Министерство образования и науки
Российской Федерации

Федеральное государственное бюджетное образовательное
учреждение высшего образования
ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ
УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)

Кафедра конструирования узлов и деталей радиоэлектронной аппаратуры
(КУДР)

А.А. Бомбизов, Е.И. Тренкаль

АССЕМБЛЕР. ВНЕШНИЕ ПРЕРЫВАНИЯ

Методические указания к выполнению
лабораторной и самостоятельной работы
по дисциплине «Микропроцессорные устройства»

Томск 2017

1 Введение

В предыдущей работе были освоены принципы условных и безусловных переходов, на основе которых был организован рабочий программный цикл с периодическим опросом состояния кнопки. Но при специально введенной задержке наблюдалось запаздывание опроса состояния кнопки, что могло привести к пропуску внешнего события.

Целью настоящей работы является освоение принципов работы контроллера внешних прерываний, с использованием которого программа может с малыми задержками получать информацию о внешних событиях независимо от выполняемого в текущий момент алгоритма.

2 Краткая теория

Прерывание (англ. interrupt) – сигнал от программного или аппаратного обеспечения, сообщающий процессору о наступлении какого-либо события, требующего немедленного внимания. Процессор отвечает приостановкой своей текущей активности, сохраняя свое состояние и выполняя функцию, называемую обработчиком прерывания (или программой обработки прерывания), который реагирует на событие и обслуживает его, после чего возвращает управление в прерванный код.

В МК Cortex-M есть два понятия:

- *Event* – это событие (аппаратное или программное), на которое могут реагировать ядро или периферийные блоки. Одним из вариантов реакции может быть прерывание.

- *Interrupt* – это прерывание работы программы с передачей управления в специализированный участок – обработчик прерывания. Взаимосвязь между *Event* и *Interrupt* заключается в следующем: каждый *Interrupt* вызывается *Event*, но не каждый *Event* вызывает *Interrupt*. Помимо прерываний, события могут активировать и другие возможности МК.

Управление и обработка прерываниями производится контроллером приоритетных векторных прерываний NVIC *(Nested Vectored Interrupt Controller)*. Контроллер прерываний является частью ядра Cortex-M.

При возникновении некоторого события контроллер прерываний автоматически прерывает выполнение основной программы и вызывает соответствующую функцию обработки прерываний. После выхода из функции обработчика прерываний программа продолжает выполнение с того места, где произошло прерывание. Все происходит автоматически. Контроллер NVIC поддерживает вложенность прерываний и приоритеты. Каждому прерыванию при настройке NVIC присваивается свой приоритет. Если во время обработки низкоприоритетного прерывания возникает

высокоприоритетное, то оно, в свою очередь, прервет обработчик низкоприоритетного прерывания.

Вход в прерывание и выход из него. При инициации прерывания NVIC переключает ядро в режим обработки прерывания. После перехода в режим обработки прерывания регистры ядра помещаются в стек. В стек перемещаются регистр статуса программы (Program Status Register (PSR)), счетчик программы (Program Counter (PC)) и регистр связи (Link Register (LR)). Благодаря этому запоминается состояние, в котором находилось ядро перед переходом в режим обработки прерываний. Автоматическое перемещение регистров в стек делает возможным их использования в функции обработки прерывания.

Непосредственно во время записи значения регистров в стек осуществляется выборка начального адреса функции обработки прерывания.

По завершении обработки прерывания все действия выполнятся в обратном порядке: извлекается содержимое стека и, параллельно с этим, осуществляется выборка адреса возврата.

С момента инициации прерывания до выполнения первой команды обработчика прерываний проходит 12 тактов, такое же время необходимо для возобновления основной программы после завершения обработки прерывания.

Выход из режима прерывания осуществляется путём сброса флага прерывания в устройстве, которое его генерировало. После этого в регистры общего назначения будут возвращены значения из стека.

Все возможные прерывания, поддерживаемые NVIC, записываются в таблицу векторов прерываний. По сути своей, таблица векторов прерываний это список адресов функций обработчиков прерываний. Номер в списке соответствует номеру прерывания.

NVIC поддерживает до 240 различных векторов прерываний. Но реализация уже зависит от конкретного производителя. В описании ядра стандартизованы только прерывания исключений ядра [1, см. раздел 2.3.2 Exception types]: Reset, NMI, HardFault, MemManage, BusFault, UsageFault, SVCall, PendSV, SysTick. Помимо этого, каждый производитель может внедрить дополнительные обязательные прерывания, связанные, например, с отладкой, программированием и др. Остальные прерывания уникальны для каждого микроконтроллера. Описание таблицы векторов для микроконтроллера STM32F429 представлены в [2, Table 62. Vector table for STM32F42xxx and STM32F43xxx]. В столбце Position указаны номера прерываний. Позиции без номера указывают обязательные прерывания ядра.

Расположение векторов прерываний и загрузка МК. Процесс загрузки контроллера начинается с адреса 0x0800000. Из начала флеш памяти ядро считывает значение SP (stack top address – вершина стека) и PC (reset routine location – адрес на начало программы (Start+1)). После двух обязательных компонентов запуска может размещаться дальнейшая таблица векторов прерываний в том же порядке, как указано в [2, Table 62. Vector table for STM32F42xxx and STM32F43xxx]. При этом все адреса, указанные в столбце Address, должны сохраниться.

.word вершина_стека
.word Start+1
.word 0 @ если нужно пропустить
.word 0 @ если нужно пропустить
.word 0 @ если нужно пропустить
…. @пропускается пока не будет достигнут нужный адрес
.word метка+1 @ устанавливается адрес обработчика требуемого прерывания

Прерывания EXTI. Кнопка, которую в рамках данной работы необходимо обработать через прерывание, подключена к порту ввода-вывода, за прерывания от которого отвечает контроллер внешних прерываний EXTI (External interrupt/event controller) [2, раздел 12.2]. EXTI может одновременно обрабатывать 23 источника прерываний, из которых первые 16 (EXTI0–EXTI15) мультиплексируют события только от портов ввода вывода (Рисунок 1).

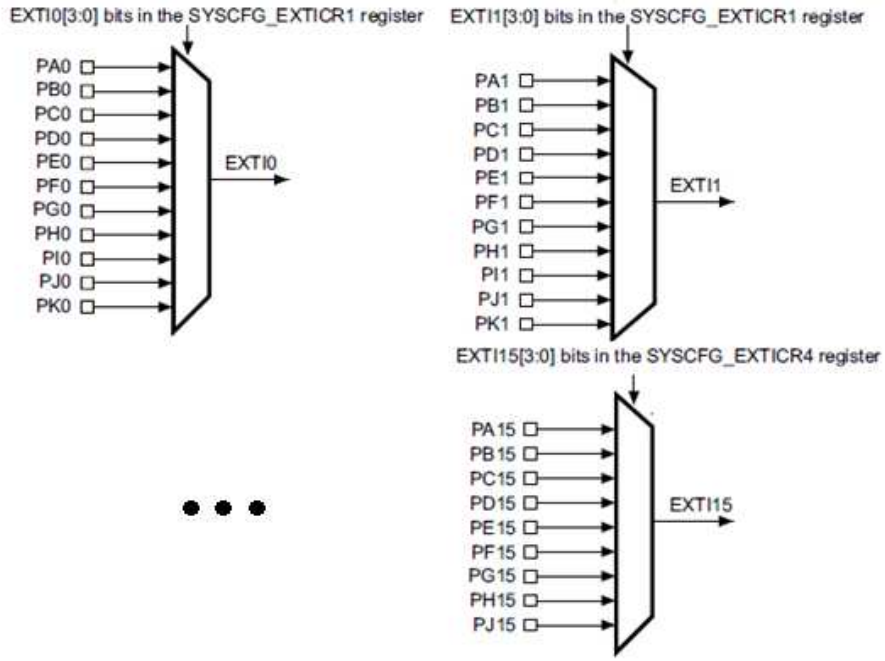
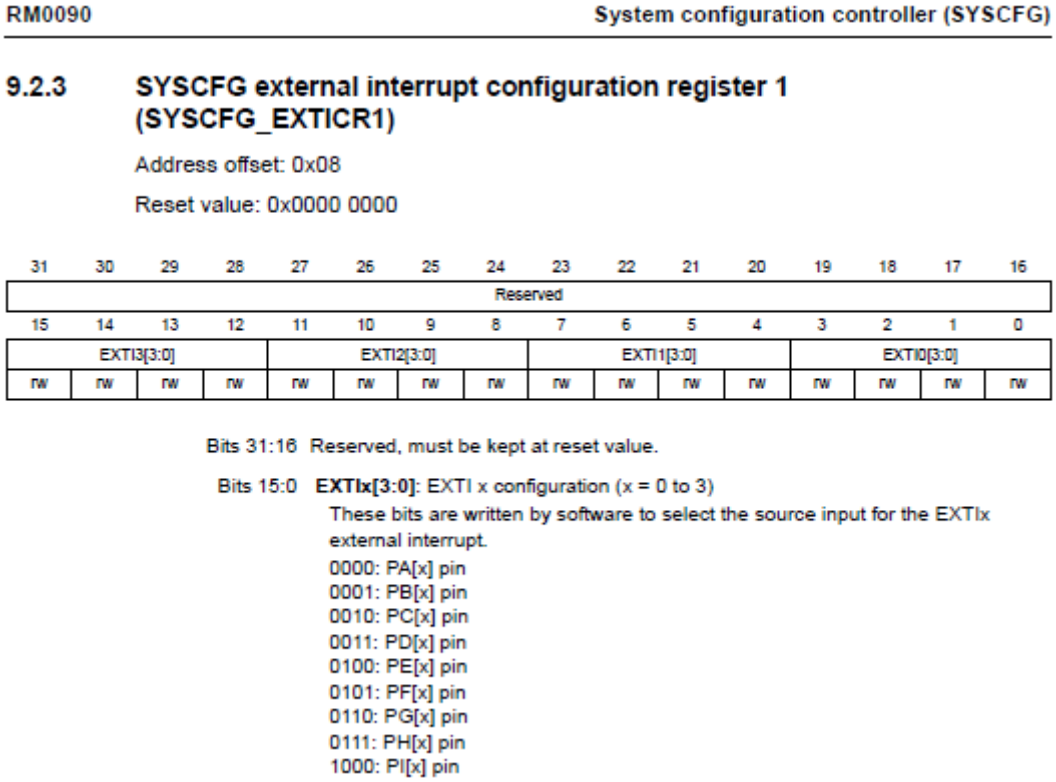


Рисунок 1 – External interrupt/event GPIO mapping (STM32F42xxx and STM32F43xxx) [2]

Исходя из рисунка, можно сделать следующие выводы:

1) внешние прерывания можно настроить только от входов портов ввода-вывода, имеющих различные номера, то есть нельзя, например, настроить одновременно работающее внешнее прерывание на вывод PA0 и PB0;

2) за переключение мультиплексора отвечает группа регистров SYSCFG_EXTICR1 – SYSCFG_EXTICR4, относящаяся к периферийному устройству SYSCFG [2, раздел 9], в которых каждая четверка битов отвечает за свой мультиплексор (Рисунок 2).



RM0090 System configuration controller (SYSCFG)

9.2.3 SYSCFG external interrupt configuration register 1 (SYSCFG_EXTICR1)

Address offset: 0x08

Reset value: 0x0000 0000

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
EXTI3[3:0]				EXTI2[3:0]				EXTI1[3:0]				EXTI0[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **EXTIx[3:0]**: EXTI x configuration (x = 0 to 3)
These bits are written by software to select the source input for the EXTIx external interrupt.
0000: PA[x] pin
0001: PB[x] pin
0010: PC[x] pin
0011: PD[x] pin
0100: PE[x] pin
0101: PF[x] pin
0110: PG[x] pin
0111: PH[x] pin
1000: PI[x] pin

x – номер мультиплексора и номер вывода порта ввода вывода.

Рисунок 2 – Регистр настройки мультиплексора [2]

За настройку EXTI отвечают четыре регистра:

EXTI_IMR – регистр маскировки прерываний. Запись ‘1’ в соответствующий бит разрешает прерывания от необходимого источника (EXTI0–EXTI22), по умолчанию там нули.

EXTI_EMR – регистр маскировки событий, запись ‘1’ в соответствующий бит разрешает событие, по умолчанию там нули. В данной работе этот регистр применяться не будет, так как будет необходима обработка только прерываний.

EXTI_FTSR и **EXTI_RTSR** – определяют, по какому фронту будет возникать прерывание. При записи ‘1’ в соответствующий бит EXTI_FTSR – по спадающему фронту (F – fail), в EXTI_RTSR – по возрастающему (R – rise). То есть, если сигнал изменяется с логической ‘1’ на ‘0’, будет

происходить событие по спадающему фронту, и чтобы регистрировать именно его, необходимо настроить EXTI_FTSR, а EXTI_RTSR оставить сброшенным. Если необходимо обрабатывать событие изменения из ‘0’ в ‘1’, то работа будет вестись с регистром EXTI_RTSR.

Помимо этого, существует регистр состояния **EXTI_PR** – флаг возникновения события/прерывания, сбрасывается вручную при записи ‘1’. Этот регистр отвечает за выход из режима прерывания.

После настройки устройства, генерирующего прерывание, необходимо выполнить его активацию в контроллере приоритетных векторных прерываний NVIC. Это осуществляется путём установки в единицу в группе регистров NVIC_ISER0 – NVIC_ISER2 [3, Table 44. NVIC register summary] бита, соответствующего номеру устройства [2, Table 62. Vector table for STM32F42xxx and STM32F43xxx, столбец Position], генерирующего прерывание. NVIC_ISER0 отвечает за позиции от 0 до 31. NVIC_ISER1 – от 32 до 63. NVIC_ISER2 – от 64 до 95 (хотя в STM32F429 из только 90).

3 Порядок выполнения работы

В ходе данной работы будет создана программа мигания одного светодиода с задержкой. Причем переключение моргающего светодиода будет происходить по нажатию кнопки, которая будет генерировать прерывание. Задача будет решаться в четыре этапа: 1) создание базового проекта; 2) создание таблицы обработчиков прерываний; 3) инициализация прерывания от порта ввода-вывода; 4) создание обработчика прерывания от кнопки.

Создание базового проекта.

3.1 Изучите предложенный в п. 2 теоретический материал.

3.2 Для начала работы скопируйте папку template_Прерывания.base в свой рабочий каталог.

3.3 Откройте в блокноте (или в программе Notepad++) файл main.s, содержащийся в каталоге template_Прерывания.base. Как видно, это проект, созданный на третьем занятии за исключением функции опроса состояния кнопки.

Инициализация прерывания от порта ввода-вывода.

3.4 Определите шину для тактования SYSCFG.

3.5 Включите тактование SYSCFG.

3.6 Определите, какой мультиплексор SYSCFG необходимо подключить для обработки прерывания от кнопки. Номер этого мультиплексора будет являться номером канала (EXTI line[x]) прерывания.

3.7 Переключите в регистре SYSCFG_EXTICRx необходимый мультиплексор таким образом, чтобы он коммутировал сигналы от кнопки с каналом прерывания.

3.8 Определите по [5] какое событие должно возникать при нажатии на кнопку. Нарастание или спад фронта?

3.9 Сконфигурируйте регистр EXTI_RTSR (EXTI_FTSR – для спада фронта) для обработки прерывания от выбранного канала (определяется номером мультиплексора).

3.10 Активируйте канал контроллера прерываний EXTI путём установки в регистре EXTI_IMR соответствующего бита.

3.11 Определите по [2, Table 62. Vector table for STM32F42xxx and STM32F43xxx, столбец Position] номер обработчика прерывания EXTI, соответствующего порту ввода-вывода, к которому подключена кнопка.

3.12 Активируйте прерывание в контроллере NVIC путём установки в группе регистров NVIC_ISER0 – NVIC_ISER2 в логическую единицу бита, соответствующего номеру обработчика прерывания.

Создание таблицы обработчиков прерываний.

3.13 Определите по [2, Table 62. Vector table for STM32F42xxx and STM32F43xxx, столбец Position] адрес обработчика прерывания EXTI, соответствующего порту ввода-вывода, к которому подключена кнопка. Это потребуется для формирования таблицы векторов прерываний.

3.14 В main.s после указания метки начала программы заполните нулями все ячейки во флеш-памяти до ячейки, соответствующей прерыванию EXTI. Для заполнения нулями рекомендуется использовать макросы .rept .endr или .space [4].

3.15 После заполнения нулями укажите адрес (int_vect_EXTI+1) обработчика прерываний (см. теоретическую часть). int_vect_EXTI – название метки обработчика прерываний. Может быть названо по усмотрению разработчика.

Создание обработчика прерывания от кнопки.

3.16 Определите функцию int_vect_EXTI.

3.17 Когда функция обработки будет вызвана микроконтроллером, внутри неё необходимо выйти из режима прерывания. Для этого нужно выполнить сброс соответствующего бита в регистре состояния EXTI_PR (см. теоретическую часть).

3.18 После выхода из режима переключите биты в регистре R7 по маске R6, по аналогии с sub_TESTBTN из предыдущей работы.

3.19 Откомпилируйте программу путём запуска командного файла make_project.bat и выполните программирование микроконтроллера. В

результате красный светодиод должен гореть постоянно, а зеленый моргать с установленной задержкой. По нажатию на кнопку начнет моргать красный, а зеленый перестанет. При этом переключение активного светодиода должно происходить практически мгновенно.

3.20 Оформите отчет, содержащий титульный лист, введение, ход выполнения работы, ответы на контрольные вопросы и выводы.

3.21 Защитите отчет у преподавателя.

4 Контрольные вопросы

4.1 Опишите последовательность определения таблицы векторов прерываний.

4.2 Какую основную функцию выполняет NVIC?

4.3 Какую основную функцию выполняет EXTI?

4.4 Как выйти из функции обработки прерывания?

4.5 Нажатие на кнопку генерирует событие нарастания или спада фронта?

Список литературы

1. Cortex-M4 Generic User Guide.– URL: http://infocenter.arm.com/help/topic/com.arm.doc.dui0553a/DUI0553A_cortex_m4_dgug.pdf.

2. RM0090. Reference manual. STM32F405/415, STM32F407/417, STM32F427/437 and STM32F429/439 advanced ARM®-based 32-bit MCUs.– URL: www.st.com/resource/en/reference_manual/DM00031020.pdf (дата обращения: 10.01.2017).

3. PM0214. Programming manual. STM32F3, STM32F4 and STM32L4 Series. Cortex®-M4 programming manual.– URL: www.st.com/resource/en/programming_manual/DM00046982.pdf (дата обращения: 10.01.2017).

4. GNU ARM Assembler Quick Reference.– URL://http://www.ic.unicamp.br/~celio/mc404-2014/docs/gnu-arm-directives.pdf.

5. MB1075. STM32F429I-DISCO schematics.– URL: http://www.st.com/resource/en/schematic_pack/stm32f429i-disco_sch.zip.