к

М Г Т У им. Н.Э.Баумана.
Кафедра "Компьютерные системы и сети".

Старший преподаватель кафедры ИУ6
Аристов Б.К.

Методические указания
к лабораторной работе по дисциплине «Микроэлектроника и микропроцессорные системы"
для студентов специальности «Метрология и взаимозаменяемость»

Знакомство с программой «Keil µVision», языками
«ассемблер» и C++ для микроконтроллеров с ядром ARM
Cortex M3.
Арифметические и логические команды. Отладка
микроконтроллерных систем.

Лабораторная работа №1

МОСКВА 2012 г

ОГЛАВЛЕНИЕ

Лабораторная работа №1.

Знакомство с программой «Keil µVision», языками «ассемблер» и C++ для микроконтроллеров с ядром ARM Cortex M3.
Арифметические и логические команды. Отладка микроконтроллерных систем.

Цель работы - освоить работу с симулятором микроконтроллеров µVision фирмы Keil. Составить программу для микроконтроллера с ядром ARM Cortex M3, отладить ее.

Продолжительность работы – 4 академических часа.

Цель работы

Изучение инструментальных средств и интегрированной среды разработки программного обеспечения для микроконтроллеров, а также изучение этапов технологии разработки и отладки программ для микроконтроллерных систем.

1. Теоретическая часть.

Микроконтроллер – это вычислительное устройство, в состав которого входит процессор, память программ, память данных, порты ввода – вывода, вспомогательные схемы, расположенные в одном корпусе. Структурная схема микроконтроллеров приведена на рис 1.1.

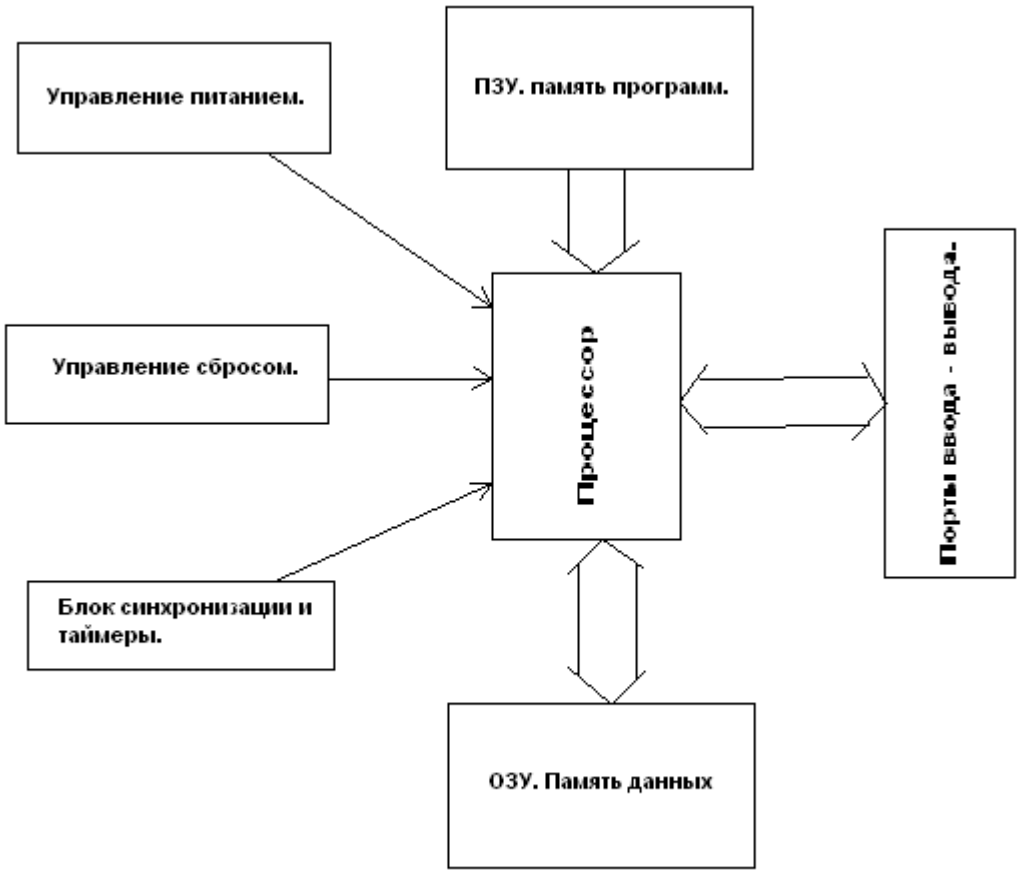


Рис. 1.1

1.1. Типы микроконтроллеров.

Микроконтроллеры можно разделить на следующие основные типы:

- встраиваемые (embedded) восьмиразрядные микроконтроллеры,

- 16- и 32-разрядные микроконтроллеры,
- цифровые сигнальные процессоры(DSP).

Встраиваемое управляющее устройство разрабатывается на основе микропроцессорной и микроконтроллерной систем, в состав которого входят периферийные устройства, дополнительные элементы памяти. Задачи, решаемые встроенными системами можно разделить на два класса:

- управление событиями в реальном времени,
- управление потоками данных.

1.2. Архитектура микроконтроллеров.

Существуют две архитектуры микроконтроллеров – гарвардская и принстонская. Принстонская архитектура (машина Фон-Неймана) разработана в Принстонском университете. Микроконтроллеры с такой архитектурой имеют общую память программ и данных.

1.3. Система команд.

Различают два набора команд – CISC и RISC.

CISC - (Complex Instruction Set Computers), т.е. микроконтроллеры (микропроцессоры) со сложной системой команд.

RISC – (Reduced Instruct Set Computers) микроконтроллеры (микропроцессоры) с сокращенной системой команд.

1.4. Структура памяти микроконтроллеров.

1.5. Аккумуляторная и регистр-регистровая архитектуры.

В микроконтроллерах используется аккумуляторная или регистр-регистровая архитектура.

В аккумуляторной архитектуре (см. рис. 1.5.1) при выполнении команды одним из операндов находится в аккумуляторе второй в регистре. Результат выполнения операции всегда помещается в аккумулятор. Аккумуляторную архитектуру имеют микроконтроллеры МК51. Любая арифметическая или логическая операция выполняется за четыре команды:

- Поместить первый операнд в аккумулятор,
- Поместить второй операнд в регистр (один из регистров регистрового файла),
- Выполнить команду,
- Запомнить результат (если это необходимо).

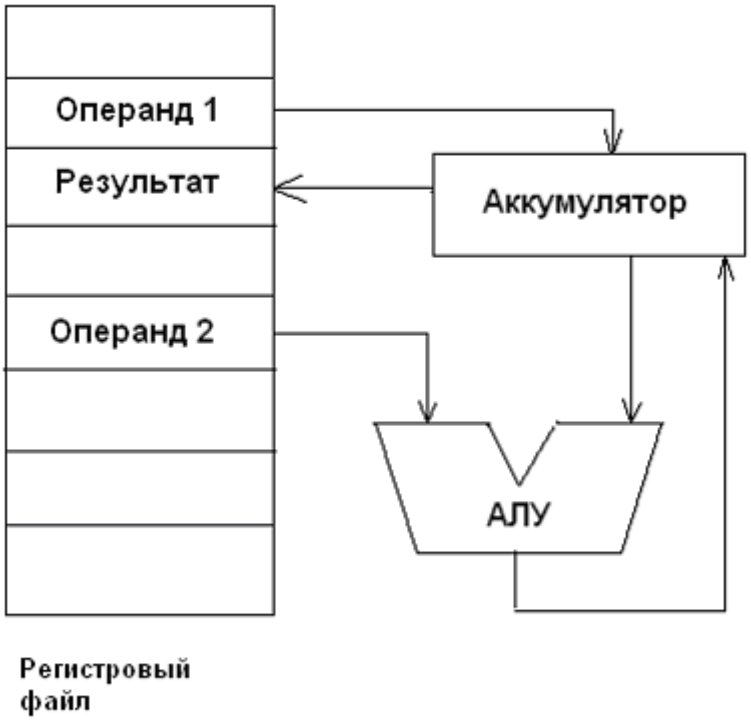


Рис. 1.5.1.

Команда регистр - регистровой архитектуры (см. рис. 1.5.2.)может содержать два или три операнда. Все операнды арифметических и логических команд берутся из файлового регистра. Результат помещается в файловый регистр. При такой архитектуре нет необходимости в командах пересылки данных, что сокращает объем программы и ускоряет ее выполнение.

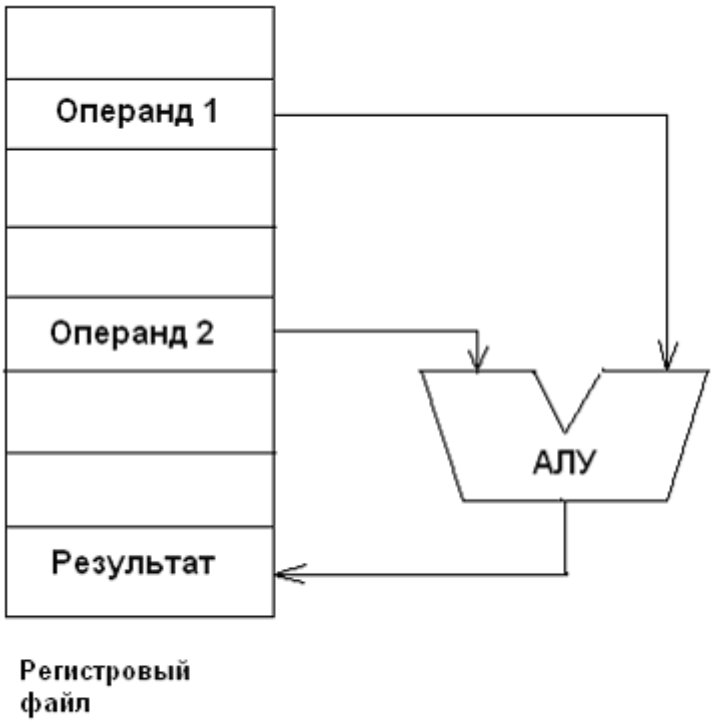


Рис. 1.5.2.

2. Этапы разработки систем на микроконтроллерах.

Разработка микропроцессорной системы (как и любой другой) начинается с разработки технического задания (Т.З.) на проектируемую систему. Особенностью микропроцессорных систем является то, что микропроцессоры это программируемые устройства, поэтому при

выработке Т.З. в возможности проектируемой системы закладываются максимальные возможности функции управления. На этом этапе становиться ясно какой микропроцессор необходим для проектируемой системы.

На следующем этапе разрабатывается алгоритм управления. Рассматриваются несколько вариантов алгоритма управления. Критерием выбора служит вариант алгоритма управления, в котором доля программного обеспечения выше аппаратной.

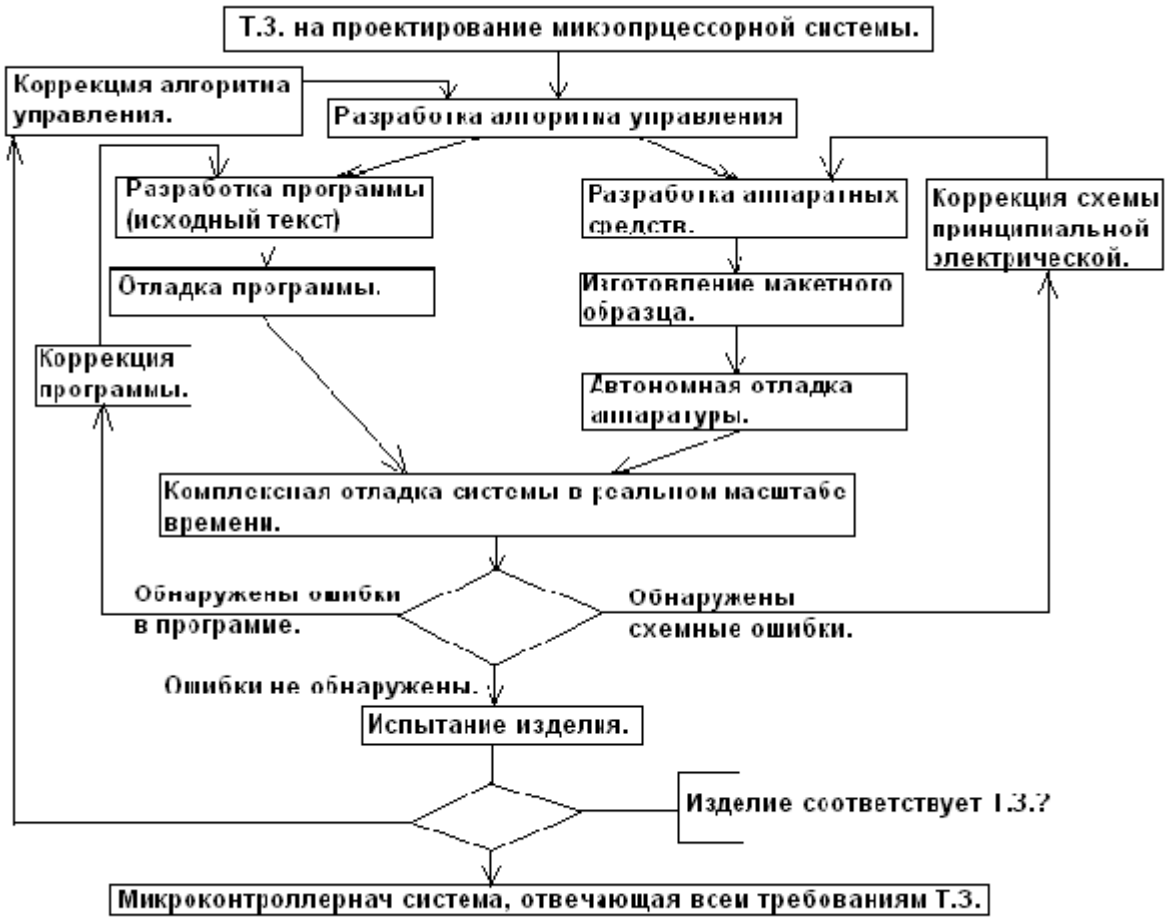


Рис.

В 70-е годы фирмой Intel был предложен метод внутрисхемной эмуляции. Этот метод требует наличия аппаратных средств необходимых для реализации проекта и средства управления отладкой. Основным недостатком такого подход является дороговизна аппаратных средств.

Интегри́рованная среда́ разрабо́тки, ИСР (IDE, Integrated development environment или integrated debugging environment) — система программных средств, используемая разработчиками микроконтроллерных систем для разработки и отладки программного обеспечения (ПО), аппаратнойчасти и комплексной отладки МК системы. Обычно среда разработки включает в себя:

текстовый редактор,
компилятор и/или интерпретатор,
средства автоматизации сборки,
отладчик

3. µVision.

Программный комплекс Keil µVision создан фирмой Keil (http://www.keil.com). Для работы с STM32L-DISCOVERY унеобходима версия не ниже 4.53.

3.2. Запись программ на языке ассемблера.

Запись программ на языке ассемблера осуществляется в определенном формате. Формат команды содержит поле метки, поле операции, поле операндов и поле комментария. Одно поле от другого отделяется пробелом. Если метка отсутствует, то первым символом в строке должен быть пробел. Метка и комментарий могут отсутствовать. Поля отделяются друг от друга хотя бы одним пробелом. Число символов в строке - не более 80.

Метка это идентификатор, представляющий собой сцепление букв и цифр, начинающейся с буквы. При написании метки можно использовать только буквы латинского алфавита. Ассемблер допускает использовать в качестве буквы символ _ (подчеркивание). Число символов в метке не должно превышать 31. Метка всегда завершается символом «:» (двоеточие).

Операция. В поле операции записывается мнемоническое обозначение команды микроконтроллера. Список мнемокодов команд приведен в приложении 2.

Операнды. В поле записываются операнды (или один операнд), участвующие в операции. Операнды разделяются запятой. При записи операндов пробелы не допускаются!

Комментарий. Комментарий начинается символом «;» (точка с запятой). Далее записывается текст комментария (можно использовать русские буквы).

Примеры:

```
Поле     Поле          Поле              Поле
метки    команды       операндов         комментариев
                MOV        R3, #57       ; Записать в регистр R3 число 57.
METKA01  ADD        R5, R1, R3    ; Сложить содержимое регистров R1 и R3,
                                          ; результат записать в регистр R5.
CIKL_UMNOJENIA
                MUL        R0, R1. R2    ; Перемножить содержимое регистров R1 и R2
                                          ; результат поместит в регистр R0.
```

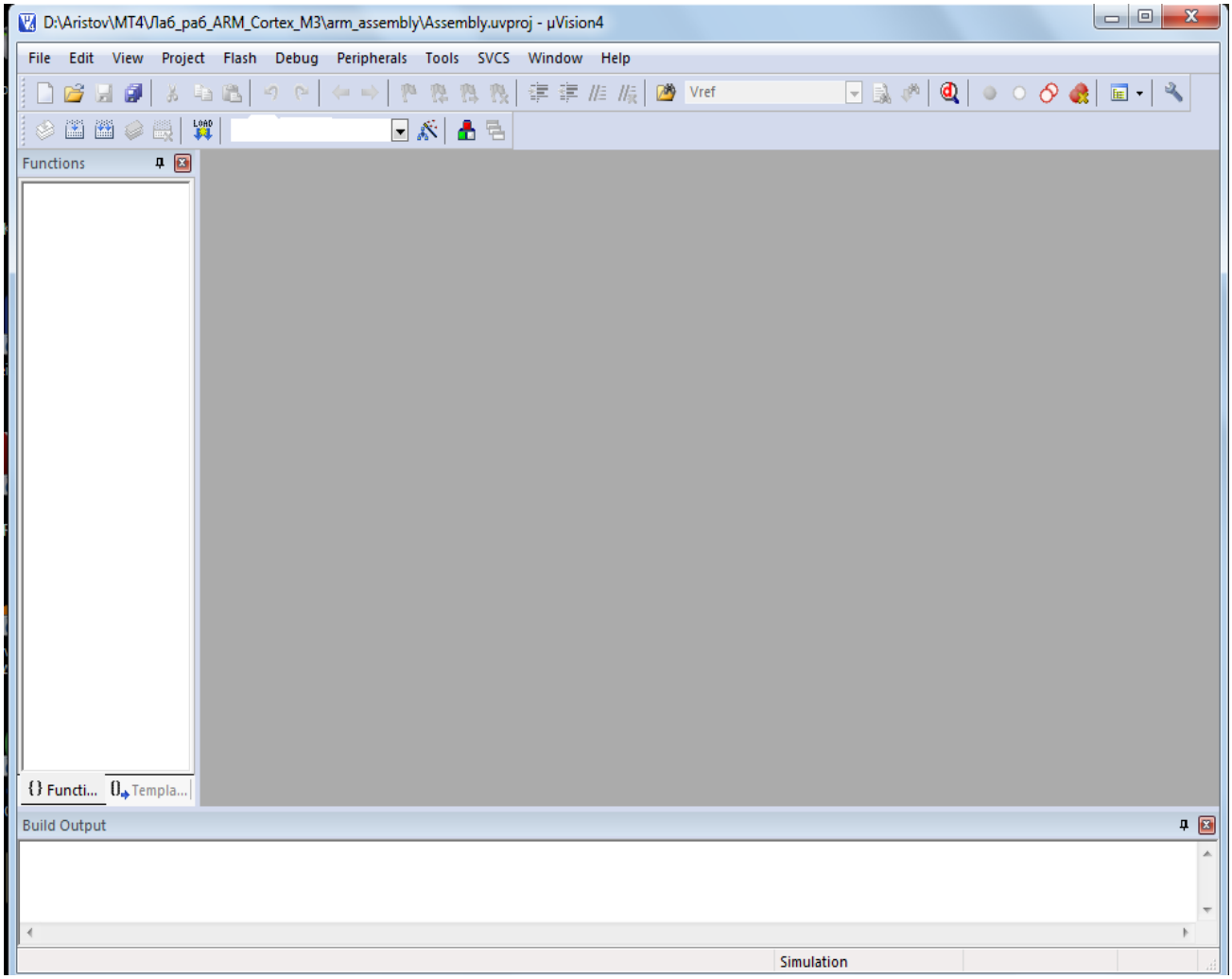
Создание проекта для МК ARM Cortex M3 на ассемблере. Keil µVision.

Вызвать программу Keil µVision можно несколькими способами:

1. На рабочем столе найдите иконку 

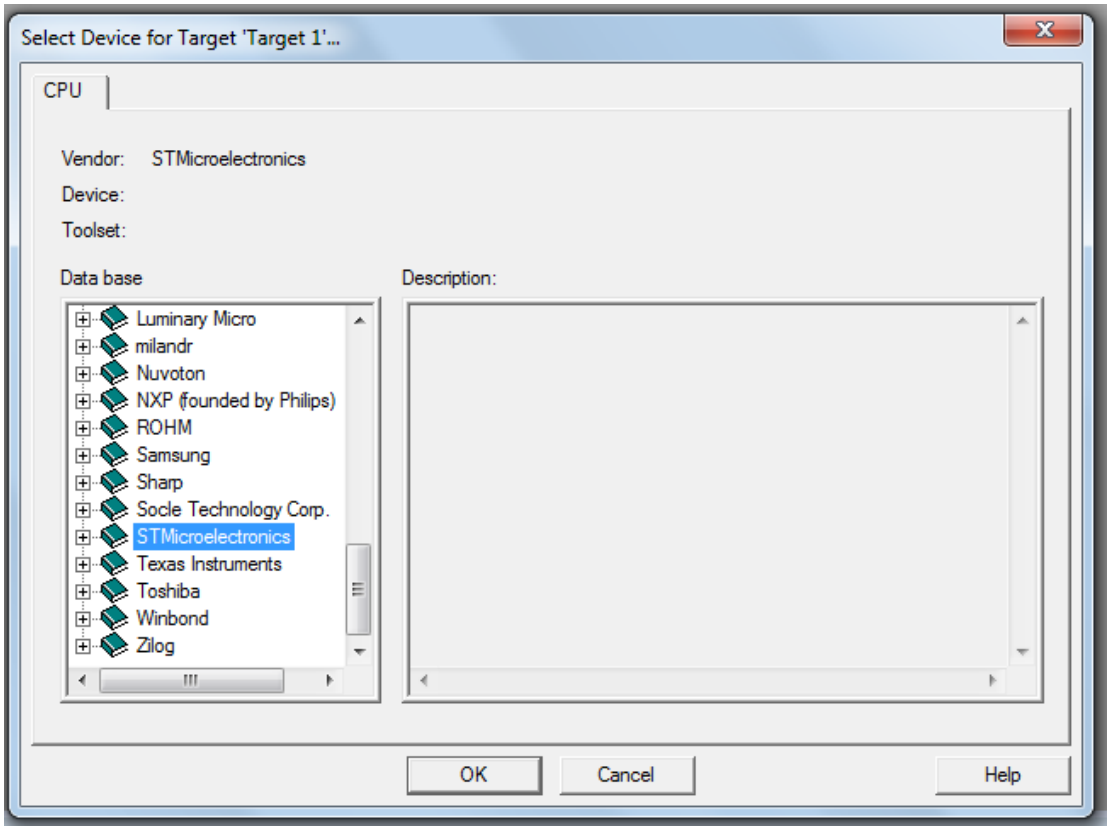
.
2. "Пуск" - ("Все программы") - Keil uVision4.
3. Из командной строки: "Пуск" - "Все программы" - "Стандартные" - "Выполнить". В открывшемся окне набрать C:\Keil\UV4\Uv4.exe и нажать клавишу Enter.

Если запуск прошел успешно, то на мониторе появится окно приведенное на рис. .

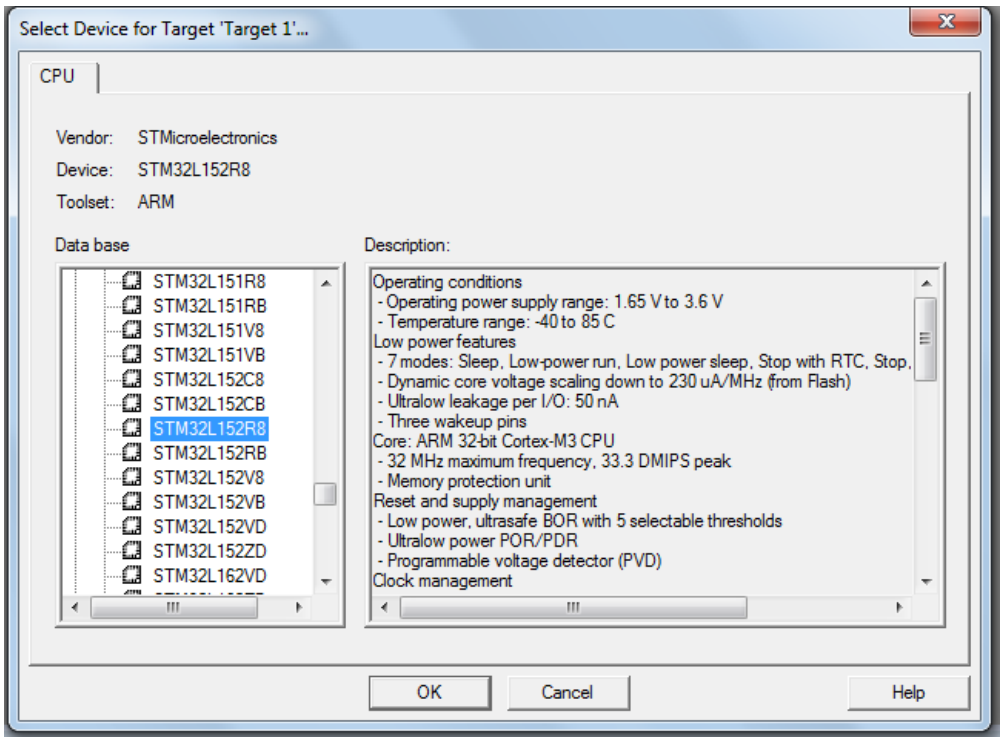

D:\Aristov\MT4\Лаб_раб_ARM_Cortex_M3\arm_assembly\Assembly.uvproj - µVision4
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
Vref
Functions
{} Functi...
{} Templa...
Build Output
Simulation

Для создания проекта выполним "Project" - "New µVision Project". В открывшемся окне выбираем диск, папку (можно создать новую папку и в нее поместить создаваемый проект) а в строке "Имя файла" записываем имя проекта и нажимаем Enter.

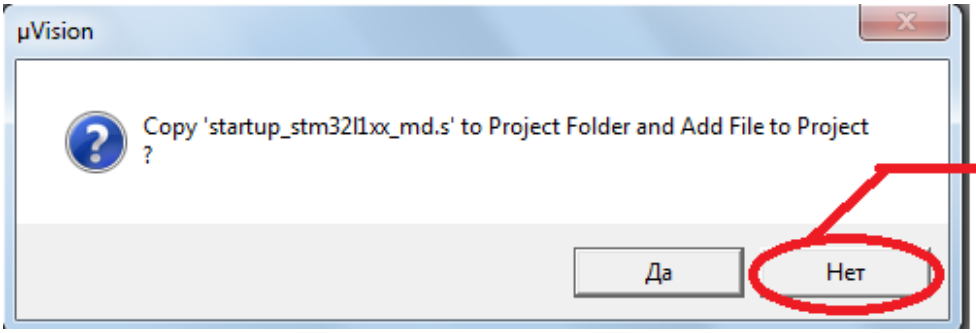
В открывшемся окне (рис.) выбираем микроконтроллер (МК) с ядром ARM фирмы STMicroelectronics(рис.), установленный на отладочной плате STM32L-DISCOVERY - STM32L152RBT6 (рис.).



Справа в окне появляются технические данные для выбранного МК.



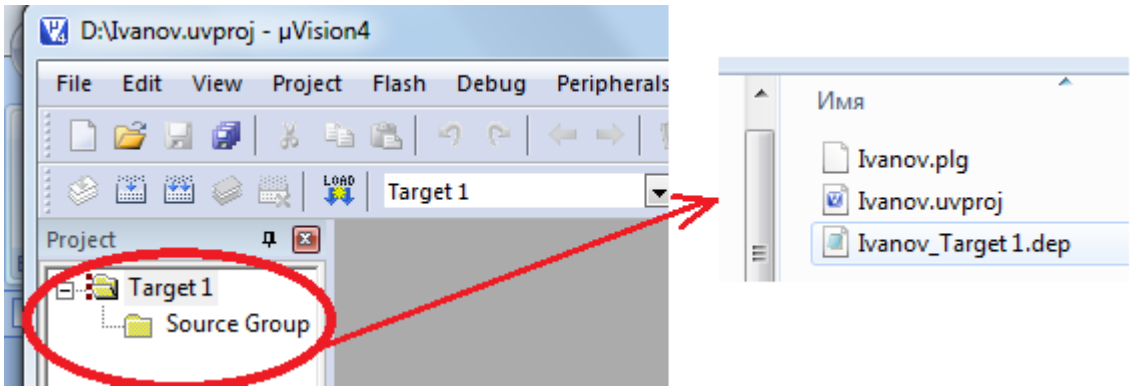
Нажимаем ОК. Keil µVicion сообщает, что он может вставить в текст программы заголовочные файлы для "C++".



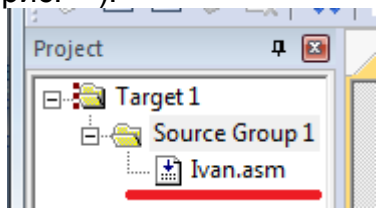
Если программа на ассемблере, выбираем НЕТ.

Т.к. программа создается на ассемблере отвечаем: **НЕТ!**

Созданы необходимы файлы для начала работы над проектом (см. рис.).



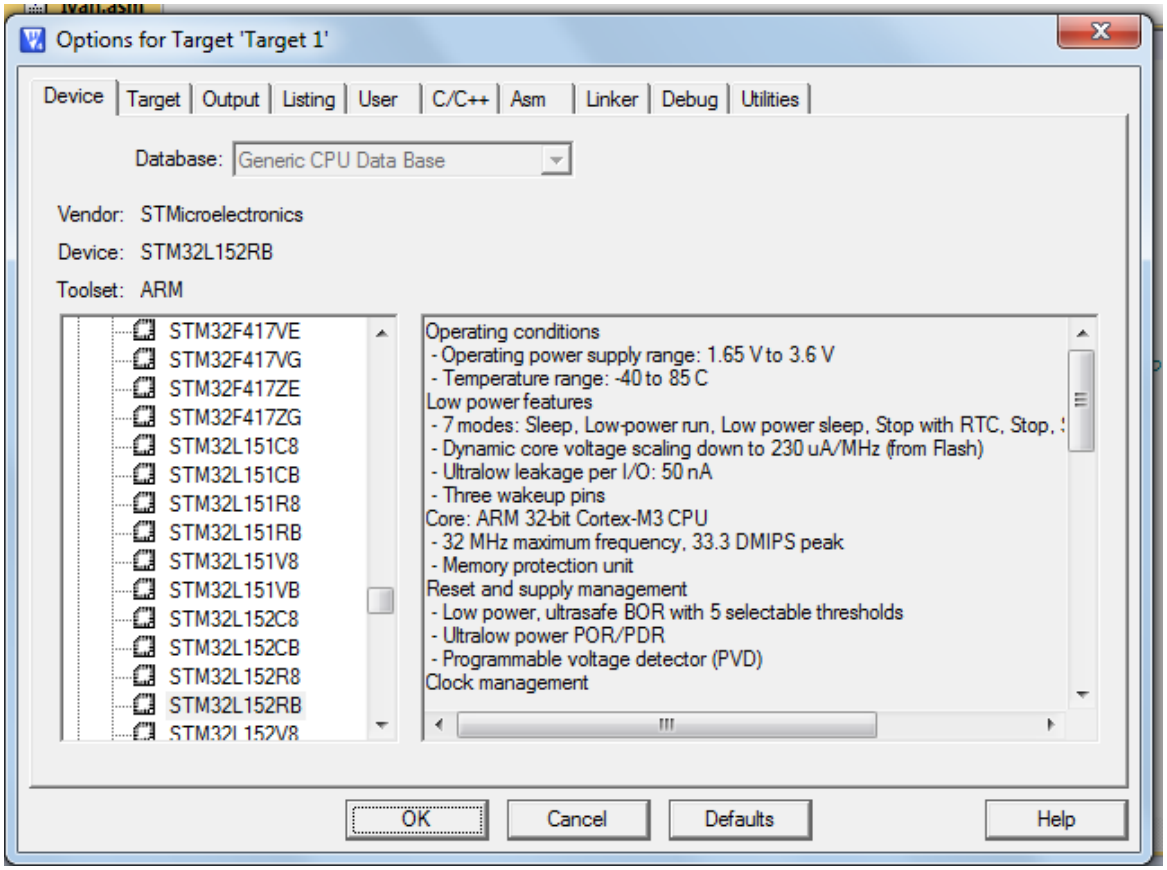
Добавим к этим файлам файл с программой на ассемблере. Это можно сделать в программе "проводник". У создаваемого файла должно быть расширение *.**asm**. В папке Ivanov создадим файл Ivan.asm и добавим его в проект. Для этого щелкнем по папке Source Group 1, в открывшемся окне найдем файл Ivan.asm и нажмем кнопку Add. Файл добавлен (см. рис.).



Сделаем все необходимые установки, чтобы проект работал правильно. Это можно сделать несколькими способами:

1. Найдите в панели инструментов кнопку  и щелкните по ней,
2. И окне Project подведите курсор к названию проекта и нажмите правую клавишу мышки. В открывшемся окне выберите Options for Target 'Target 1',
3. Project - Options for Target 'Target 1',
4. Alt + F7.

На экране появляется окно настройки проекта (рис.):



Рисунок

Device

На первой закладке ."Device" выбирается тип МК. Тип МК мы выбрали .

Target

Окно описания карты памяти (начальный адрес и размер памяти программ и начальный адрес и размер памяти данных).

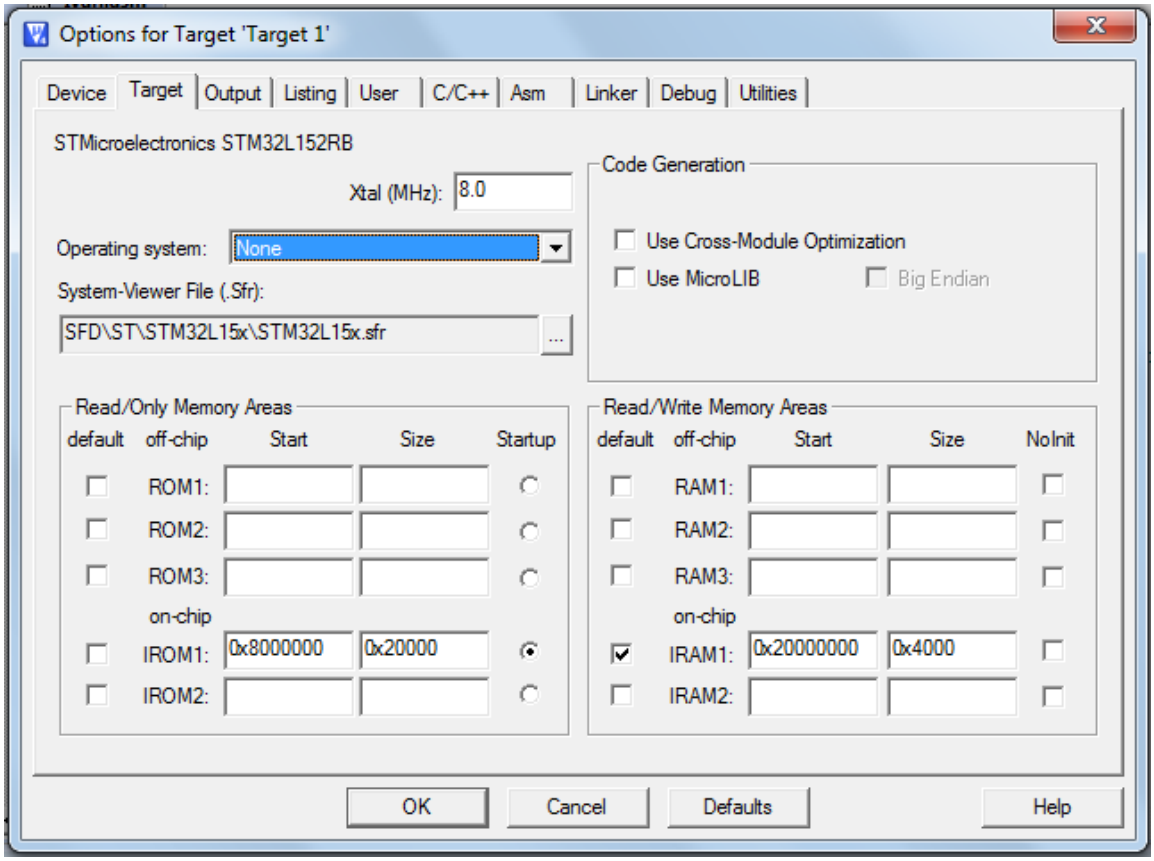


Рисунок .

Output

Окно установки для вывода результатов работы компановщика.

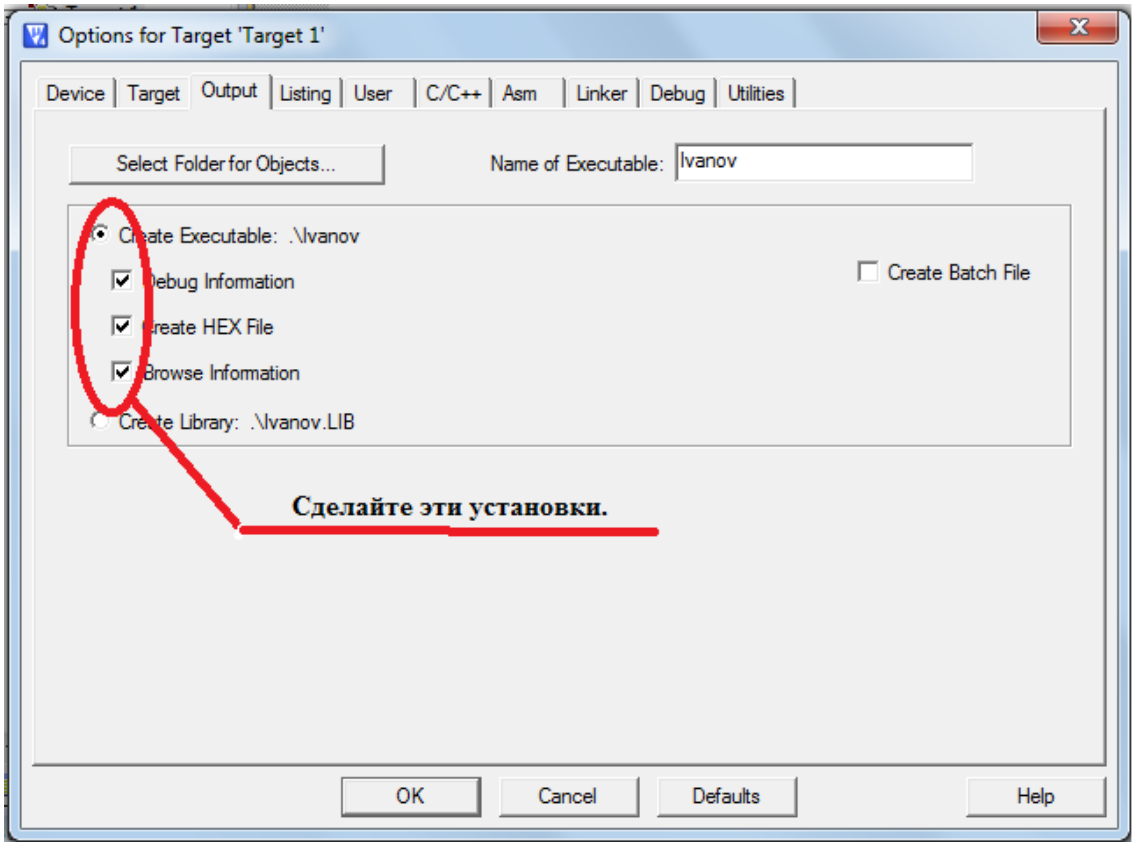


Рисунок .

Listing

Окно установки параметров вывода результатов трансляции и сборщика (Listing).

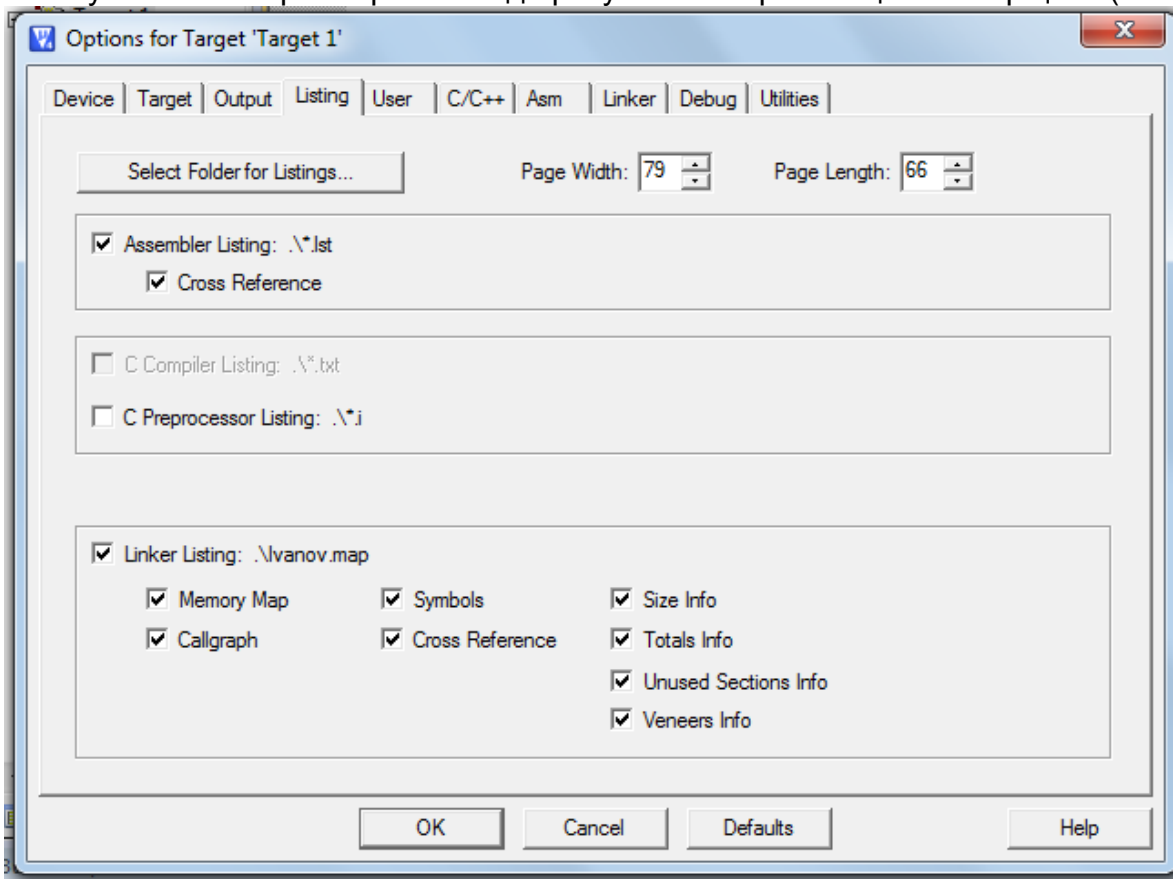


Рисунок .

User

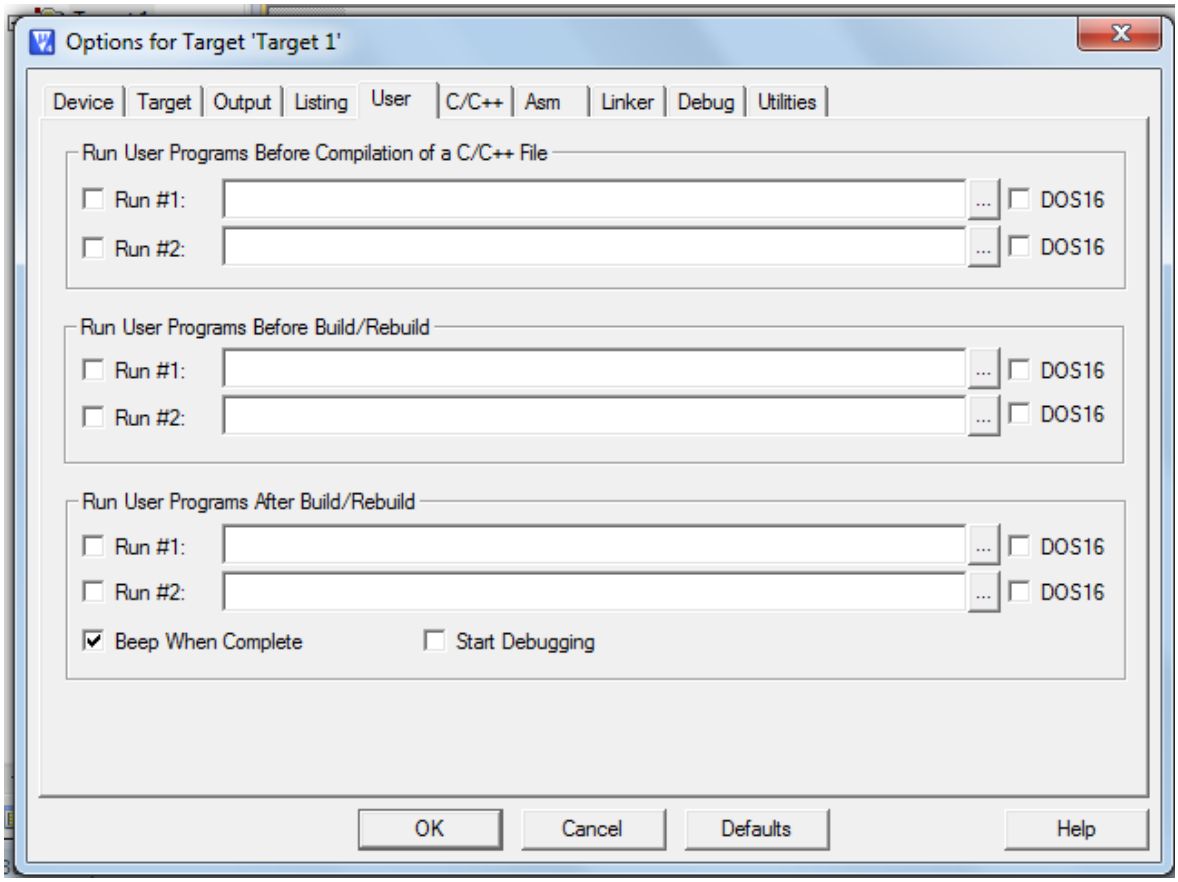


Рисунок .

C/C++

Окно установки параметров транслятора C++.

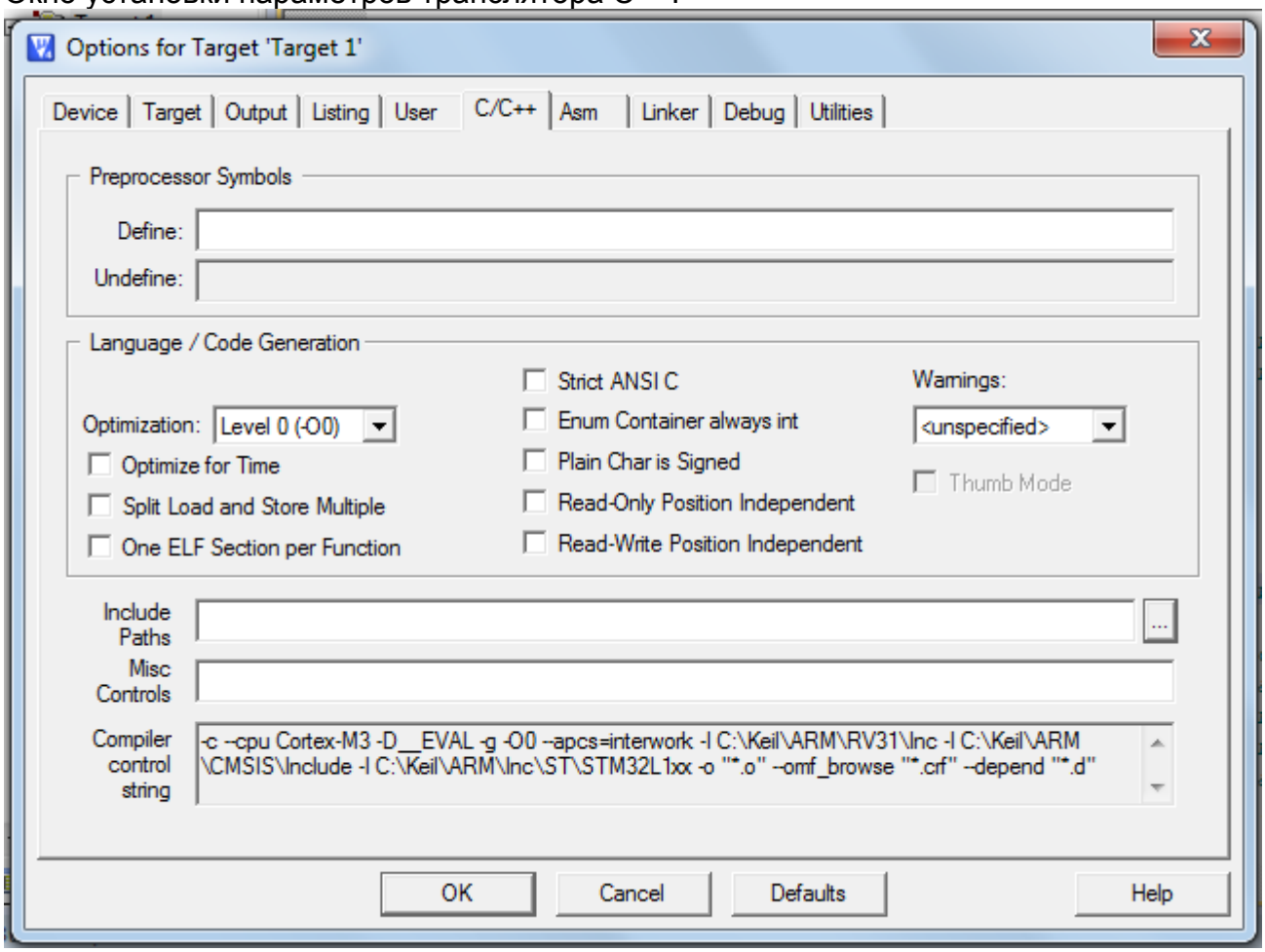


Рисунок .

Asm

Окно установки параметров транслятора с языка ассемблера.

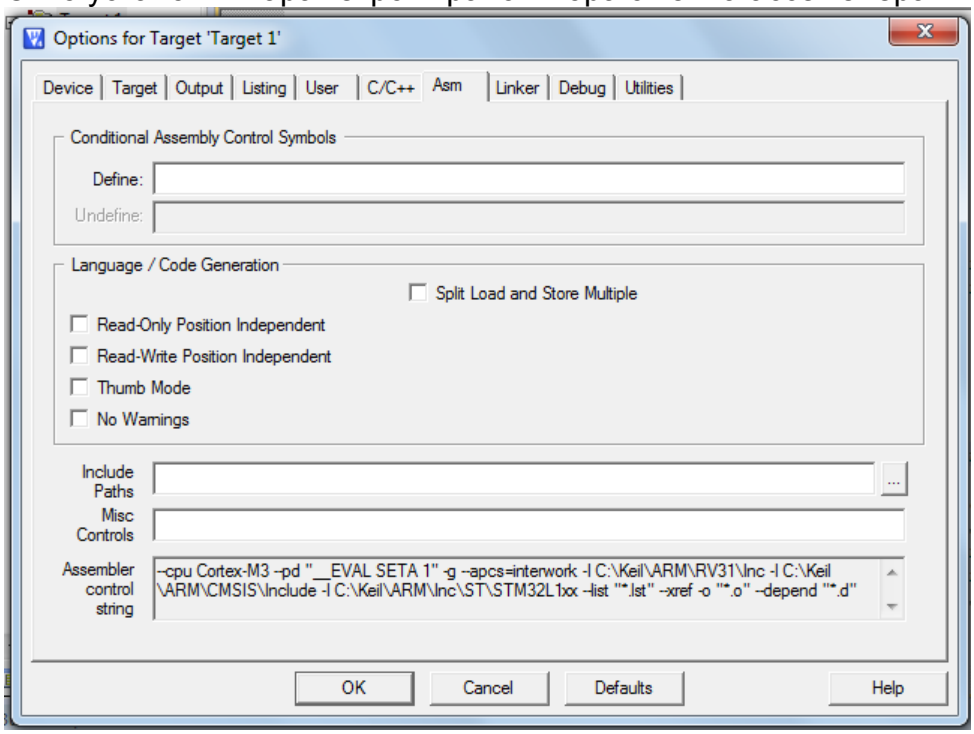


Рисунок .

Linker

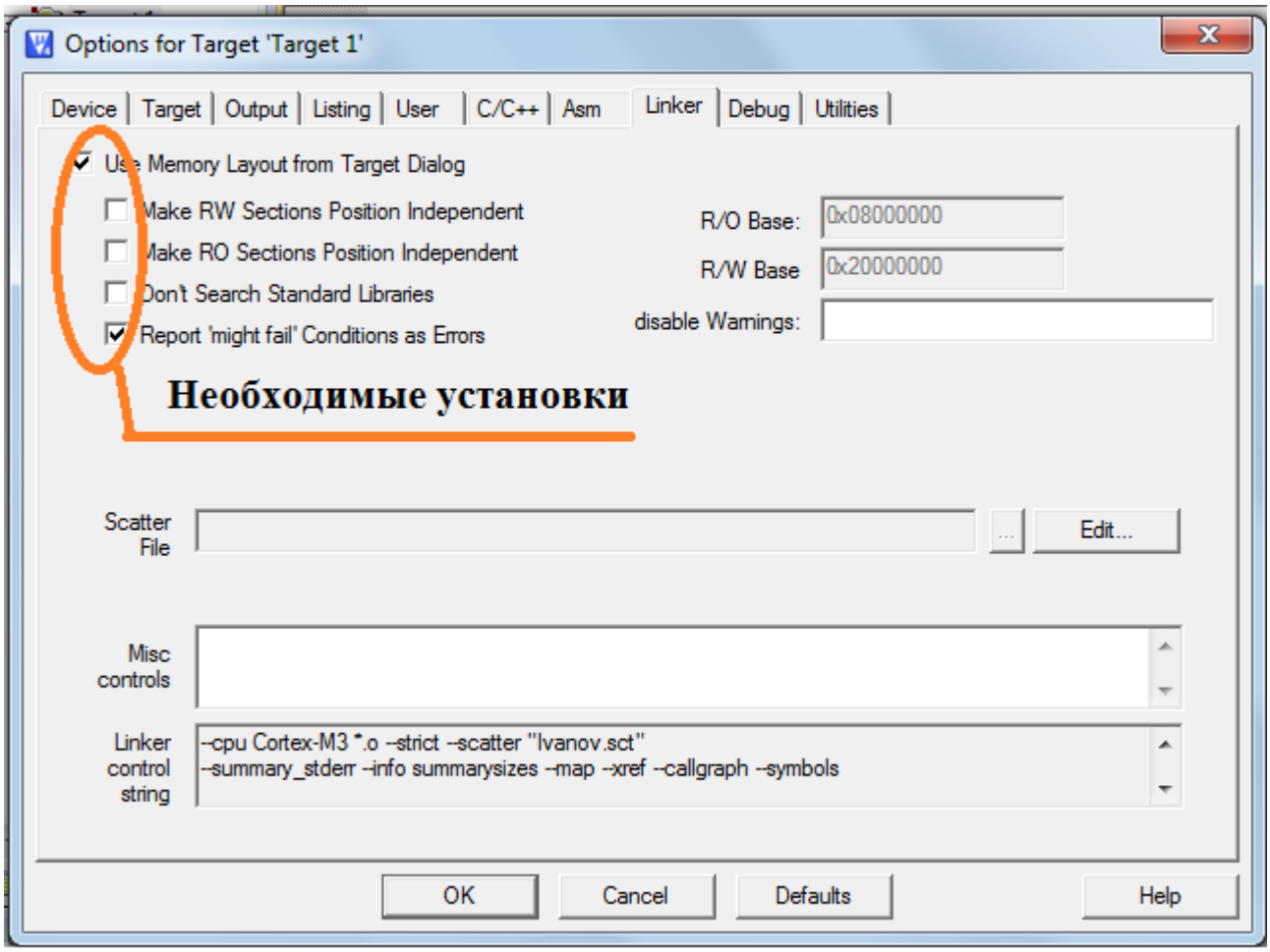


Рисунок .

Debug

В данном окне можно выбрать режимы:

Use Simulation - режим симуляции выбранного МК.

Use - отладка с помощью отладочной платы. (STM32L-DISCOVERY)

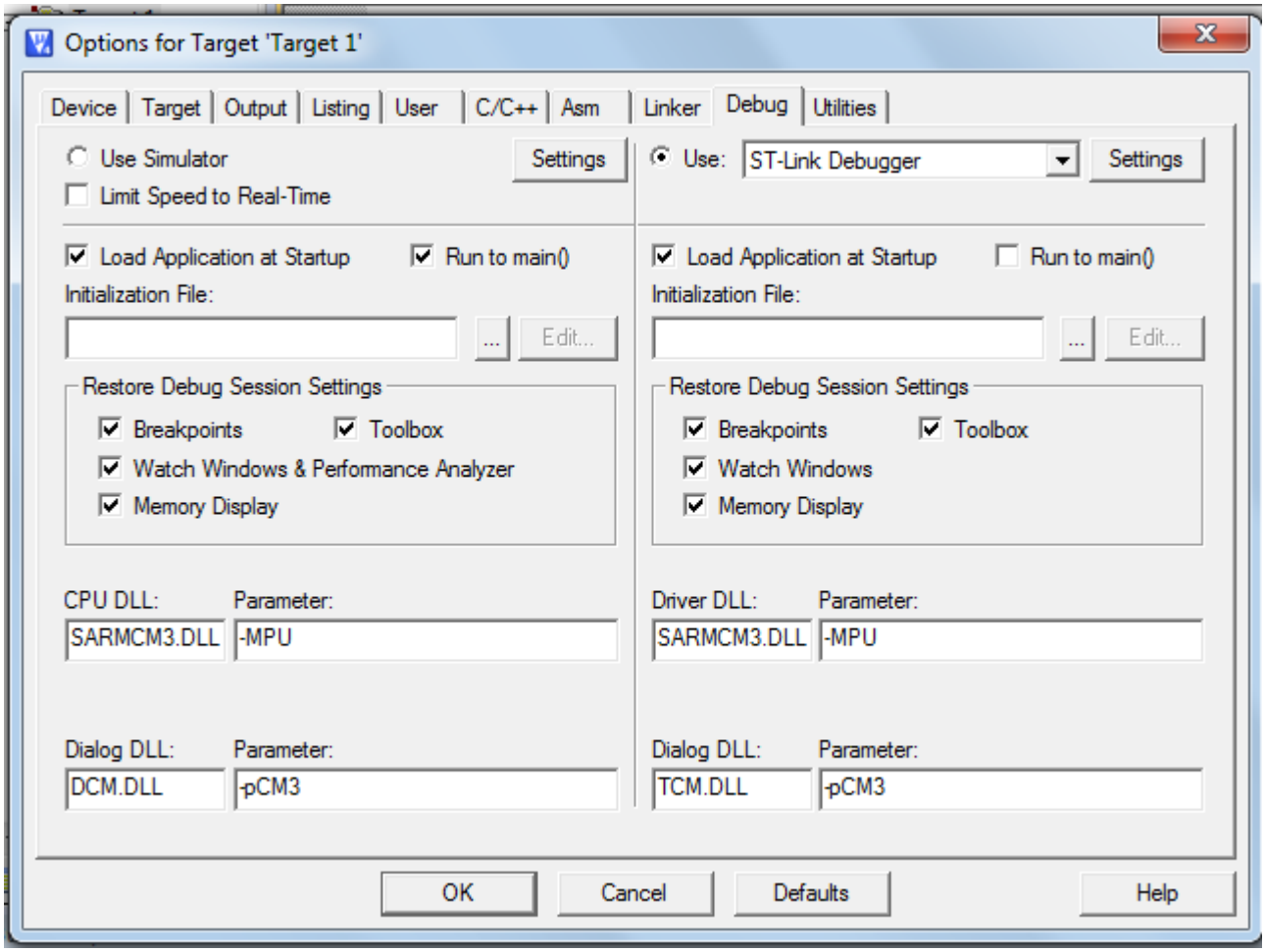


Рисунок .

Utilites

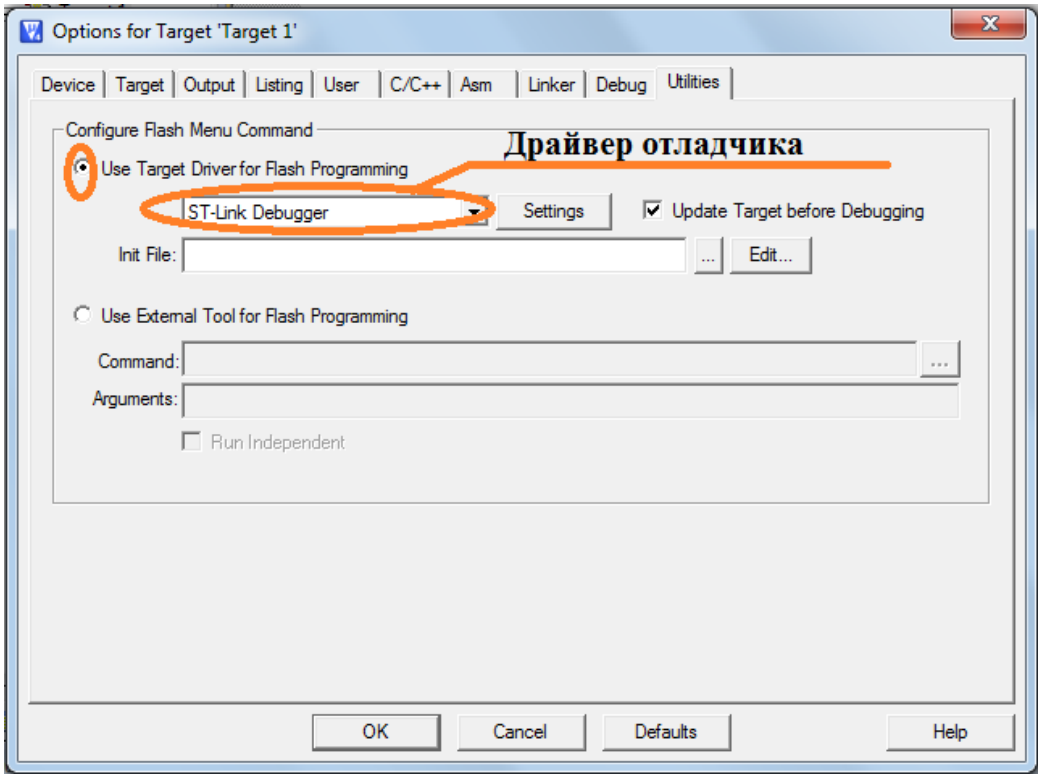


Рисунок .

В этом окне устанавливается имя драйвера, с помощью которого будет отлаживаться МК система.

Настройки выполнены. Можно приступать к написанию программы на языке ассемблер.

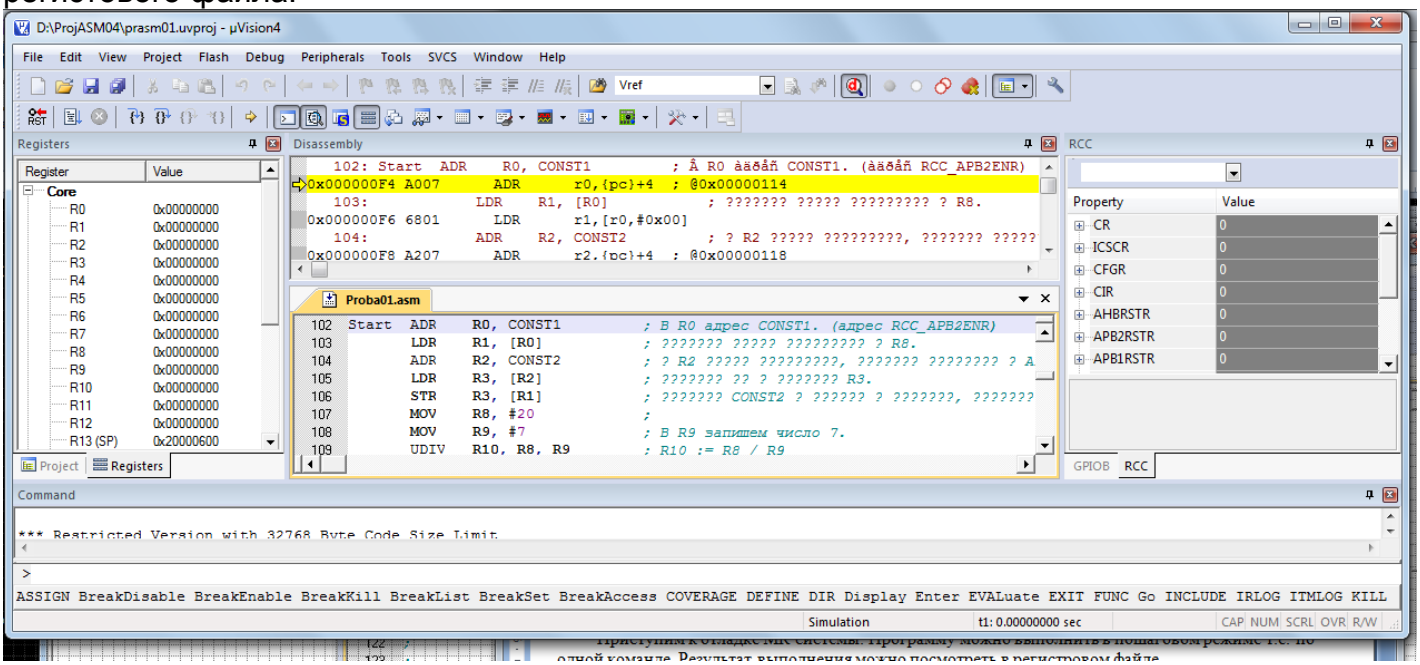
В созданный ассемблерный файл скопируем содержимое файла Shablon_lab_1.asm. Удаляем небольшой демонстрационный пример, находящийся между двумя строками комментариев. (Начало и конец программы пользователя). На освободившемся пространстве запишем нашу программу.

Программу необходимо компилировать и линковать. Это можно сделать двумя способами:

1. Найти в панели инструментов кнопку  .
2. Выполнить следующие действия: Project - Rebuild all Target files.

Если в программе имеются синтаксические ошибки, то транслятор укажет номер строки программы с ошибкой. Исправте все выявленные ошибки и повторите трансляцию.

После устранения всех ошибок запишем программу в память МК нажав кнопку  на панели инструментов, либо выполнить Debug - Start/Stop Debug Session либо Ctrk+F5. На экране появляется окно с дисассемблированной программой и окно с содержимом регистового файла.



Приступим к отладке МК системы. Программу можно выполнить в пошаговом режиме т.е. по одной команде. Результат выполнения можно посмотреть в регистровом файле.

Программу можно выполнить:

 - по одной команде,

 - по одной команде в основной программе; процедура выполняется как одна команда,

 - выполнить и остановиться,

 - выполнить до курсора.

Системы тактирования RCC (модуль Reset and clock control).

Смещение адреса этого регистра относительно базового адреса порта равно 0x18.

После сброса микроконтроллера в этом регистре устанавливаются следующие значения 0x0000 0000.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
COMP RST	Res.	DAC RST	PWR RST	Reserved				USB RST	I2C2 RST	I2C1 RST	Reserved		USART 3 RST	USART 2 RST	Res.
rw		rw	rw					rw	rw	rw			rw	rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	SPI2 RST	Reserved		WWDG RST	Res.	LCD RST	Reserved			TIM7 RST	TIM6 RST	Res.	TIM4 RST	TIM3 RST	TIM2 RST
	rw			rw		rw				rw	rw		rw	rw	rw

Порты ввода/вывода (GPIO).

Порты ввода/вывода микроконтроллера предназначены для приема данных с датчиков (температуры, давления, влажности и т.д.) и вывода преобразованных данных. Количество портов ввода/вывода (GPIO) зависит от типа микроконтроллера. Порты микроконтроллеров STM32 имеют по16 линий. кажда линия может быть сконфигурирлована как наввод (аналоговый или цифровой) так и на вывод. Для настройки линий портов на ту или иную функцию служат регистры конфигурации.

Регистры конфигурации порта.

GPIO каждый порт входа/выхода имеет 4 тридцатидвухразрядных регистра конфигурации:
1. (GPIOx_MODER,
2. GPIOx_OTYPER,
3. GPIOx_OSPEEDR,
4. GPIOx_PUPDR).

Два тридцатидвухразрядных регистра данных
5. GPIOx_ID
6. GPIOx_ODR).
7. тридцатидвухразрядный комплект/переустановленный регистр (GPIOx_BSRR),
8. трицатидвухразрядный фиксирующий регистр (GPIOx_LCKR) ,
9. трицатидвухразрядный REZERVNый регистр выбора функции 2 (GPIOx_AFRH и GPIOx_AFRL).

GPIO.

Имя регистра	Адрес	Назначение	
MODER	0x40020000	Port mode register	
OTYPER	0x40020004	Выбор схемы вывода в порт.	
OSPEEDER	0x40020008	В этом регистре указывается скорость вывода.	
PUPDR	0x4002000C	Указывается способ подключения выходного каскада порта - с подтяжкой к потенциалу "земли" или к напряжению питания.	
IDR	0x40020010	Входной регистр порта.	
ODR	0x40020014	Выходной регистр порта.	
BSRR	0x40020018	Регистр сброса/установки бит порта.	
LCKR	0x4002001C	Регистр - защелка конфигурации порта.	
AFRL	0x40020020	Указатель альтернативных функций порта (старшие разряды 16 - 31).	

AFRH'	0x40020024	Указатель альтернативных функций порта (младшие разряды 0 - 15).	

Регистр GPIOx_MODER

Смещение адреса этого регистра относительно базового адреса порта равно 0x00.

После сброса микроконтроллера в этом регистре устанавливаются следующие значения:

0xA800 0000 для порта A

0x0000 0280 для порта B

0x0000 0000 для всех остальных портов.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MODER15[1:0]		MODER14[1:0]		MODER13[1:0]		MODER12[1:0]		MODER11[1:0]		MODER10[1:0]		MODER9[1:0]		MODER8[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MODER7[1:0]		MODER6[1:0]		MODER5[1:0]		MODER4[1:0]		MODER3[1:0]		MODER2[1:0]		MODER1[1:0]		MODER0[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Биты 2y и 2y+1 регистра MORERy задают режим работы каждого вывода порта:

00 - ввод информации,

01 - вывод информации,

10 - альтернативные функции,

11 - аналоговые данные.

Регистр GPIOx_OTYPER.

Смещение адреса этого регистра относительно базового адреса порта равно 0x04.

После сброса микроконтроллера в этом регистре устанавливаются следующие значения:

0x0000 0000 для всех портов.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OT15	OT14	OT13	OT12	OT11	OT10	OT9	OT8	OT7	OT6	OT5	OT4	OT3	OT2	OT1	OT0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Биты 31 - 16 зарезервированы для будущих расширений.

Биты 15 - 0 **OTYPERy[1:0]: PORTx** задают тип вывода в ножку порта (y = 0..15)

Бит может принимать следующие значения:.

0 - push-pull выход (устанавливается при сбросе микроконтроллера)

1 - выход с открытым стоком.

Регистр GPIOx_OSPEEDER.

Смещение адреса этого регистра относительно базового адреса порта равно 0x08.

После сброса микроконтроллера в этом регистре устанавливаются следующие значения:

0x0000 00C0 для порта B,

0x0000 0000 для всех портов.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OSPEEDR15[1:0]		OSPEEDR14[1:0]		OSPEEDR13[1:0]		OSPEEDR12[1:0]		OSPEEDR11[1:0]		OSPEEDR10[1:0]		OSPEEDR9[1:0]		OSPEEDR8[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OSPEEDR7[1:0]		OSPEEDR6[1:0]		OSPEEDR5[1:0]		OSPEEDR4[1:0]		OSPEEDR3[1:0]		OSPEEDR2[1:0]		OSPEEDR1[1:0]		OSPEEDR0[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Биты 2y и 2y+1 **OSPEEDRy[1:0]: PORTx** (y = 0..15)
Эти биты задают скорость вывода для каждой ножки порта:
00 - 400 kHz,
01 - 1 MHz ,
10 - 10 MHz,
11 - 40 MHz, если емкостьCн не более 50 pF (50 MHz, если Cн менее 30 pF).

Регистр GPIOx_PUPDR.

Смещение адреса этого регистра относительно базового адреса порта равно 0x0C.

После сброса микроконтроллера в этом регистре устанавливаются следующие значения:
0x6400 0000 для порта A,
0x0000 0100 для порта B,
0x0000 0000 для всех портов.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PUPDR15[1:0]		PUPDR14[1:0]		PUPDR13[1:0]		PUPDR12[1:0]		PUPDR11[1:0]		PUPDR10[1:0]		PUPDR9[1:0]		PUPDR8[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PUPDR7[1:0]		PUPDR6[1:0]		PUPDR5[1:0]		PUPDR4[1:0]		PUPDR3[1:0]		PUPDR2[1:0]		PUPDR1[1:0]		PUPDR0[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Биты 2y и 2y+1 **PUPDRy[1:0]: POTRx** (y = 0..15)
Значения этих бит:
00 - подтягивающие резисторы не включены,
01 - включен резистор подтягивающий вывод к напряжению питания,
10 - включен резистор подтягивающий вывод к напряжению "земли",
11- резерв.

Регистр GPIOx_IDR

Регистр входных данных.

Смещение адреса этого регистра относительно базового адреса порта равно 0x10.

После сброса микроконтроллера в этом регистре устанавливается следующее значение:
0x0000 xxxx

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IDR15	IDR14	IDR13	IDR12	IDR11	IDR10	IDR9	IDR8	IDR7	IDR6	IDR5	IDR4	IDR3	IDR2	IDR1	IDR0
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Биты 0 - 15 этого регистра доступны только для чтения.
Биты 16 -31 - резерв.

Регистр GPIOx_ODR

Регистр выходных данных.

После сброса микроконтроллера в этом регистре устанавливается следующее значение:
0x0000 0000 для всех портов.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ODR15	ODR14	ODR13	ODR12	ODR11	ODR10	ODR9	ODR8	ODR7	ODR6	ODR5	ODR4	ODR3	ODR2	ODR1	ODR0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Биты 0 - 15 этого регистра доступны как для записи так и для чтения.

Биты 16 -31 - резерв.

Задание:

1. Составить программу вычисления по формуле согласно варианту из таблицы 1 Программу составить на языке «Ассемблер» для микроконтроллера ARM CORTEX M3. Вывод результата произвести в указанный в задании порт. Числовые данные переменных a и b разместить в команде, остальные в памяти программ.
2. Отладить программу, используя симулятор Keil µVision в режиме Use Simulation и с использованием отладочной платы STM32L-DISCOVERY.
3. Выполнить составленную программу в пошаговом режиме. Дать объяснение результатам выполнения каждой арифметической команды и состоянию регистра .
4. Вывести результат расчета по формуле в указанный порт. Выведенный результат перевести из шестнадцатеричной системы счисления в десятичную.
5. Составить отчет.
6. Защитить работу.

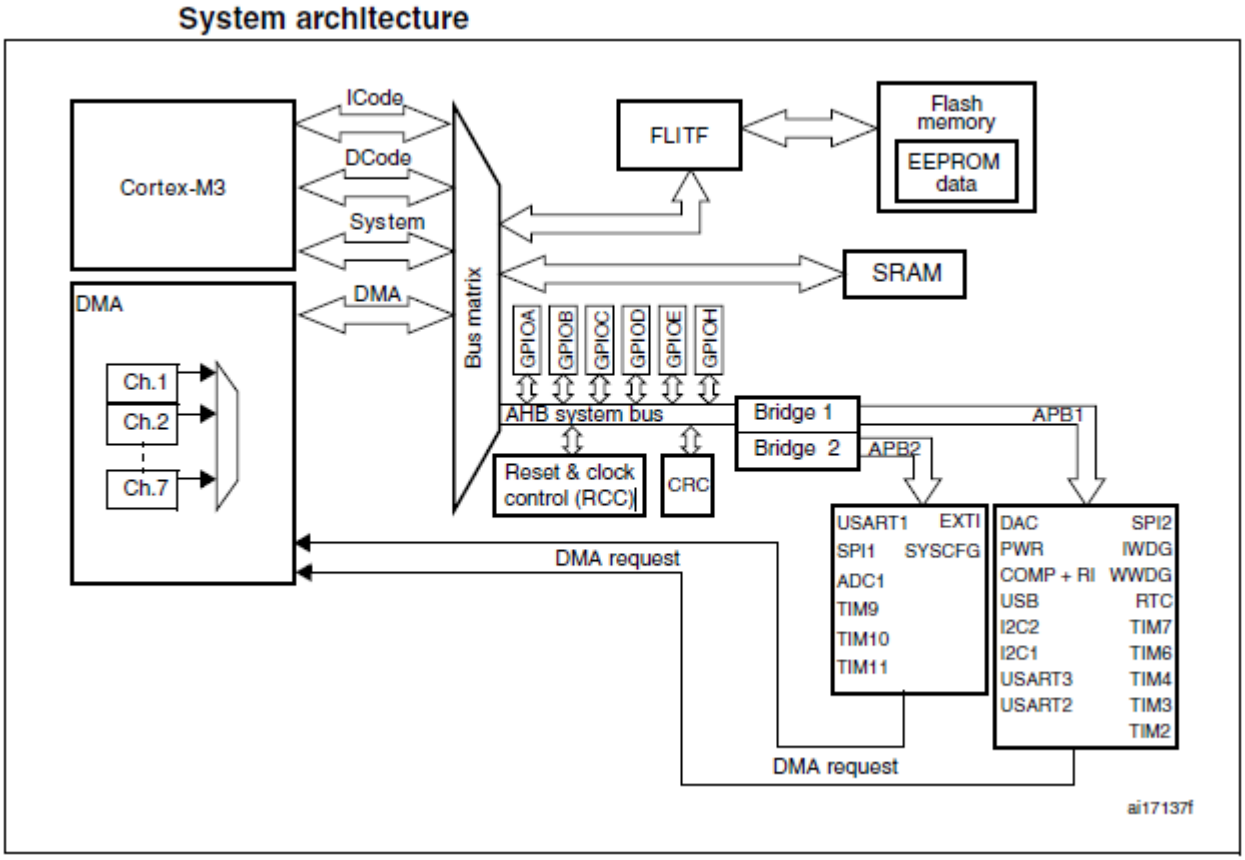
Таблица 1.

№вар.	Формула	a	b	c	d	e	f	Порт вывода
1	X=a*b/c+(a-b)/(e-f)							PORTA
2	X=(d-e+a*b)-c/(a+b)							PORTB
3	X=a-{(c-d)/(a+b)+e}/f							PORTC
4	X=a+b-c+(a+b*c)/(a+b)							PORTA
5	X=c-d+(a-b*c)/(c-d)/(a+f)							PORTB
6	X=a/b+f-(c+d)/(a*b)							PORTC
7	X=(c+a*b)*(d+a)-a/(d+a)							PORTA
8	X=d+e-(f-a/b)*{f/(d+e)}							PORTB
9	X=(a+b)*(c-d/e)+d/(a+b)							PORTC
10	X=c**2+(a+b*c)/f+a*b/(a+b)							PORTA
11	X=a+b+c+d+e*b/(c+d)-10/f							PORTB
12	X=20/f+(a+b*c)/(a-b)*e							PORTC
13	X=f/a*b+(a+b*c-2)/e							PORTA
14	X=a-b+c+(a/b+d*e)*a							PORTB
15	X=a/d+(a/b/c+d**2)-a-d							PORTA
16	X=a*b*c-(d+e/f)*(a+b)							PORTB
17	X=(a-b)*(b+c)-a*d/e/(b+c)							PORTC
18	X=f-a+(c*d+d/a)+(a*b-c)							PORTA
19	X=2*a+b/c-(a*b-c/d**2)							PORTB
20	X=a/2+b*4-(a/b+c*d)/(a+b)							PORTC
21	X=35/f+(a+b)*(a*c-b**2)							PORTA
22	X=a+b+c+d*(a-b)/(c*d)							PORTB
23	X=f+d-c+b/(f+d)+(c*d-f)/2							PORTC
24	X=(c/a-b*a)+(a+b)/4-f*e							PORTA
25	X=a/b*(a+b)-e/f+(a/b-f*(a+b))							PORTB
26	X=f*b/c/(a+b)+(c-d)*(a+b)							PORTC
27	X=e/a+b/e+(a+b*c)*(b/e-f)							PORTA
28								PORTB

Все числовые значения таблицы 1 заданы в десятичной системе счисления.

Приложение №1

Структурная схема микроконтроллера ARM Cortex M3.



Приложение №2

Внутренние регистры МК ARM.

Внутренние регистры МКARM можно классифицировать следующим образом:

1. Регистровый файл ЦПУ или регистры общего назначения (РОН).
2. Регистр статуса ЦПУ - xPSR.
3. Регистры управления периферийными устройствами.

ЦПУ с ядром ARM выполнено по RISC архитектуре. Перед выполнением арифметической или логической операции операнды необходимо поместить в регистровый файл. Регистровый файл состоит из шестнадцати 32-битных регистров. Регистры R0 - R12 используются для хранения значений переменных. Регистр R13 - указатель вершины стека. R14 - регистр связи, используется для хранения адреса возврата из процедуры. И, наконец регистр R15 - является программным счетчиком.

Регистровый файл ЦПУ

Приложение №3

Система команд Thumb-2 микроконтроллеров ARM Cortex M3.

Арифметические операции

ADD	ADD R0, R1, Operand2 ADD R0, R1, 12bit const.	Сложение R0 = R1 + R2	N, Z, C, V	S
ADC	ADC R0, R1, R2 ADC R0, R1, 8bit const	Сложение с учётом переноса R0 = R1 + R2 + C	N, Z, C, V	S
ADDW	ADD R0, R1, 12bit const.	Сложение с 12bit константой	N, Z, C, V	
SUB	SUB R0, R1, R2 SUB R0, R1, 12bit const.	Вычитание R0 = R1 - R2	N, Z, C, V	S
SBC	SBC R0, R1, R2 SBC R0, R1, 8bit const.	Вычитание с учётом переноса R0 = R1 - R2 - C	N, Z, C, V	S
SUBW	SUB R0, R1, 12bit const.	Вычитание с 12bit константой	N, Z, C, V	
RSB	RSB R0, R1, R2 RSB R0, R1, 8bit const. RSB R0, R1, R2, ASR#23	Вычитание с противоположным порядком аргументов. R0 = R2 - R1 или R0 = 8bit const - R1	N, Z, C, V	S

Умножение. Деление.

MUL	R0 = R1 x R2	Умножение. 32bit результат		
MLA	R0 = (R1 x R2) + R3	Умножение и сложение	N, Z	S
MLS	R0 = R3 - (R1 x R2)	Умножение и вычитание		
UMULL	RLo, RHi = R2 x R3	Умножение. 64bit результат		
UMLAL	RLo, RHi += R2 x R3	Умножение и сложение		
SMULL	RLo, RHi = R2 x R3	Умножение со знаком. 64bit результат		
SMLAL	RLo, RHi += R2 x R3	Умножение со знаком и сложение		
UDIV	R0 = R1 / R2	Деление без знака		
SDIV	R0 = R1 / R2	Деление со знаком		

Доступ к памяти

ADR	ADR R0, label +/- 12bit ADR.W R0, label +/- 32bit	Загрузка адреса метки в регистр.
LDR STR	LDR R0, [R1, #8bit const.]! - прединкремент LDR R0, [R1], #8bit const. - постинкремент LDRB R0, [R1] - без инкремента STRB R0, [R1], #1	Загрузка/сохранение регистра в режиме адресации со смещением. B=байт, SB=байт со знаком (только загрузка) H=полуслово, SH=полуслово со знаком (только загрузка)
LDR STR	LDR R0, [R1, R2, {LSL#0..3}] STR R0, [R1, R2, {LSL#0..3}]	Смещение задаётся третьим регистром. B, SB, H, SH работают аналогично
LDR LDRD	LDR R0, label LDRD R0, R1, label	Смещение задаётся адресом метки. B, SB, H, SH работают аналогично Невозможно применить к STR/STRD.
LDRT STRT		Непривилегированный доступ. Аналогичны простому LDR/STR.
LDRD STRD	LDRD R0, R1, [R2, #10bit const.]! - прединкремент LDRD R0, R1, [R2], #10bit const. - постинкремент LDRD R0, R1, [R2] - без инкремента STRD R0, R1, [R2]	Загрузка/сохранение двух регистров в режиме адресации со смещением. Константа должна быть кратна 4.
LDM STM	LDM R0, {R1-R3} LDM R0!, {R1-R3} - постинкремент R0 IA, DB, FD, EA - см. описание	Загрузка/сохранение множества регистров. IA - с увеличением адреса DB - с уменьшением адреса.
PUSH POP	PUSH {R0, R2-R7, R12} POP {R0, R2-R7, R12}	Загрузка/чтение из стека
LDREX	LDREX R1, [R2, #10bit const.]	Эксклюзивное чтение/запись регистра.
STREX	STREX R0, R1, [R2, #10bit const.]	B=байт, H=полуслово.
CLREX	CLREX (без параметров)	Сброс признака эксклюзивного доступа.

Перемещение и обработка данных

MOV MVN	MOV R0, R1 MOV R0, 16bit const. MVN R0, R1	Загрузка/перемещение регистров Загрузка 16-бит в мл. полуслово, сброс старшего Перемещение регистра с инверсией	N, Z, C, V	S
MOVT	MOVT R0, 16bit const.	Загр. 16bit в старшее полуслово. Младш. остаётся без изменений.		
CMP	CMP R0, R1	Сравнение	N, Z, C, V	S
CMN	CMN R0, R1	Сравнение с противоположным знаком		
TST	TST R0, Operand2	Проверить значение битов по маске	N, Z, C	S
TEQ	TEQ R0, Operand2	Проверить равенство двух величин	N, Z	S
REV REV16 REVSH RBIT	REV R0, R1	Изменение порядка битов или байтов в слове		
CLZ	CLZ R0, R1	Подсчет количества ведущих нулей		

Логические операции

AND	И	0 в маске сбрасывает биты. Второй Operand2 для всех команд		
ORR	ИЛИ	1 в маске устанавливает биты		
EOR	Исключающее ИЛИ	1 в маске инвертирует биты	N, Z, C	S
BIC	Сброс битов по маске. И-НЕ	1 в маске сбрасывает биты		
ORN	ИЛИ-НЕ	0 в маске устанавливает биты		

Операции сдвига

ASR	Арифметический сдвиг вправо	N, Z, C	S
LSL	Логический сдвиг влево		
LSR	Логический сдвиг вправо		
ROR	Циклический сдвиг вправо		
RRX	Сдвиг вправо на 1 позицию через перенос		

Ветвление. Подпрограммы.

B	B label	Переход к метке
BX	BX R0	Переход по адресу в регистре
BL	BL label	Выполнить подпрограмму label
BLX	BLX R0	Выполнить подпрограмму по адресу в регистре
BX LR	BX LR	Возврат из подпрограммы
CBZ	CBZ R0, label	Переход, если R0 = 0
CBNZ	CBNZ R0, label	Переход, если R0 != 0
TBB	TBB [R0, R1]	Табличный переход по индексу. Короткий переход.
TBH	TBH [R0, R1, LSL #1]	Длиннй переход

IT	IT{x{y{z}}} cond	Блок условно исполняемых инструкций

Работа с битовыми полями

BFC	BFC R0, #lsb, #width	Сброс поля в ноль
BFI	BFI R0, R1, #lsb, #width	Копирует младшие биты R1 в поле R0
UBFX	UBFX R0, R1, #lsb, #width	Копирует поле R1 в мл. биты R0 с заполнением нулями
SBFX	SBFX R0, R1, #lsb, #width	Копирует поле R1 в мл. биты R0 с расширением знака
UXTB	UXTB R0, R1 {, ROR #8,16,24}	Преобразование байта с заполнением нулями
UXTH		Преобразование полуслова
SXTB	SXTB R0, R1 {, ROR #8,16,24}	Преобразование байта с расширением знака
SXTH		Преобразование полуслова

Преобразование данных с насыщением

SSAT	SSAT Rd, #n, Rm {, shift #s}	Число со знаком в число со знаком,	Q
USAT	USAT Rd, #n, Rm {, shift #s}	Число со знаком в число без знака	

Управление системой

MRS	MRS R0, PSR	Чтение/запись специальных регистров.
MSR	MSR PSR, R0	
CPSIE		Разрешение/запрет прерываний
CPSID		
WFE	WFE	Ожидать событие
WFI	WFI	Ожидать прерывание
BKPT		Точка останова
DMB	DMB	Барьер синхронизации доступа к ОЗУ
DSB		
ISB	ISB	Сброс конвейера
SEV	SEV	Дёрнуть ножкой события для внешних камней
SVC	SVC 8bit const.	Вызов системного сервиса
NOP	NOP	

Суффиксы условного исполнения

EQ	Z = 1	Равенство
NE	Z = 0	Неравенство
CS, HS	C = 1	Больше или равно, беззнаковое сравнение
CC, LO	C = 0	Меньше, беззнаковое сравнение
MI	N = 1	Отрицательное значение, меньше нуля
PL	N = 0	Положительное значение, больше или равно нулю
VS	V = 1	Переполнение
VC	V = 0	Нет переполнения
HI	C = 1 и Z=0	Больше, беззнаковое сравнение
LS	C = 0 или Z=1	Меньше или равно, беззнаковое сравнение
GE	N = V	Больше или равно, знаковое сравнение
LT	N != V	Меньше, знаковое сравнение
GT	Z = 0 и N = V	Больше, знаковое сравнение
LE	Z = 1 и N != V	Меньше или равно, знаковое сравнение
AL	1	Безусловное исполнение

Встроенные псевдоинструкции компиллятора

MOV32	MOV32 R0, 32bit const. (label)	Загрузить слово в регистр.
LDR	LDR R0, =label	Загрузить 32bit адрес метки или #число
ADRL	ADRL R0, label	Загрузить 32bit адрес метки

Поле Условия

В режиме ARM все команды выполняются в зависимости состоянию регистра CPSR и поля условия самой команды. Это поле (биты 31:28) содержит условие, при которых команда будет выполнена. Если флаги C, N, Z и V установлены (сброшены) согласно коду поля условия, то команда будет выполнена, в противном случае эта команда будет проигнорирована.

Всего существуют 16 различных условий, каждое из которых определяется двухсимвольным суффиксом, добавляемым к мнемонике команды. Например, команда переход (Branch - "B" в ассемблере) становится командой перейти, если равно - BEQ (Branch if Equal), которая означает, что переход будет выполнен, если установлен флаг Z.

Практически можно использовать только 15 различных условий, 16-е условие (1111) зарезервировано и не применяется.

При отсутствии в мнемонике команды условия выполнения или установлен суффикс AL, то эта команда будет выполнена безусловно, независимо от состояния флагов условия (регистр CPSR).

Перечень кодов условий выполнения

Код	Суффикс	Флаги	Значение
0000	EQ	Z установлен	Равно
0001	NE	Z сброшен	Не равно
0010	CS	C установлен	Выше или равно
0011	CC	C сброшен	Ниже
0100	MI	N установлен	Отрицательный результат
0101	PL	N сброшен	Положительный результат либо ноль
0110	VS	V установлен	Переполнение
0111	VC	V сброшен	Нет переполнения
1000	HI	C установлен and Z сброшен	Выше
1001	LS	C сброшен or Z установлен	Ниже или равно
1010	GE	N равно V	Больше или равно
1011	LT	N не равно V	Меньше
1100	GT	Z сброшен AND (N равно V)	Больше
1101	LE	Z установлен OR (N не равно V)	Меньше или равно
1110	AL	(проигнорировано)	Всегда

Приложение №4
Директивы ассемблера.
Директива AREA.

Директива AREA позволят создать программу, состоящую из нескольких сегментов: сегменты кода программы, сегменты данных.

Синтаксис:

AREA sectionname{,attr}{,attr}...

sectionname - имя сегмента. Имя сегмента может быть любым. Если имя начинается не с буквы, то такое имя записывается между вертикальными линиями.

attr - задает атрибуты сегмента. Если атрибутов несколько, то они разделяются запятой.

CODE - этот атрибут указывает на то, что за директивой AREA, где он расположен следуют сегмент содержащий команды МК.

DATA - описывает сегмент данных.

READONLY - сегмент только для чтения.

READWRITE - сегмент области памяти, предназначенной как для чтения так и для записи.

ALIGN = выражение - этот атрибут задает выравнивание сегмента. Выражением может быть целое число n в интервале от 0 до 31. Выравнивание осуществляется по границе 2^n. Если данный атрибут не задан, выравнивание осуществляется на границе 2 байт.

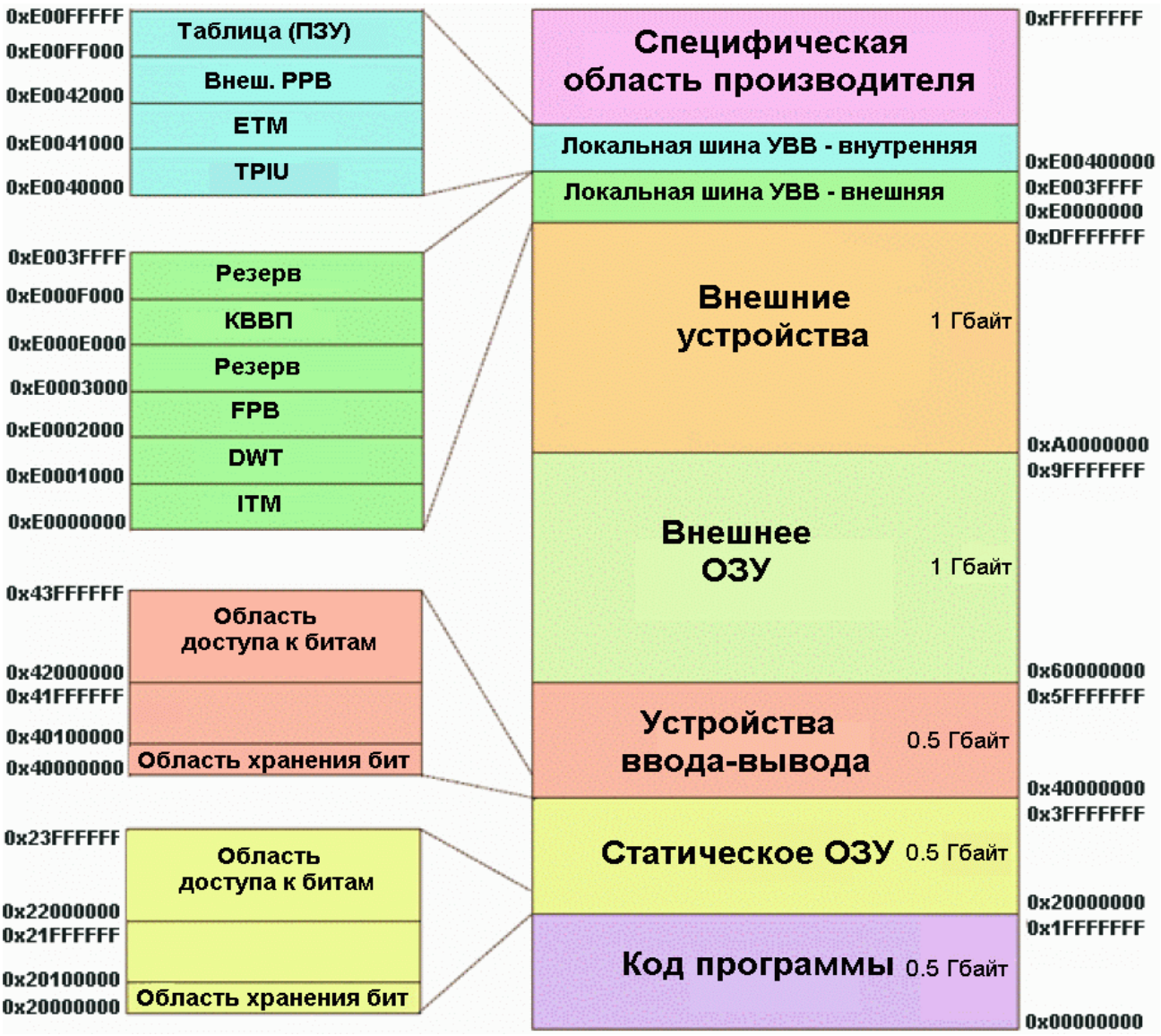
Примечание

Не используйте ALIGN = 0 или ALIGN = 1 для сегмента кода с командами ARM.
Не используйте ALIGN = 0 для сегмента кода с командами THUMB.

COMMON - это общий сегмент данных. В данном сегменте должны содержаться оды команд или данные. При сборке программы компоновщиком (Link-ром) данная область памяти заполняется нулями. Все общие области с одинаковыми именами будут находится в одной и той же области памяти.

Приложение № 5.

Карта памяти МК ARM Cortex M3

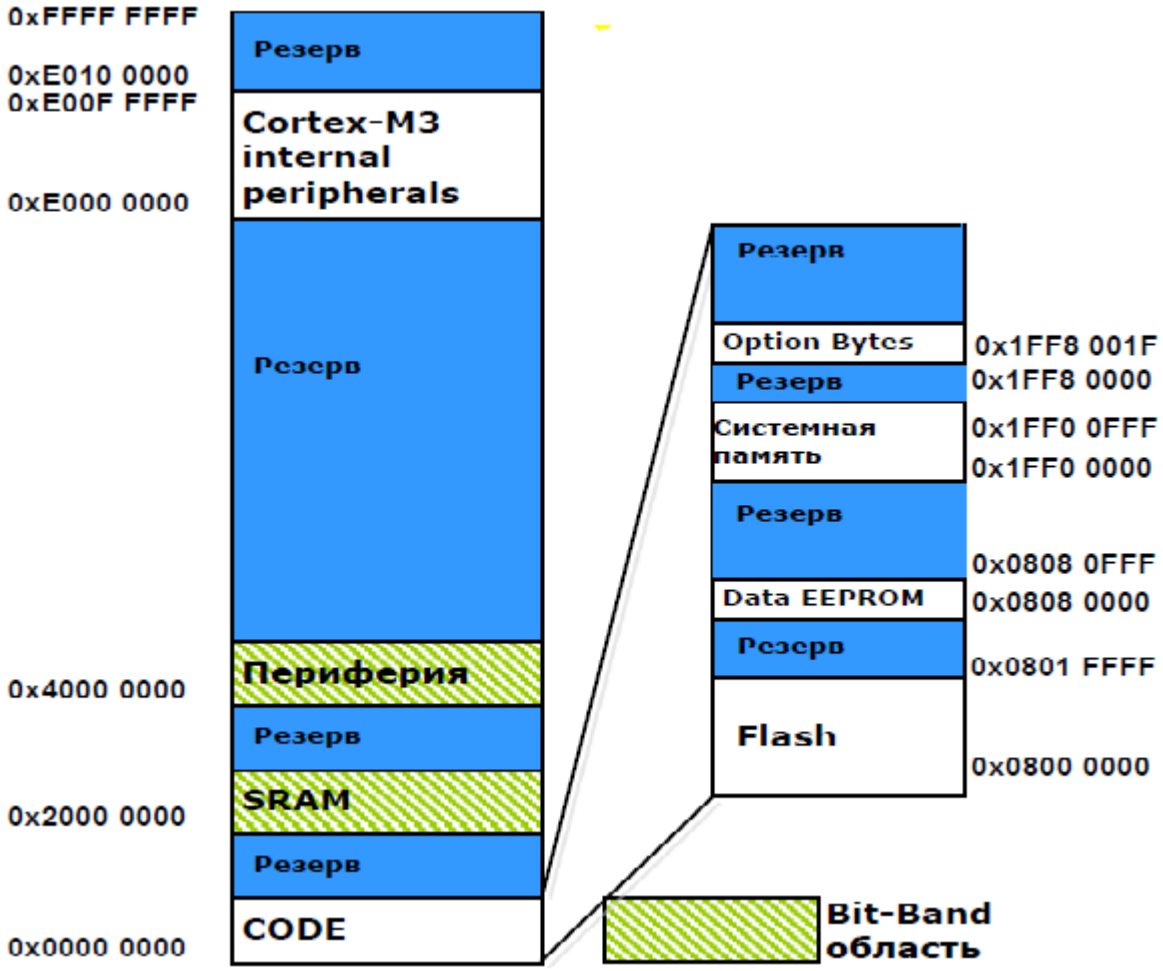


Распределение памяти МК STM32L15x.

Объем адресного пространства - 4 Гб
RAM : до16 Кб
FLASH : до128 Кб
Data EEPROM: до 4 Кб
Режимы загрузки.

В зависимости от конфигурации загрузки, встроенная Flash –память, системная память или встроенная SRAM –память доступна с адреса @0x00(remapping). Системная память и встроенная SRAM –память может также доступной с @0x00 адреса программно, установкой специальных бит.

		Режим загрузки	Замечания
BOOT1	**BOOT0**		
X	0	User Flash	Встроенная Flash. Загрузочная область.
0	1	System Memory	Системная память. Загрузочная область.
1	1	Embedded SRAM	Встроенная SRAM. Загрузочная область.



Приложение № 6.

Системы тактирования (модуль Reset and clock control (RCC)).
Регистры RCC.

Приложение № 7.

Литература.

1. П.П. Редькин. 32/16-битные микроконтроллеры ARM7 семейства AT91SAM7 фирмы ATMEL. Руководство пользователя. Москва, издательский дом «Додека XXI». 2008.
2. Б. Пахомов. C/C++ и MS Visual C++ 2008 для начинающих. Санкт-Петербург, «БХВ-Петербург» 2009.
3. Joseph Yiu. The Defi nitive Guide to the ARM Cortex-M3.

4. RM0038. Reference manual. STM32L151xx and STM32L152xx advanced ARM-based 32-bit MCUs.