитер. Для решения этой проблемы необходимо носить антистатический браслет, который безопасно снимает статический заряд с человеческого тела. Или стараться не касаться к выводам во время лабораторных работ.

Для начала измерений включите питание осциллографа. Подключите пробник к первому каналу (CH1) осциллографа. Для этого, совместите пазы разъёма пробника с

выступами входного разъёма BNC канала CH1 и зафиксируйте его поворотом по часовой стрелке. Переключатель ослабления на пробнике установите в положение 1х. При измерениях сигналов требуется наличие двух соединений: соединение наконечника пробника с исследуемой цепью и соединение контакта «земли» пробника с «землей» исследуемого устройства, обычно обозначаемые как GND (ground) или \perp . Для соединения с «землей» пробники, как правило, оснащаются зажимом типа «крокодил». На практике, необходимо прикрепить зажим «крокодил» к известной точке заземления исследуемого устройства, а затем прикасаться наконечником пробника к контрольным точкам тестируемой схемы. Для проверки работоспособности и предварительной настройки осциллографа подключите наконечник пробника и его зажим заземления к выходу осциллографа для компенсации пробников:



Осциллограф ПрофКиП С8-2021 имеет кнопку AUTO ( автоматическая настройка) и кнопку DEFAULT SETUP ( установка по умолчанию), позволяющие настраивать органы управления в один приём (подраздел 1.2 руководства [3]). Нажмите кнопку DEFAULT SETUP, для восстановления настроек изготовителя, далее нажмите кнопку AUTO, и через несколько секунд на экране появится осциллограмма меандра с частотой 1 кГц и амплитудой 3 В (см. рис. 1.11). Если меандр не появился или его параметры не соответствуют указанным, попробуйте поменять пробник на другой, возможно неисправность в нём.

Наиболее общий метод измерения напряжения – это подсчёт числа делений вертикальной шкалы, перекрытых разверткой сигнала (рис. 1.11). Выберите такую чувствительность вертикального отклонения (В/дел), чтобы развертка сигнала занимала большую часть площади экрана по вертикали. Чем большую площадь экрана вы займете, тем точнее будут результаты.



Рисунок 1.11 – Меандр амплитудой 3В и частотой 1кГц.

Для измерения времени используем горизонтальную шкалу осциллографа. Измерения времени включают в себя измерения периода и длительности импульсов. Частота – величина обратная периоду, таким образом, зная период, можно рассчитать значение частоты (рис. 1.11). Точно также как и при измерениях напряжения, измерения времени более точны при позиционировании развертки измеряемого сигнала так, чтобы один период покрывал большую площадь экрана.

Осциллограф обладает экранными маркерами, позволяющими измерять параметры сигналов автоматически, без подсчета делений шкалы вручную. Маркер – обычная линия, которую можно перемещать по экрану осциллографа. Две горизонтальные маркерные линии можно перемещать вверх и вниз для того, чтобы измерить амплитуду сигнала между ними, две вертикальные линии можно перемещать вправо и влево, чтобы измерить время. Управление экранными маркерами можно изучить самостоятельно по руководству [3] подраздел 2.11.2.

В некоторых случаях очень важно знать точные параметры формы импульса. В стандартные измерения параметров импульсов входят измерения длительности импульса и времени нарастания фронта импульса. Время нарастания – это время, необходимое перепаду импульса для перехода от низкого уровня до высокого уровня. Для удобства время нарастания фронта импульса измеряется от 10% до 90% от значения полного размаха импульса. Такой подход устраняет любые погрешности, связанные с переходными углами. Длительность импульса – это время, требуемое импульсу для перехода от низкого уровня к высокому и обратно к низкому. Для удобства длительность импульса измеряется по уровню 50% от значения полного размаха импульса. На рисунке 1.12 проиллюстрированы эти измерения.

Ручные измерения параметров импульсов требуют точной настройки системы запуска осциллографа и уверенное владение техникой измерения импульсов. Поэтому на начальном этапе лучше использовать режим автоматических измерений, который необходимо самостоятельно изучить (руководство [3], подраздел 2.11.3).

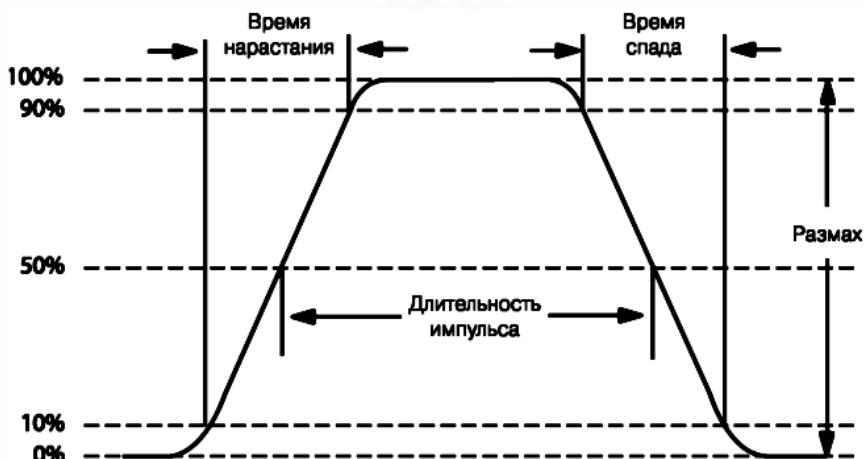


Рисунок 1.12 – Измерение времени нарастания фронта и длительности импульса.

1.2 Часть 2. Изучение отладочного комплекта и отладочных инструментов ИСР Keil

Учебное время 2 часа.

Цель работы: Привитие практических навыков по работе с ИСР Keil и технической документацией.

1.2.1 Содержание работы

- 1) Изучить состав отладочного комплекта Open32F3-D.
- 2) Создать проект на основе примера. Задать размеры стека и 'heap' согласно формуле: $0x200 + 0x80 \times [\text{номер варианта}]_{16}$. Переменным: 'a1, b1, c1, d1' типа unsigned char; 'a2, b2, c2, d2' типа unsigned short; 'a4, b4, c4, d4' типа unsigned int; 'a8, b8, c8, d8' типа unsigned long long присвоить повторяющиеся значение $0x11 + 0x9 \times [\text{номер варианта}]_{16}$ (пример: номер варианта $26_{10} = 1A_{16}$, $0x11 + 0x9 \times 0x1A = 0xFB$,

$a1=0xFB$, ..., $c4=0xFBFBFBFB$, ...), переменной '*name1*' присвоить своё имя, переменной '*name2*' фамилию в латинской транскрипции, '*name3*' номер группы.

- 3) Найти в файле карты компоновки (map-файле): затраты оперативной (RAM) и постоянной (ROM) памяти МК для вашего проекта; адрес расположения и размер стека; адрес расположения и размер таблицы векторов; адрес расположения и размер функции *main* ().
- 4) Проанализировать переменные ' $a1 \div d8$, $name1 \div 3$ ' инструментами отладки ИСР Keil. Определить адреса переменных. По адресам определить расположение в памяти. Сохранить отпечаток всей области памяти этих переменных в файл logdat.txt.
- 5) Оформить отчёт.

1.2.2 Отладочный комплект Open32F3-D Standard

Отладочный комплект Open32F3-D Standard [5] реализован как функциональная плата подключения различных модулей, показан на рисунке 1.13. Схему и описание к ней можно скачать в описании [5]. Основным модулем необходимым для работы комплекта является отладочная плата STM32F3-Discovery [6]. Функциональная плата Open32F3-D расширяет возможности отладочной платы STM32F3-Discovery, позволяя подключать различные модули.

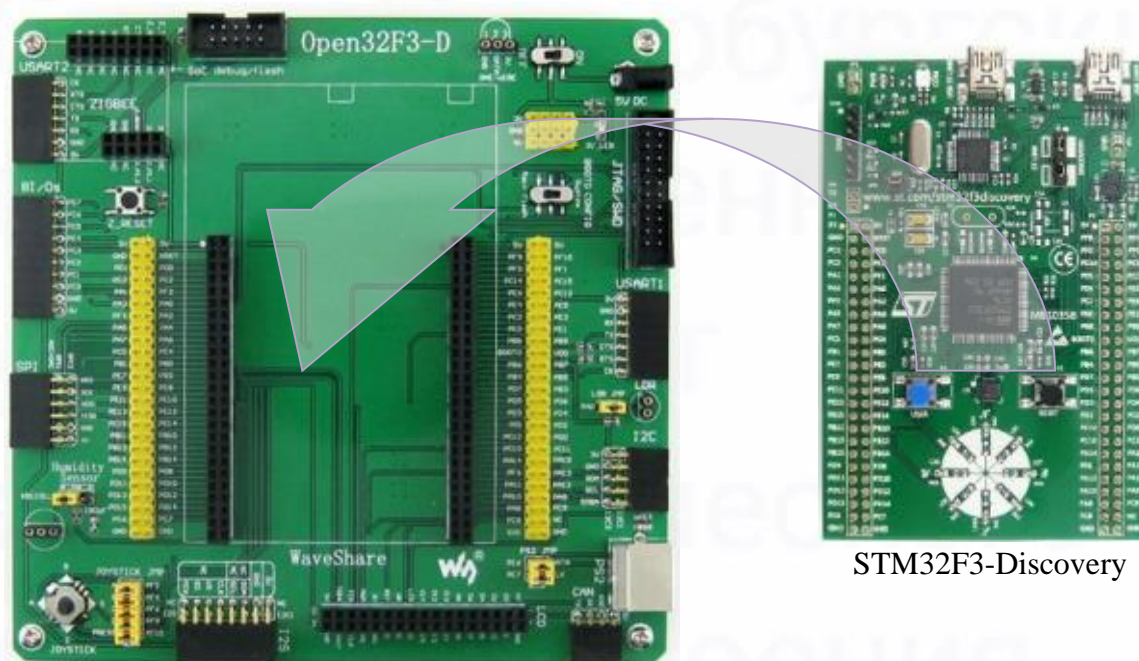
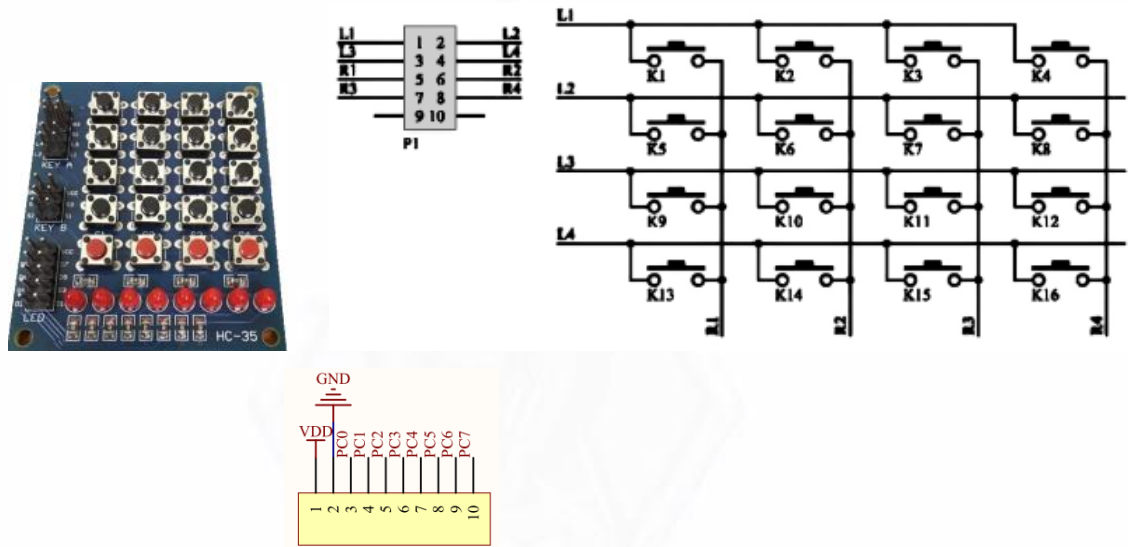


Рисунок 1.13 – Плата отладочного комплекта Open32F3-D Standard.

Перечислим модули, которые будем затрагивать при изучении курса:

1) Кнопочная панель HC-35 (или подобная 4x4 матрица кнопок), схема в части клавишной панели 4x4:

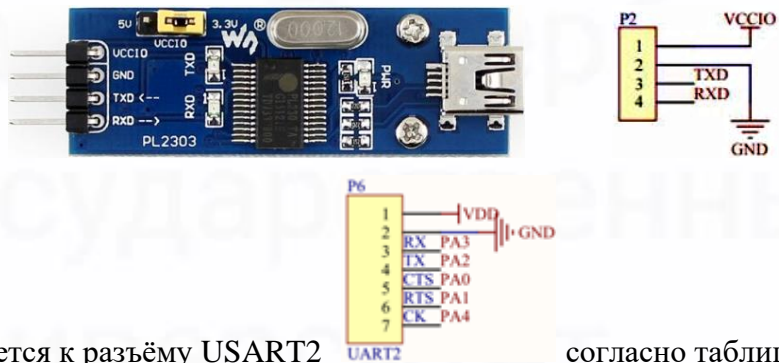


Подключается к разъёму 8I/Os согласно таблице 1.2.

Таблица 1.2 – Контакты подключение кнопочной панели.

Выходы со стороны Open32F3-D разъёма 8I/Os	PC7	PC6	PC5	PC4	PC3	PC2	PC1	PC0
Выходы со стороны кнопочной панели, разъёма KEY A(P1)	L4	L3	L2	L1	R4	R3	R2	R1

2) Преобразователь интерфейса USB-UART:



Подключается к разъёму USART2 согласно таблице 1.3.

Таблица 1.3 – Контакты подключение преобразователя интерфейса USB-UART.

Выходы со стороны Open32F3-D разъёма USART2	PA3	PA2	GND	VDD
Выходы со стороны преобразователя USB-UART	RXD	TXD	GND	VCCIO

3) LCD дисплей подключается по схеме показанной на рисунке 1.14

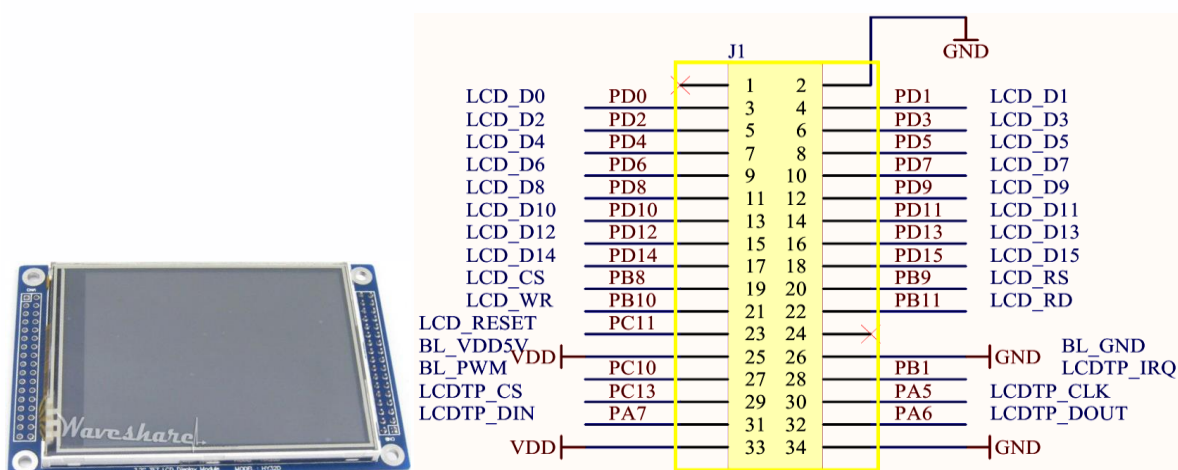
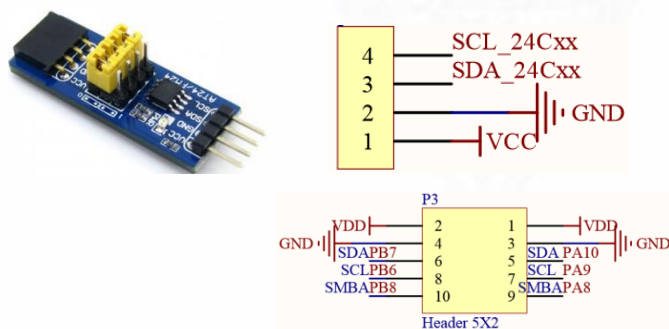


Рисунок 1.14 – Схема подключения LCD дисплея к разъёму LCD платы Open32F3-D.

4) Модуль F-RAM FM24CL16BG – сегнетоэлектрическая оперативная память:



Подключается к разъёму I2C1 I2C1 I2C2 согласно таблице 1.4.

Таблица 1.4 – Контакты подключение модуля F-RAM.

Выводы со стороны Open32F3-D разъёма I2C1	PB6	PB7	GND	VDD
Выводы со стороны модуля FM24CL16BG	SCL_24Cxx	SDA_24Cxx	GND	VCC

5) Модуль тестирования аналоговых сигналов:



Подключается по схеме, показанной в таблице 1.5.


Таблица 1.5 – Контакты подключение модуля тестирования аналоговых сигналов.

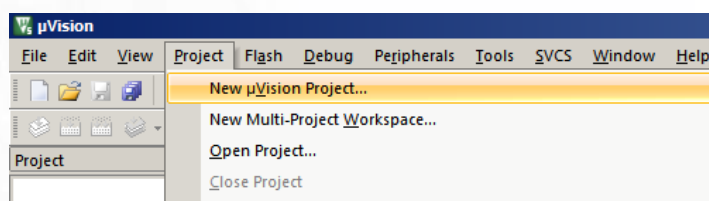
Выводы со стороны Open32F3-D разъёма SPI	PA4	PF4	PF2	GND	VDD
Выводы со стороны модуля - разъёма P1, на разъём H1 пять вольт выведены отдельным проводом.	DAC	ADC2	ADC1	GND	VCC (3.3V)

Более подробное описание модулей и их схемы здесь [5].

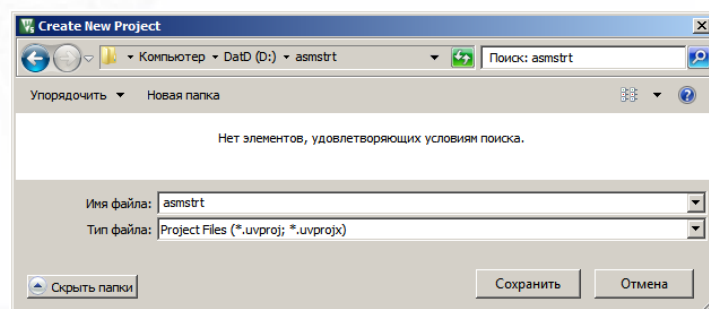
1.2.3 Порядок создания проекта в ИСР Keil

Далее при описании работы с ИСР Keil будут рассмотрены основные необходимые операции по созданию и управлению проектом. Для более детального изучения имеются ссылки на англоязычную документацию, которые помогут желающим подробнее рассмотреть интересующие их вопросы.

Итак, для создания своего первого проекта [18] запустите среду Keil (). Войдите в меню *Project >> New µVision Project* и создайте новый проект.

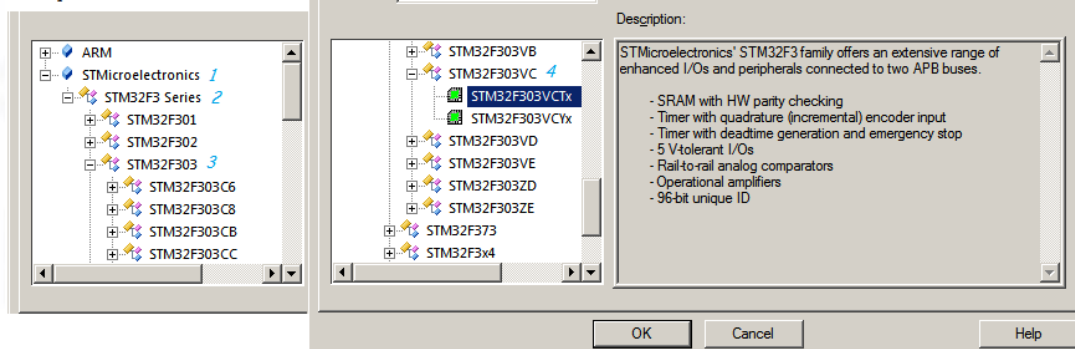


Программа попросит указать папку на диске, где в дальнейшем будут находиться файлы вашего проекта. Выберите (создайте папку) и введите имя проекта. Например, 'asmstrt', далее 'Сохранить'.



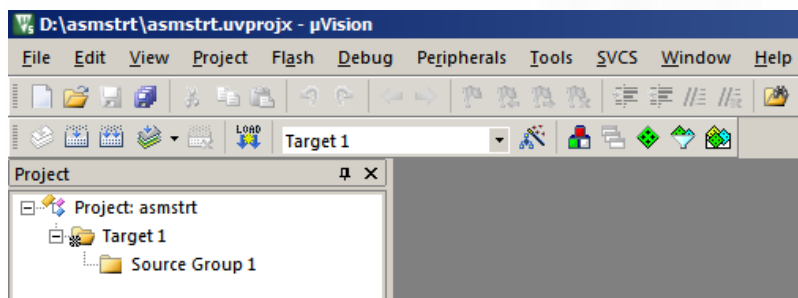
В следующем окне будет предложено выбрать тип МК, под который создаётся проект. Выбираем STM32F303VCTx. Если данной вкладки нет, значит, не установлен пакет программного обеспечения для поддержки МК STM32F303VCT6 – [8] (см. п. 1.1.4).

разворачиваем вкладку 1
STMicroelectronics
далее вкладку STM32F3 Series 2
далее вкладку STM32F303 3
и во вкладке STM32F303VC 4
выбираем STM32F303VCTx

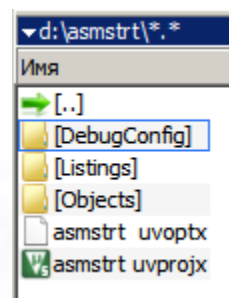


В следующем окне настройка работы с библиотеками ARM MDK-Professional и STM HAL, пока ничего не выбираем, нажимаем *OK*.

Автоматически создан проект '*asmstr*' включающий цель проекта '*Target 1*', а в нём группа файлов '*Source Group 1*'. *Project* — это репозиторий для файлов и ресурсов, необходимых для сборки программного продукта. *Target* — точно определяет, какой продукт будет собран, и содержит инструкции для сборки проекта из набора файлов:

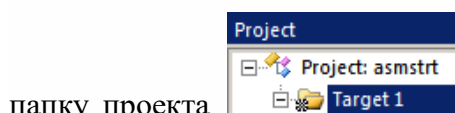



вид основного окна проекта в среде Keil



файлы проекта на диске

Переименуем названия созданных папок проекта. Для этого наводим указатель на



папку проекта ('*Target 1*') и нажимаем правую клавишу мышки. Далее в контекстном меню выбираем опцию  (*Manage Project Items...*) и в появившемся окне меняем названия на свои (например '*Target 1*' на '*asmstr*', а '*Source Group 1*' на '*src*').


Также в этом окне можно создавать группы, добавлять и удалять файлы из проекта по этим группам. С группами файлов (особенно когда их много) намного удобнее работать в этом окне, чем непосредственно через контекстное меню в основном окне.

В дальнейшем, для исключения путаницы в файлах при работе со средой Keil лучше согласовывать папки проекта с реальным расположением папок на диске. Поэтому на диске в папке проекта создадим директорию '*src*'.

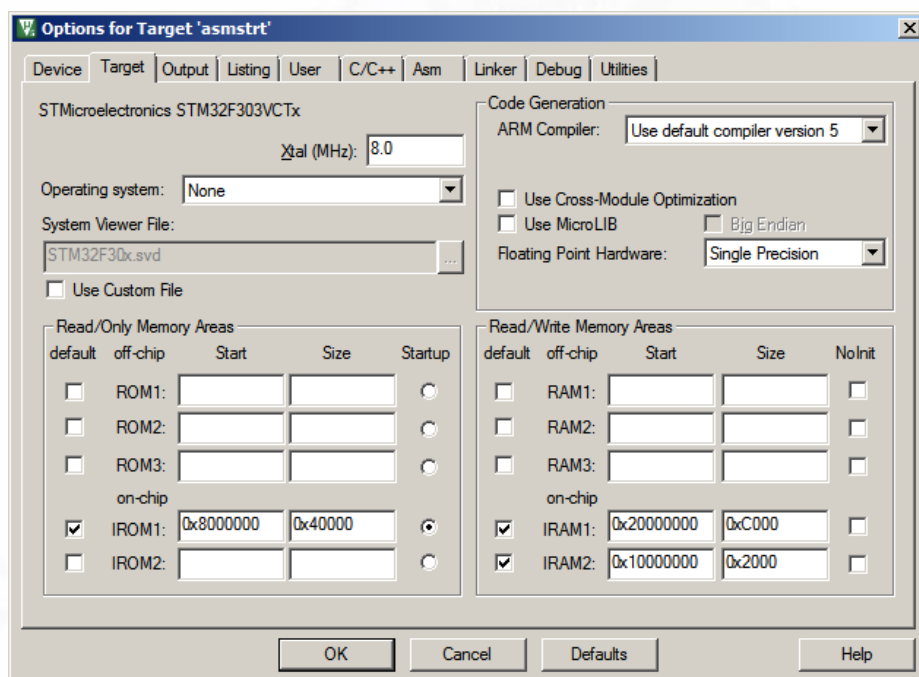
Теперь перейдем к наполнению своего проекта файлами. Для начала, это будет два файла: один ассемблерный (*.s), второй си-файл (*.c). Для создания файла наведите курсор на имя группы '*src*' и нажмите правую клавишу мышки.

Далее в контекстном меню выбираем опцию *Add New Item to Group 'src'....* В появившемся окне выбираем опцию '*Asm File (.s)*' и внизу в строке '*Name:*' вводим имя создаваемого файла (например '*strtstm32F303*'). Далее, ниже в строке '*Location:*' выбираем папку '*src*', нажатием '*Add*' создаём файл. Подобным образом создаём файл '*main.c*'.

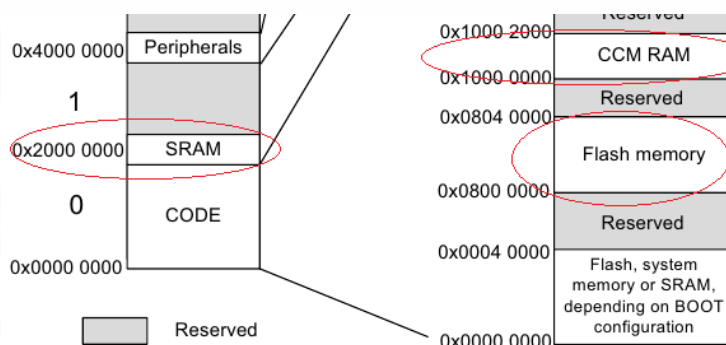
1.2.4 Свойства проекта в ИСР Keil

Прежде чем продолжить работу, настроим среду (более детально см. [19]). Для этого наведите курсор на вкладку 'asmstr' дерева проекта и нажмите правую кнопку, выберете вкладку 'Options for target', либо используйте значок  в панели инструментов или нажмите *Alt + F7*.

На открывшейся вкладке 'Target' указан начальный адрес flash-памяти микроконтроллера (IROM1), который начинается с 0x08000000 и имеет размер 0x40000 (256 Кб). Адреса оперативной памяти: IRAM1 соответствует 0x20000000 размер которой 0xA000 (40Кб); IRAM2 соответствует CCM RAM 0x10000000 размером 0x2000 (8Кб).



Эти данные автоматически выставляются при выборе типа микроконтроллера. Также их можно узнать из технического описания на STM32F303VCT6 (DS9118 [2]). В котором на странице 53 показано:



и на странице 12 в таблице 2 указан размер этих областей:

Table 2. STM32F303xB/STM32F303xC family device features and peripheral counts

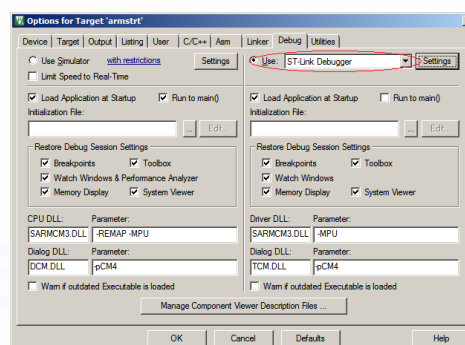
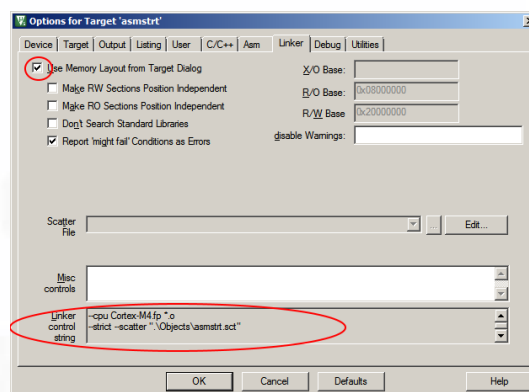
Peripheral	STM32F303Cx		STM32F303Rx		STM32F303Vx	
Flash (Kbytes)	128	256	128	256	128	256
SRAM (Kbytes) on data bus	32	40	32	40	32	40
CCM (Core Coupled Memory) RAM (Kbytes)	8					

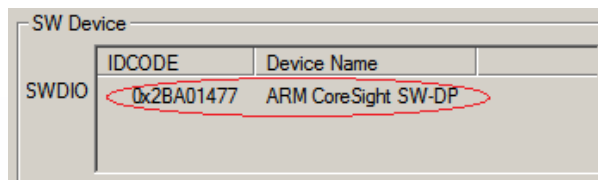
На вкладке *'Device'* можно поменять модель МК. На вкладке *'Output'* отметки должны стоять напротив *'Debug Information'* (автоматическая вставка на этапе компиляции в код проекта отладочных модулей) и *'Browse Information'*. При усложнении проектов, установка опции *'Browse Information'* влияет в сторону увеличения времени компиляции проекта. На вкладке *'Listing'* настраиваются параметры создаваемых файлов листинга, здесь по умолчанию для файла карты компоновки (map-файла) *'Linker Listing'* должны быть установлены все секции. На вкладке *'User'* можно настроить запуск различных пользовательских программ и сценариев в процессе построения проекта. На вкладке *'C/C++'* выбираются опции компиляции, пути к внешним файлам и прочее. Так как наша работа в основном будет вестись в режиме отладки, то необходимо отключить оптимизацию, выставив опцию *'Optimization:'* в положение *'Level 0(-O0)'*. Вкладка *'Asm'* предназначена для настройки процедур ассемблирования.

На вкладке компоновщика – *'Linker'*, устанавливаем опцию – *'Use Memory Layout from Target Dialog'*, для автоматического согласования раскладки областей памяти с указанным во вкладке *'Device'* типом МК. При компиляции проекта будет создан файл с расширением *.sct, где будут указаны эти области памяти. В строке *'Linker control string'* появится запись о подключении данного файла.

Переходим на вкладку *'Debug'*, из выпадающего списка *'Use:'* выбираем *'ST-Link Debugger'*. Если отсутствует отладочная плата, например при самостоятельной работе дома, то можно работать на модели МК используя симулятор, для этого выберете *'Use Simulator'* [20].

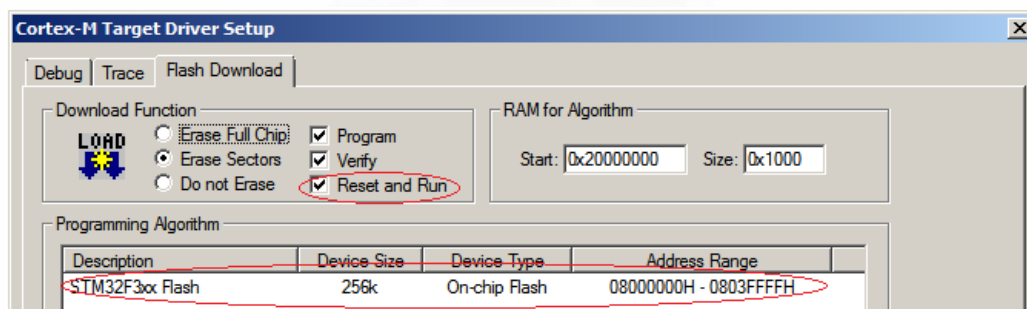
Далее нажимаем рядом со списком кнопку *'Settings'* и в открывшемся окне на вкладке *'Debug'* проверяем, что есть связь с микроконтроллером. В разделе *'SW Device'* будет написан код изготовителя микроконтроллера и тип протокола обмена.





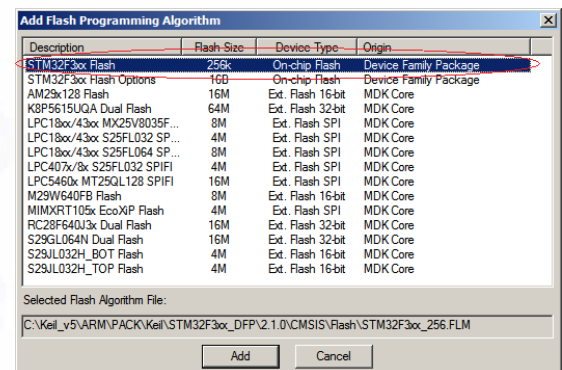
Если нет связи программатора с микроконтроллером, то будет надпись '*ST-LINK connection error*', если нет связи компьютера с программатором '*NO ST-LINK detected*'.

Далее переходим к вкладке '*Flash Download*'. Отмечаем пункт '*Reset and Run*'. Это обеспечит автоматический запуск программы после ее прошивки во flash-память микроконтроллера.



Если *Keil* не смог правильно распознать ваш микроконтроллер, то окно алгоритмы программирования Flash '*Programming Algorithm*' будет пустым. В этом случае часть программного обеспечения для стирания или загрузки на устройство Flash данного типа микроконтроллера нужно добавить вручную.

Для этого нажимаем кнопку '*Add*' и в появившемся окне выбираем тип нашего микроконтроллера STM32F3xx Flash 256k. Нажимаем кнопку *Add*. После чего он отобразится в окне '*Programming Algorithm*'.



Последняя вкладка, '*Utilities*', предназначена для настройки программирования МК из ИСР Keil. Оставляем все как есть. Должна быть включена опция '*Use Debug Driver*', тогда будут использованы наши настройки программатора из вкладки '*Debug*'.

1.2.5 Простейшая программа на ассемблере запуска основной функции 'main'

Наберите следующую программу на ассемблере. Для этого выберем двойным щелчком мыши файл 'strtstm32F303' в окне 'Project' и наполним открывшийся ассемблерный файл следующим кодом:

```
Stack_Size    EQU    0x00000080
               AREA   STACK, NOINIT, READWRITE, ALIGN=3
Stack_Mem     SPACE   Stack_Size
__initial_sp
Heap_Size     EQU    0x00000030
               AREA   HEAP, NOINIT, READWRITE, ALIGN=3
__heap_base
Heap_Mem      SPACE   Heap_Size
__heap_limit

PRESERVE8
THUMB
AREA  RESET, DATA, READONLY
EXPORT __Vectors
__Vectors    DCD     __initial_sp
               DCD     Reset_Handler
__Vectors_End
__Vectors_Size    EQU    __Vectors_End - __Vectors
               AREA   |.text|, CODE, READONLY
Reset_Handler    PROC
               IMPORT main
               LDR     R0, =main
               BX      R0
               ENDP
               EXPORT __initial_sp
               EXPORT __heap_base
               EXPORT __heap_limit
END
```

Этот же код в ИСР Keil:

```
strstm32F303.s  main.c
1  Stack_Size  EQU 0x00000080
2      AREA    STACK, NOINIT, READWRITE, ALIGN=3
3  Stack_Mem   SPACE   Stack_Size
4  __initial_sp
5  Heap_Size   EQU 0x00000030
6      AREA    HEAP, NOINIT, READWRITE, ALIGN=3
7  __heap_base
8  Heap_Mem    SPACE   Heap_Size
9  __heap_limit
10
11      PRESERVE8
12      THUMB
13      AREA    RESET, DATA, READONLY
14      EXPORT  __Vectors
15  __Vectors   DCD  __initial_sp
16              DCD  Reset_Handler
17  __Vectors_End
18  __Vectors_Size  EQU  __Vectors_End - __Vectors
19      AREA    |.text|, CODE, READONLY
20  Reset_Handler  PROC
21      IMPORT  main
22      LDR R0, =main
23      BX  R0
24      ENDP
25      EXPORT  __initial_sp
26      EXPORT  __heap_base
27      EXPORT  __heap_limit
28  END
```

При наборе кода следует учитывать, что в строках, где символы (метки адресов) идут с начала строки, то так и должно быть. Не допускается вставка даже пробела между ними и началом строки. В остальных случаях, пробелы в начале строки обязательны, в противном случае компилятор примет команду за метку (более подробно по синтаксису ассемблера ARM – chapter 3 [21]).

Далее выберем файл *'main.c'* и добавим следующий код:

```
int main (void) {
    for(;;){}
    return 0; }
```

Так как программа работы микроконтроллера никогда не должна заканчиваться, вставлен бесконечный цикл *for*. В противном случае можем получить непредсказуемое поведение микроконтроллера.

После набора программы, нажатием клавиши **F7**, скомпилируем программу. Также возможно запустить сборку кнопкой *'Build'* на главной панели или через меню *'Project >> Build Target'*.

После сборки, в окне *'Build Output'* получаем:


```

Build Output
Rebuild started: Project: asmstrt
*** Using Compiler 'V6.19', folder: 'C:\dev\Keil_v5\ARM\ARMCLANG\Bin'
Rebuild target 'asmstrt'
src/main.c(4): warning: 'return' will never be executed [-Wunreachable-code-return]
    return 0;
    ^
1 warning generated.
compiling main.c...
assembling strtstm32F303.s...
linking...
Program Size: Code=208 RO-data=24 RW-data=0 ZI-data=128
".\Objects\asmstrt.axf" - 0 Error(s), 1 Warning(s).
Build Time Elapsed: 00:00:01

```

где 'Code' - размер кода, 'RO-data' - константы в коде (*Read Only Data*), 'RW-data' - переменные (*Read Write Data*), 'ZI-data' - переменные, инициализируемые нулями (*Zero-Initialized Data*). Исходя из этих данных можно подсчитать, сколько оперативной (RAM) и постоянной (ROM) памяти нужно нашему проекту: $RAM = RW + ZI = 0 + 128 (0x80) = 128$ байт и $ROM (FLASH \text{ память}) = Code + RO + RW = 208 + 24 + 0 = 232$ байт. Если во время компоновки (*linking...*) возникла ошибка типа: *Error: L6320W: Ignoring --entry command. Cannot find argument 'Reset_Handler'*, то установите опцию 'Use Memory Layout from Target Dialog' на вкладке компоновщика - 'Linker'.

1.2.6 Файл карты компоновки

Карта компоновки создается по умолчанию, выводится в файл [имя проекта].*map* и содержит справочную информацию о результатах сборки проекта. Компоновщик (*Linker*) выдает в карту компоновки отчет о вычислении переменных, размещении входных секций, файлов и библиотечных объектов в выходных секциях исполняемой программы. Данные по размеру памяти необходимой исполняемому коду можно также посмотреть и в этом файле. Физически он расположен в папке проекта 'Listings'.

Для открытия его в среде Keil наведите курсор на вкладку 'asmstrt' дерева проекта и нажмите правую кнопку, в контекстном меню выберите вкладку 'Open Map File'.

В открывшемся *map*-файле, промотав в конец, последние две строки как раз и показывают необходимые RAM и ROM:

Total RO	Size (Code + RO Data)	232 (0.23kB)
Total RW	Size (RW Data + ZI Data)	128 (0.12kB)
Total ROM	Size (Code + RO Data + RW Data)	232 (0.23kB)

Также в разделе 'Memory Map of the image' показаны адреса, по которым расположены стек:

Exec Addr	Load Addr	Size	Type	Attr	Idx	E Section Name	Object
0x20000000	-	0x00000080	Zero	RW	1	STACK	strtstm32f303.o
адрес расположения стека		размер стека					

и область памяти с неупорядоченным хранением данных ('*heap*'), а также адреса, по которым происходит загрузка исполняемого кода в ROM.

В нашем файле раздел '*heap*' отсутствует, так как компилятор убрал его, посчитав, что раз не используется, значит, не нужен, о чём и оповестил после секции перекрестных ссылок (*Section Cross References*):

```
=====
Removing Unused input sections from the image.

Removing strtstm32f303.o(HEAP), (48 bytes).
Removing main.o(.text), (0 bytes).
Removing main.o(.ARM.exidx.text.main), (8 bytes).
Removing main.o(.ARM.use_no_argv), (4 bytes).

4 unused section(s) (total 60 bytes) removed from the image.
=====.
```

В разделе таблица символов ('*Image Symbol Table*') показаны метки и их адреса, используемые в программе:

Reset_Handler	0x0800008d	Thumb Code	4	strtstm32f303.o(.text)
STACK	...	Section	32	strtstm32f303.o(STACK)
__initial_sp	0x20000080	Data	0	strtstm32f303.o(STACK)
__Vectors	...	Data	4	strtstm32f303.o(RESET)
main	0x08000139	Thumb Code	4	main.o(i.main)

Так как мы подключили файл на языке *си* (функцию *main*) то компилятор ARM автоматически подключает код монитора отладки '*Angel*', библиотеки которого усложняют восприятие тар-файла. Поэтому пользуйтесь поиском при анализе файла карты компоновки (более детально см. [22]).

1.2.7 Разбор основных конструкций программы на ассемблере

Наша программа на ассемблере (см. п. 1.2.5) имеет следующую структуру:

- 1) Объявление области стека (*stack*).
- 2) Объявление области памяти с неупорядоченным хранением данных (*heap*).
- 3) Таблица векторов прерываний.
- 4) Процедура, исполняемая после сброса (*reset handler*).

- 1) Объявление области стека (stack) состоит из следующих директив:

```
1 Stack_Size EQU 0x00000080
2         AREA STACK, NOINIT, READWRITE, ALIGN=3
3 Stack_Mem SPACE Stack_Size
4 __initial_sp
```

В первой строке объявляем метку `Stack_Size` со значение `0x80` (128 байт) - размер стека. Здесь `EQU` (equal, равно) директива эквивалентности и присваивания. Во второй строке объявление секции `STACK`. Директивой `AREA` создается отдельная секция в памяти с названием `STACK`. За названием следуют атрибуты: `NOINIT` – данные в секции остаются как есть (не инициализируются); `READWRITE` – секция доступна для чтения и записи; `ALIGN=3` – выравнивание секции по границе кратной восьми байт ($2^3=8$) (chapter 12 [21]).

В третьей строке выделяется пространство в памяти заданного размера (`Stack_Size`) для области стека. Директива `SPACE` просто резервирует место в памяти. В метке `Stack_Mem` прописывается начальный адрес резервируемой области памяти (дно стека), а в метке `__initial_sp` следующий адрес после этой области (вершина стека).

- 2) Объявление области памяти с неупорядоченным хранением данных (heap) начинается с 5ой строки и по структуре идентична объявлению стека:

```
5 Heap_Size EQU 0x00000030
6         AREA HEAP, NOINIT, READWRITE, ALIGN=3
7 __heap_base
8 Heap_Mem SPACE Heap_Size
9 __heap_limit
```

Объявляем метку `Heap_Size` равной `0x30` (48 байт) – размер 'heap' которая нам нужна. Объявление секции `HEAP` атрибуты имеют те же значения что и при объявлении стека. `__heap_base` – начальный адрес 'heap', `__heap_limit` – адрес, следующий после области 'heap'.

Названия меток для стека и 'heap' такие как: `Stack_Size`, `Stack_Mem`, `__initial_sp`, `Heap_Size`, `__heap_base`, `__heap_limit`, `Heap_Mem`, и др. стандартизированы под восприятие компилятора языка си. Как будет видно далее, в проектах ассемблерный файл кода запуска МК формируется автоматически и менять в нём рекомендуется только размеры стека и 'heap'.

Далее в 10ой строке `PRESERVE8` указывает компоновщику сохранять 8-байтное выравнивание стека требование стандарта AAPCS, а `THUMB` указывает ассемблеру интерпретировать последующие инструкции как THUMB-инструкции (особенности ARM архитектуры).

- 3) Таблица векторов прерываний (адресов):

```
12         AREA RESET, DATA, READONLY
13         EXPORT __Vectors
14 __Vectors DCD __initial_sp
15         DCD Reset_Handler
16 __Vectors_End
17 __Vectors_Size EQU __Vectors_End - __Vectors
```

Сначала создаём секцию RESET с атрибутами: DATA – секция содержит данные, а не инструкции; **READONLY** – секция только для чтения. Далее в строке 13 директивой **EXPORT** указываем компилятору, что метка `__Vectors` является глобальной и видна за пределами нашего файла. В строке 14 директивой **DCD** распределяем в ПЗУ четыре байта памяти под метку `__initial_sp` (адрес вершины стека), а метке `__Vectors` сопоставляется адрес начала таблицы векторов прерываний `0x08000000` который указан в *map*-файле в разделе *Image Symbol Table* (п. 1.2.6):

<code>__Vectors</code>	<code>0x08000000</code>	<code>Data</code>	<code>4</code>	<code>strtstm32f303.o(RESET)</code>
имя метки	адрес	тип секции	размер байт	название секции

Далее в строке 15 распределяем следующие четыре байта, это адрес первой команды подпрограммы, исполняемой после сброса (Reset) – указатель на процедуру `Reset_Handler`. Код этой процедуры будет выполняться всегда после сброса МК.

Таблица векторов содержит только адреса обработчиков прерываний и указатель вершины стека, причём адреса подпрограмм обработки прерываний располагаются строго в порядке указанным в таблице 'Table 82. STM32F303xB/C/D/E, STM32F358xC and STM32F398xE vector table' руководства *RM0316* [1]. На рисунке 1.15 показано совмещение таблицы векторов в программе и в руководстве.

Table 82. STM32F303xB/C/D/E, STM32F358xC and STM32F398xE vector table

Position	Priority	Type of priority	Acronym	Description	Address
-	-	-	-	Reserved	0x0000 0000
-	-3	fixed	Reset	Reset	0x0000 0004
-	-2	fixed	NMI	Non maskable interrupt. The RCC Clock Security System (CSS) is linked to the NMI vector.	0x0000 0008
-	-1	fixed	HardFault	All class of fault	0x0000 000C

12	<code>AREA</code>	<code>RESET, DATA, READONLY</code>
13	<code>EXPORT</code>	<code>__Vectors</code>
14	<code>__Vectors</code>	<code>DCD __initial_sp</code>
15		<code>DCD Reset_Handler</code>
16	<code>__Vectors_End</code>	
17	<code>__Vectors_Size</code>	<code>EQU __Vectors_End - __Vectors</code>

Рисунок 1.15 – Сопоставление таблицы векторов руководства *RM0316* и кода программы.

Если прерывания не использовать, таблицу можно сократить до двух строк. Поэтому на этом наша таблица заканчивается и далее идёт метка `__Vectors_End` – адрес конца таблицы векторов прерываний. Далее, вычисляем размер таблицы векторов прерываний `__Vectors_Size`, просто вычитая из конечного адреса начальный. Для отображения значений меток в *map*-файле необходимо объявить их глобальными директивой **EXPORT**.

Таблица векторов прерываний является неотъемлемой составляющей для запуска процессора, а также работы стека. Это связано с тем, что процессор при запуске начинает выполнение кода с адреса 0x00000000. А именно, читает первые четыре байта таблицы векторов (значение вершины стека) и записывает их в регистр R13(SP) (Stack Pointer - указатель стека). Затем считывает следующие четыре байта с адреса 0x00000004 и записывает их в регистр R15(PC - Program Counter - счётчик команд). Считанные четыре байта являются адресом первой команды процедуры Reset_Handler. Так как загрузка исполняемого кода происходит из памяти ПЗУ (Flash) располагающейся по адресу 0x08000000 (п. 0), то микроконтроллер отражает таблицу векторов на пространство памяти начальной загрузки (0x00000000), т.е. происходит совмещение адресного пространства двух регионов (более подробно см. стр. 62, RM0316 [1]).

4) Код процедуры, исполняемой после сброса (Reset_Handler), это место, с которого начинается выполнение программы пользователя.


```
18      AREA    |.text|, CODE, READONLY
19  Reset_Handler  PROC
20      IMPORT  main
21      LDR R0, =main
22      BX  R0
23      ENDP
```

Сначала в строке 18 объявляем секцию |.text| с атрибутами: **CODE** – секция содержит инструкции (исполняемый код); **READONLY** – секция только для чтения. Название секции |.text| используется общепринятое, но может быть любым другим. Строка 19 начинается с объявления метки Reset_Handler процедурой с помощью директивы **PROC** и далее следует исполняемый этой процедурой код до директивы **ENDP**. Весь наш код состоит из двух команд: команды загрузки в регистр R0 указателя на функцию main и команды перехода по этому указателю. Строки 24, 25, 26 объявляют метки __initial_sp, __heap_base, __heap_limit идентификаторами доступными в других файлах и библиотеках. И заканчивается наш файл директивой **END**, которая информирует компилятор об окончании файла.

1.2.8 Инструменты отладчика в ИСР Keil

Ядро микроконтроллера ARM Cortex-M4 оснащено технологией отладки и трассировки ARM CoreSight (подробнее [23]), которая позволяет осуществлять доступ к памяти «на лету», то есть во время исполнения рабочей программы, без использования каких-либо дополнительных программных модулей. При выполнении программы разработчик непрерывно получает информацию об актуальном состоянии памяти и переменных

контроллера. Существует возможность устанавливать точки останова как триггер для какой-либо переменной.

Переход в режим отладки запускается нажатием *Ctrl + F5*, либо через главное меню *Debug >> Start/Stop Debug Session*, либо значком  на панели программных инструментов. После того как Keil предупредит нас о неполноценности версии с ограничением на размер исполняемого кода в 32 Кб, нажимаем *OK* и продолжаем работу. Данное напоминание будет появляться постоянно при переходе в режим отладки. В режиме отладки окно программы примет вид, показанный на рисунке 1.16.

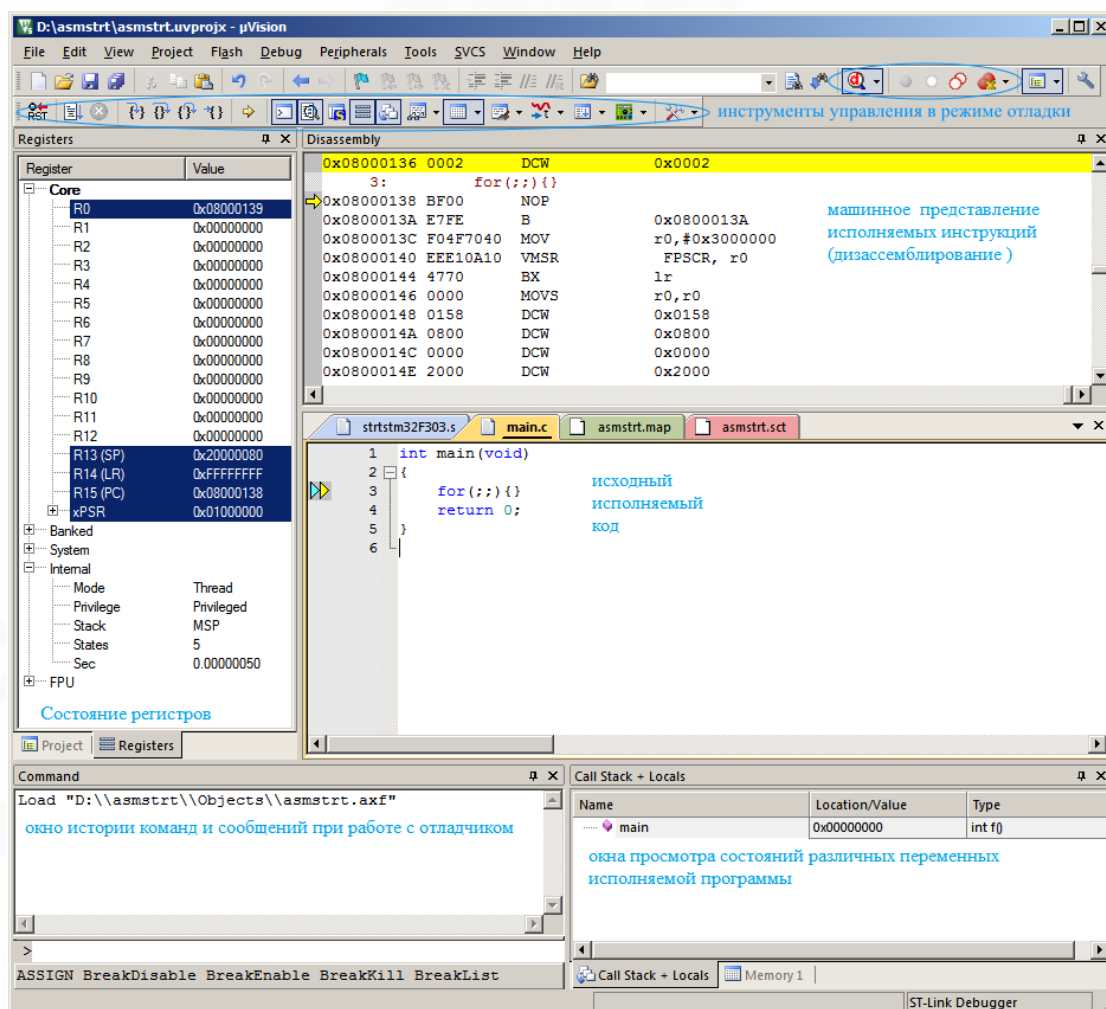


Рисунок 1.16 – Режим отладки ИСР Keil.

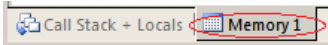

Окно состояния регистров - 'Registers'. Как видно из рисунка 1.16, в окне метка `__initial_sp` уже загружена в R13(SP), а в регистр R15(PC) уже загружен адрес следующей исполняемой команды функции *main*. Регистр R14 (LR - регистр связи, используется для запоминания адреса возврата при вызове подпрограмм) инициализирован значением 0xFFFFFFFF. Регистр xPSR содержит флаги результатов выполнения арифметических и

логических операций, состояние выполнения программы и номер обрабатываемого в данный момент прерывания. В документации об этом регистре пишут во множественном числе, так как к его трём частям можно обращаться как к отдельным регистрам. Эти «подрегистры» называются: APSR — регистр состояния приложения (тут хранятся флаги), IPSR — регистр состояния прерывания (содержит номер обрабатываемого прерывания) и EPSR — регистр состояния выполнения (более подробно стр. 18 PM0214 [15]).

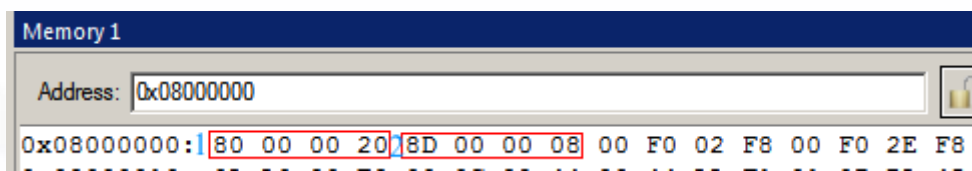
В окне 'Disassembly' (дизассемблирования) наблюдаем несколько столбцов. Первый столбец – это адреса, по которым хранятся исполняемые команды, второй столбец – это машинные коды этих команд, третий столбец мнемоническое обозначение команд на ассемблере вместе с операндами:

→ 0x0800013A	E7FE	B	0x0800013A
адрес команды	код команды	код ассемблера	операнд

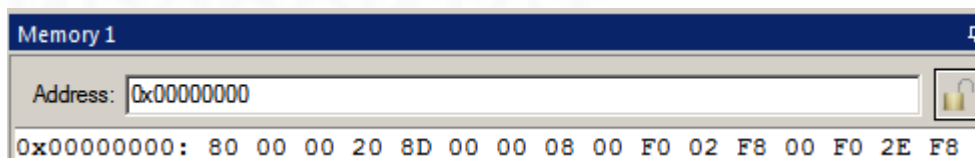
жёлтая стрелка указывает на текущую команду, подлежащую исполнению.

Для просмотра областей памяти нажмите вкладку 'Memory 1' в окне просмотра состояний различных переменных . Если вкладка 'Memory 1' отсутствует, то вызовите её через инструмент на панели:  или через меню View >>Memory Windows >> Memory 1(2,3,4).

В появившемся окне 'Memory 1' в поле 'Address:' введите адрес начала flash памяти 0x08000000 и нажмите ввод. Получим текущий отпечаток памяти:



из которого видно, что первые четыре байта это указатель вершины стека - адрес 0x20000080 (архитектура little-endian, порядок байт от младшего к старшему), вторые четыре байта это адрес функции Reset_Handler 0x0800008D. Далее сравним эту область с памятью по нулевому адресу. Изменим адрес на 0x00000000:




Как видно значения абсолютно одинаковые, так как эти адресные регионы совмещены.

Выйдем из режима отладки (повторное нажатие *Ctrl* + *F5*) и допишем следующий код в функцию *main*:

```

1 int main(void)
2 {
3     volatile unsigned char a1=0x12;
4     volatile unsigned short a2=0x1234;
5     volatile unsigned int a4=0x12345678;
6     volatile unsigned long long a8=0x1234567887654321;
7     volatile char name[]="Vladimir Solov'ev";
8     for(;;){}
9     return 0;
10 }

```

Откомпилируем (F7) и запустим отладку (Ctrl + F5). Переключимся на окно *Disassembly* выбрав его щелчком мыши. В инструментах отладки нажимаем F11, либо значок  в панели инструментов, либо через меню *Debug >> Step*. Нажимая F11, наблюдаем за исполнением ассемблерных команд инициализации первой переменной 'a1': сначала командой *MOVS* в регистр R2 записывается константа 0x12; затем командой *STRB* значение регистра R2 сохраняется в памяти по адресу $R13(sp) + 0x2B = 0x20000048 + 0x2B = 0x20000073$

```

0x080001AC 2212      MOVS      r2,#0x12
2:          volatile unsigned char a1=0x12;
→0x080001AE F88D202B  STRB      r2,[sp,#0x2B]

```

Посмотрим область памяти по этому адресу. Во вкладке *Memory 1* в поле *Address:* вводим адрес 0x20000078. Получим текущий отпечаток памяти:

```

Memory 1
0x20000073
0x20000073: 12 00 00 00

```

По адресу 0x20000073 сохранилось значение нашей переменной. При этом обратите внимание, как меняется регистр счётчика команд R15(PC) и значение регистра R2 в окне '*Registers*'. Расположение кодов операций команд в памяти также можно посмотреть через окно '*Memory*'.

Давайте теперь сохраним память по этому адресу в файл. Для этого в окне *Command* введем команду *log > a1.log* которая создаст файл регистрации *a1.log*. Далее *d &a1,&a1* , где *d* (команда *Display* [24]) сохранит значения памяти начиная с адреса *&a1* и заканчивая адресом *&a1*, всего один байт. Можно также указывать непосредственно сами адреса (*d 0x20000073,0x20000073*). Далее командой *log off* закроем файл регистрации *a1.log*.

```

Command
Load "D:\\asmstrt\\Objects\\asmstrt.axf"
log > a1.log
d &a1,&a1+3
0x20000078 12 00 00 00
log off

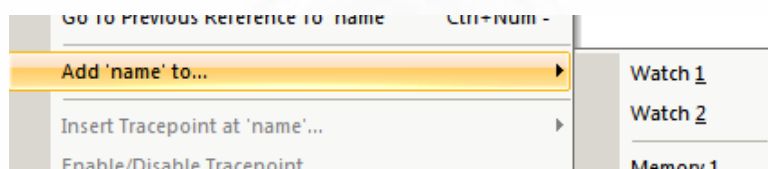
```


В результате получится файл *a1.log* со снимком значения в памяти переменной 'a1'.

Также значение переменной можно посмотреть на вкладке *Call Stack + Locals*:

Call Stack + Locals		
Name	Location/Value	Type
main	0x00000000	int f()
a1	0x12	auto - uchar
a2	0x1234	auto - ushort


Но удобнее пользоваться инструментом *Watch* (список наблюдаемых переменных). В режиме отладки, щёлкните на переменную правой клавишей мыши, в появившемся контекстное меню выберите *'Add 'название переменной' to ...'* и в выпавшем списке выберите *'Watch 1'*. Например, для переменной *'name'*:



Также окно *'Watch 1'* можно открыть через панель инструментов  или через меню *View->Watch Windows -> Watch 1(2)* и вбить имя нужной переменной вручную в открывшемся окне в поле *'Name'*.

Добавим переменные *'a1,a2,a4,a8'* в окно *'Watch 1'*. И по мере нажатия *F11*, с каждым шагом выполнения, наблюдаем инициализацию значениями наших переменных:

Watch 1		
Name	Value	Type
name	0x20000054 "Vladimir Solov'ev"	uchar[18]
a8	0x1234567887654321	__uint64
a4	0x12345678	uint
a2	0x1234	ushort
a1	0x12	uchar

Для получения адресов наших переменных воспользуемся операцией «взятия адреса» введя амперсанд (&)  перед переменными *'a1,a2,a4,a8'* в окне *'Watch 1'*. И далее по полученным адресам сопоставим со значениями в памяти как показано на рисунке 1.17.

Name	Value	Type
name	0x20000054 "Vladimir Solov'ev"	uchar[18]
&a8	0x20000068	pointer
&a4	0x20000070	pointer
&a2	0x20000074	pointer
&a1	0x20000078	pointer

Memory 1
 0x20000054

0x20000054:	56 6C 61 64
0x20000058:	69 6D 69 72
0x2000005C:	20 53 6F 6C
0x20000060:	6F 76 27 65
0x20000064:	76 00 00 00
0x20000068:	21 43 65 87
0x2000006C:	78 56 34 12
0x20000070:	78 56 34 12
0x20000074:	34 12 00 00
0x20000078:	12 00 00 00

name[]="Vladimir Solov'ev"

a8=0x1234567887654321

a4=0x12345678

a2=0x1234

a1=0x12

Рисунок 1.17 – Представление переменных в памяти МК.

Как видно из рисунка 1.17 данные выровнены на границу по четыре байта. Это позволяет компилятору лучше оптимизировать код. Если об этом забыть и не учитывать в программе (например, при работе с памятью), то могут возникать неожиданные ошибки при копировании блоков памяти и т.п.

Сохраним отпечаток этой памяти в файл (log > dat.log ; d &name[0],&a1+3 ; log off):

```
log > dat.log
d &name[0],&a1+3
0x20000054  56 6C 61 64 69 6D 69 72 - 20 53 6F 6C 6F 76 27 65 Vladimir Solov'e
0x20000064  76 00 00 00 21 43 65 87 - 78 56 34 12 78 56 34 12 v...!Ce.xV4.xV4.
0x20000074  34 12 00 00 12 00 00 00 -                               4.....
log off
```

Содержимое файла dat.log:

```
0x20000054  56 6C 61 64 69 6D 69 72 - 20 53 6F 6C 6F 76 27 65 Vladimir Solov'e
0x20000064  76 00 00 00 21 43 65 87 - 78 56 34 12 78 56 34 12 v...!Ce.xV4.xV4.
0x20000074  34 12 00 00 12 00 00 00 -                               4.....
```

Для более детального изучения команд отладчика воспользуйтесь [24].

1.3 Требования к содержанию отчёта

Первая часть отчёта должна содержать:

- 1) Сохранённую осциллограмму по образцу рисунка 1.18 (согласно номеру варианта), с указанием расчёта по ней амплитуды и частоты.

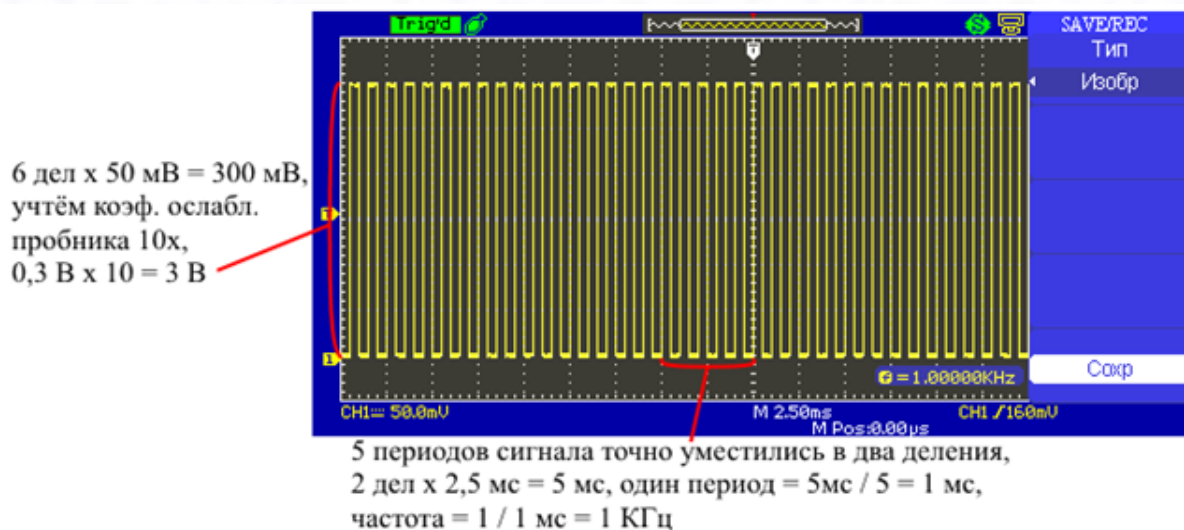
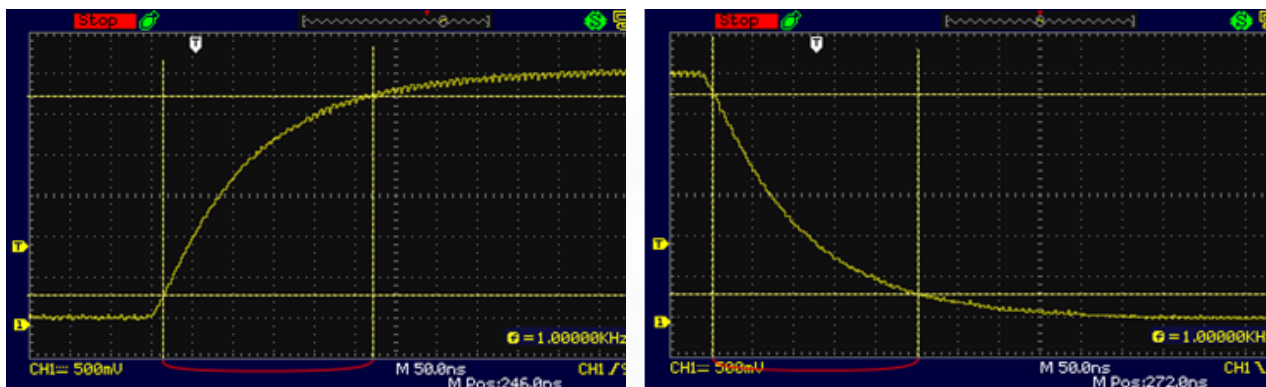


Рисунок 1.18 – Образец сохранённой осциллограммы для настроек: 50 мВ/дел, 2.5 мс/дел.

- 2) Время нарастания фронта (Rise Time), спада среза (Fall Time) меандра рисунок 1.19, при установленной чувствительности по напряжению согласно варианта. Сравнить порядок времени нарастания с периодом.



- а) время нарастания фронта $5\text{ дел} \cdot 50\text{ нс} = 250\text{ нс}$ б) время спада среза $5\text{ дел} \cdot 50\text{ нс} = 250\text{ нс}$

Рисунок 1.19 – Определенное в ручном режиме время нарастания и спада сигнала.

Вторая часть отчёта должен содержать:

- 1) Текст задания согласно варианту, с указанием размера стека, значений переменных.
- 2) Программу проекта.
- 3) Выписку из файла карты компоновки: затрат оперативной и постоянной памяти проекта; адрес расположения и размер стека; адрес расположения и размер таблицы векторов; адрес расположения и размер функции main.
- 4) Адреса расположения в памяти переменных 'a1,b1,...,c8,d8, name1÷3' (подобно рис. 1.17).
- 5) Содержимое файла logdat.txt с отпечатком участка памяти содержащим значения переменных 'a1,b1,...,c8,d8, name1÷3'.

1.4 Контрольные вопросы

- 1) Требования безопасности во время занятий.
- 2) Что запрещается студенту? Номер телефона медицинской службы и пожарной охраны.
- 3) Название отладочного комплекта, отладочной платы, интегрированной среды разработки.
Названия основных используемых программных библиотек.
- 4) Перечень необходимой документации для программирования МК STM32F3xx.
- 5) Назначение осциллографа, порядок проверки работоспособности осциллографа.
- 6) Основные системы настройки осциллографа, назначение.
- 7) Системы вертикального и горизонтального отклонения осциллографа, органы управления и их настройка.
- 8) Системы запуска и отображения осциллографа, режимы работы, органы управления системой и их настройка.

- 9) Порядок измерения амплитуды и частоты сигнала в ручном и автоматическом режимах.
- 10) Измерение времени нарастания фронта и спада среза импульса.
- 11) Состав отладочного комплекта Open32F3-D.
- 12) Понятие проекта, свойства проекта в ИСР Keil.
- 13) Определить адреса размещения стека, области памяти с неупорядоченным хранением данных, таблицы векторов.
- 14) Общая структура программы на ассемблере запуска МК.
- 15) Режим отладки, назначение окон и панелей инструментов.
- 16) Инструменты слежения за состоянием переменных в режиме отладки.
- 17) Работа с памятью в режиме отладки, порядок сохранения отпечатка памяти на диск.
- 18) Типы данных, количество байт, выделяемое каждому типу.
- 19) Найти в файле отпечатка памяти значение указанной переменной и выделить.

Литература

- 1 RM0316. Reference manual STM32F303xB/C/D/E, STM32F303x6/8, STM32F328x8, STM32F358xC, STM32F398xE advanced ARM®-based MCUs. URL: https://www.st.com/resource/en/reference_manual/rm0316-stm32f303xbcd-e-stm32f303x68-stm32f328x8-stm32f358xc-stm32f398xe-advanced-armbased-mcus-stmicroelectronics.pdf
- 2 DS9118. Datasheet STM32F303xB STM32F303xC. URL: <https://www.st.com/resource/en/datasheet/stm32f303vc.pdf>
- 3 Осциллографы цифровые UDS1000. Руководство по эксплуатации. URL: http://micromir-nn.ru/Oscil/uniontest/UnionTest_UDS1000.pdf
- 4 Осциллографы. Основные принципы измерений. Учебное пособие. URL: <https://download.tek.com/document/03U-8605-5%20Scopes%20Manual.pdf>
- 5 Open32F3-D Standard, STM32F3 Development Board. Part Number: Open32F3-D Standard. URL: <https://www.waveshare.com/open32f3-d-standard.htm>
- 6 STM32F3DISCOVERY. Discovery kit with STM32F303VC MCU. URL: <https://www.st.com/en/evaluation-tools/stm32f3discovery.html>
- 7 MDK-ARM. URL: <https://www.keil.com/demo/eval/arm.htm>
- 8 STMicroelectronics STM32F3-Discovery Software Pack. URL: https://www.keil.com/dd2/pack/#Keil.STM32F3xx_DFP
- 9 STM32CubeProg. STM32CubeProgrammer software for all STM32. URL: <https://www.st.com/en/development-tools/stm32cubeprog.html>
- 10 Common Microcontroller Software Interface Standard (CMSIS). URL: https://arm-software.github.io/CMSIS_5/Build/html/index.html
- 11 STSW-STM32108. STM32F301x/302x/303x/334x DSP and standard peripherals library. URL: <https://www.st.com/en/embedded-software/stsw-stm32108.html>
- 12 UM1581. Description of STM32F30xx/31xx Standard Peripheral Library. URL: https://www.st.com/resource/en/user_manual/um1581-description-of-stm32f30xx31xx-standard-peripheral-library-stmicroelectronics.pdf
- 13 STM32CubeF3. STM32Cube MCU Package for STM32F3 series. URL: <https://www.st.com/en/embedded-software/stm32cubef3.html>
- 14 UM1786. Description of STM32F3 HAL and low-layer drivers. URL: https://www.st.com/resource/en/user_manual/um1786-description-of-stm32f3-hal-and-lowlayer-drivers-stmicroelectronics.pdf

15 PM0214. STM32F3 Series, STM32F4 Series, STM32L4 Series and STM32L4+ Series Cortex®-M4 programming manual. URL:

https://www.st.com/resource/en/programming_manual/pm0214-stm32-cortexm4-mcus-and-mpus-programming-manual-stmicroelectronics.pdf

16 UM1570. User manual. Discovery kit with STM32F303VC MCU. URL:

https://www.st.com/resource/en/user_manual/um1570-discovery-kit-with-stm32f303vc-mcu-stmicroelectronics.pdf

17 MB1035-F303C-D01 Board Schematic. URL:

https://www.st.com/resource/en/schematic_pack/mb1035-f303c-d01_schematic.pdf

18 UM1562. Getting started with software and firmware environments for the STM32F3DISCOVERY Kit. URL: https://www.st.com/resource/en/user_manual/dm00062662-getting-started-with-software-and-firmware-environments-for-the-stm32f3discovery-kit-stmicroelectronics.pdf

19 The Options for ... dialogs allow you to configure the development environment. URL:

<https://developer.arm.com/documentation/101407/0538/Dialogs/Project/Options>

20 Simulation. URL: <https://www2.keil.com/mdk5/simulation/>

21 Assembler User Guide. URL:

<https://developer.arm.com/documentation/dui0473/m/preface/about-this-book>

22 Arm Compiler Reference Guide. URL:

<https://developer.arm.com/documentation/101754/latest>

23 CoreSight™ Technology. URL: <https://www2.keil.com/coresight/>

24 Debug Commands. URL:

<https://developer.arm.com/documentation/101407/0538/Debug-Commands>