**Работа №5**

**"Обработка прерываний в контроллерах 1986ВЕ9x. Внешние прерывания EXT\_INT. Системный таймер"**

**Цель работы:** получить навыки работы с контроллером обработки прерываний NVIC.

**Задачи:**

- закрепить на практике процесс обработки прерываний в микропроцессорной технике;

- получить навыки инициализации параметров контроллера прерываний, а также разработки программ для микроконтроллера с использованием системы прерываний;

- изучить особенности контроллера прерываний NVIC;

- изучить функции библиотеки CMSIS по обработке прерываний;

- изучить системный таймер контроллера;

**-** получить навыки программирования системного таймера.

**Используемое оборудование:**

1. Отладочная плата с микроконтроллером Миландр MDR1986BE92QI/MDR1986BE93У.

2. Комплект Программатора

2.1 программатор JLINK (USB-JTAG);

2.2 Кабель USB 2.0 A – B;

2.3 Шина (20 проводников).

3. Источник питания 5В (или дополнительный USB кабель)

4. Комплект Логического анализатора

4.1 Логический анализатор;

4.2. Кабель USB 2.0A – mini-B;

4.3 Шина 10 проводников.

5. Кнопочный модуль

6. Провода для штыревых выводов (мама-мама) 3 штуки.

**Используемая документация:**

1. Техническое описание на ядро Миландр MDR1986BE9x (файл 1\_Тех\_описание\_ядро\_1986ВЕ9X.pdf или по ссылке с сайта разработчика [https://ic.milandr.ru/upload/iblock/a33/ioaf9ygfmq1lbxfhd5aad0mukg3dc93s/1986%D0%92%D0%959X.pdf](https://ic.milandr.ru/upload/iblock/a33/ioaf9ygfmq1lbxfhd5aad0mukg3dc93s/1986ВЕ9X.pdf))

2. Выводы отладочной платы микроконтроллера 1986BE92QI (файл 2\_Выводы\_платы\_1986BE92QI.pdf или по ссылке\* с сайта разработчика <https://ic.milandr.ru/upload/iblock/8f6/8f67b8b736b3ec94edbbeb4777a9c4db.zip>)

3.Выводы отладочной платы микроконтроллера 1986BE93 (файл 2\_Выводы\_платы\_1986BE93У.pdf или по ссылке\* с сайта разработчика <https://ic.milandr.ru/upload/iblock/782/782c4c3b486d6f8d92995e9a44a94401.zip>)

При загрузке схемотехнической документации с сайта Milandr (информация о выводах платы) загружается zip архив, который содержит 1 или 2 файла со схемой размещения выводов микроконтроллера на плате и несколько схемотехнических файлов для разводки и печати платы.

**Исключения и прерывания**

Исключение – это особенное событие, требующее обработки от системы.

Прерывание – это процесс остановки выполнения исполняемого программного кода при возникновении некоторого события для дальнейшей обработки этого события.

Алгоритм обработки прерываний:

1. при возникновении исключения, по которому разрешено прерывание, выполнение исполняемого кода приостанавливается, сохраняется адрес следующей на выполнение команды;
2. переход к таблице векторов прерываний по адресу возникшего события;
3. переход по вектору прерывания к процедуре обработки;
4. выполнение процедуры обработчика прерывания;
5. возврат к выполнению исполняемого кода.

Если контроллер прерываний не сохраняет контекст выполняемой задачи, то это должно быть прописано кодом в обработчике прерываний.

**Виды исключений**

Исключения делятся на 2 группы:

 внутренние (синхронные), которые возникают внутри процессора во время работы;

 внешние (асинхронные), которые формируются во внешних по отношению к процессору устройствах.

**Внутренние исключения MDR1986ВЕ9x**

**Reset** вызывается при включении питания и сбросе. Модель исключений трактует Reset как специальную форму исключения.

**Hard fault** возникает при ошибке в обработке исключений или потому что исключение не может быть обработано каким-либо другим механизмом. Hard fault имеет фиксированный приоритет -1.

**Memory message fault** возникает при обращении к защищенной памяти. Блок MPU (Memory Protection Unit – блок аппаратной защиты памяти) или фиксированные защитные настройки определяют это исключение как для данных, так и для инструкций.

**Bus fault** возникает при ошибке памяти при выполнении выборки инструкций или обращения к данным, например, обращение в несуществующую память.

**Usage fault** возникает при сбоях при выполнении инструкции. Например:

 выполнение неизвестной инструкции;

 обращение к некорректно выровненным данным (например, невыровненный адрес при обращении за полусловами halfword и словами word);

 некорректное состояние при выполнении инструкции (например, деление на 0);

 ошибка при возвращении из обработчика.

**SVCall** (Supervisor Call) возникает при выполнении инструкции супервизора в приложениях с использованием операционных систем. Инструкция SVC может использоваться для доступа к функциям ОС и драйверам устройств.

**PendSV –** запросы сервисов системного уровня. В приложениях с использованием ОС PendSV используется для переключения контекстов, когда нет других активных исключений.

**Systick** генерируется системным таймером, когда он обнуляется. Программное обеспечение также может генерировать исключение SysTick. В приложениях с использованием ОС процессор может использовать это исключение для подсчета системных циклов.

**Внешние исключения MDR1986ВЕ9x**

**IRQn –** это исключения, вызываемые периферийными устройствами (таймерами, аналогово-цифровыми и цифро-аналоговыми преобразователями, контроллерами последовательных интерфейсов, и т.п.), асинхронные по отношению к выполняемым инструкциям. IRQn прерывания перечислены в таблице 5.1.

Таблица 5.1 – внешние прерывания контроллеров семейства **MDR1986ВЕ9x.**

|  |  |  |
| --- | --- | --- |
| **Прерывание** | **Блок** | **Описание** |
| IRQ0 | CAN1 | Сигнал прерывания от блока CAN1 |
| IRQ1 | CAN2 | Сигнал прерывания от блока CAN2 |
| IRQ2 | USB | Прерывания от USB Host или USB Slave |
| IRQ3...IRQ4 | Резерв |  |
| IRQ5 | DMA | Прерывания контроллера прямого доступа к памяти |
| IRQ6 | UART1 | Сигнал прерывания от блока UART1 |
| IRQ7 | UART2 | Сигнал прерывания от блока UART2 |
| IRQ8 | SSP1 | Сигнал прерывания от блока SSP1 (SPI) |
| IRQ9 | Резерв |  |
| IRQ10 | I2C | Сигнал прерывания от блока I2C |
| IRQ11 | POWER | Сигнал прерывания от POWER Detector |
| IRQ12 | WWDG | Сигнал прерывания от WWDG |
| IRQ13 | Резерв |  |
| IRQ14 | Timer 1 | Сигнал прерывания от Таймера 1 |
| IRQ15 | Timer 2 | Сигнал прерывания от Таймера 2 |
| IRQ16 | Timer 3 | Сигнал прерывания от Таймера 3 |
| IRQ17 | ADC | Сигналы прерываний от АЦП. |
| IRQ18 | Резерв |  |
| IRQ19 | COMP | Сигнал прерывания от блока компаратора |
| IRQ20 | SSP2 | Сигнал прерывания от блока SSP2 (SPI) |
| IRQ21... IRQ26 | Резерв |  |
| IRQ27 | BACKUP | Прерывание от BKP и часов реального времени |
| IRQ28 | Внешнее прер-е 1 | Сигнал EXT\_INT1. Прерывание по уровню:  0 - нет прерывания 1 - есть прерывание |
| IRQ29 | Внешнее прер-е 2 | Сигнал EXT\_INT2. Прерывание по уровню:  0 - нет прерывания 1 - есть прерывание |
| IRQ30 | Внешнее прер-е 3 | Сигнал EXT\_INT3. Прерывание по уровню:  0 - нет прерывания 1 - есть прерывание |
| IRQ31 | Внешнее прер-е 4 | Сигнал EXT\_INT4. Прерывание по уровню:  0 - нет прерывания 1 - есть прерывание |

**Состояния исключений**

**Pending** – исключение находится в состоянии ожидания обработки процессором. Запрос прерывания от периферийных блоков или программы может изменить состояние соответствующего прерывания на состояние Pending.

**Active** – исключение начало обрабатываться процессором, но еще не закончено. Обработчик исключения может быть прерван другим обработчиком исключения. В этом случае оба исключения находятся в состоянии Active.

**Inactive** – исключение не находится в стадии Active или Pending.

**Active** и **Pending** – исключение начало обрабатываться процессором, но появилось новое исключение в состоянии pending от того же источника.

**Контроллер NVIC**

**NVIC (Nested Vectored Interrupt Controller)** **–** контроллер вложенных векторизированных прерываний. Контроллер обеспечивает следующие возможности:

 программное задание уровня приоритета в диапазоне от 0 до 7 независимо каждому прерыванию. Более высокое значение уровня соответствует меньшему приоритету, таким образом, уровень 0 отвечает наивысшему приоритету прерывания;

 срабатывание сигнала прерывания по импульсу и по уровню;

 динамическое изменение приоритета прерываний;

 разделение исключений по группам с одинаковым приоритетом и по подгруппам внутри одной группы;

 передача управления из одного обработчика исключения в другой без восстановления контекста;

 автоматическое сохранение в стеке состояния процессора (контекста) по входу в обработчик прерывания и восстановление его по завершению обработчика без необходимости непосредственного программирования этих операций, что обеспечивает обработку исключительных ситуаций с малой задержкой.

Управление контроллером осуществляется достаточно большим количеством регистров. Описание регистров контроллера NVIC представлено в таблице 5.2.

Таблица 5.2 – Регистры контроллера NVIC (Обобщенная модель)

|  |  |  |  |
| --- | --- | --- | --- |
| Смещение | Название | Доступ | Описание |
| 0x000 | ISER[0] | RW | Регистр разрешения прерываний  1 – разрешение прерывания  0 – не влияет |
| … |  |
| 0x01C | ISER[7] |
| … |  |  |  |
| 0x080 | ICER[0] | RW | Регистр запрета прерываний  1 – запрет прерывания  0 – не влияет |
| … |  |
| 0x09C | ICER[7] |
| … |  |  |  |
| 0x100 | ISPR[0] | RW | Регистр установки состояния ожидания для прерываний  1 – перевод в «ожидание»  0 – не влияет |
| … |  |
| 0x11C | ISPR[7] |
| … |  |  |  |
| 0x180 | ICPR[0] | RW | Регистр сброса состояния ожидания для прерывания  1 – сброс «ожидания»  0 – не влияет |
| … |  |
| 0x19C | ICPR[7] |
| … |  |  |  |
| 0x200 | IABR[0] | RO | Регистр активных прерываний |
| … |  |
| 0x21C | IABR[7] |
| … |  |  |  |
| 0x300 | IP[3], IP[2], IP[1], IP[0] | RW | Регистр приоритета прерываний  Для задания приоритета в элементе массива, соответствующем номеру прерывания, в старшие 3 бита (из 8) заносится значение, соответствующее требуемому уровню приоритета (от 0 до 7) |
| … |  |
| 0x3F0 | IP[239],  IP[238],  IP[237],  IP[236] |
| … |  |  |  |
| 0xE00 | STIR | WO | Регистр программного формирования прерываний  Для вызова прерывания необходимо записать в регистр его номер, т.е. для IRQ3 – 0x00000003 |

\*RO – read only, WO – write only, RW – read/write

В таблице указана реакция контроллера при записи значений в его регистры. При чтении: ISER и ICER – 1 разрешено, 0 запрещено, ISPR и ICPR – 1 в «ожидании», 0 не ожидает обслуживания, IABR – 1 в данный момент прерывание обрабатывается, 0 в данный момент не обрабатывается.

В целях повышения эффективности разработки программного обеспечения в CMSIS предусмотрен упрощенный доступ к регистрам контроллера прерываний NVIC из среды разработки программного обеспечения. Регистры разрешения, запрета, установки и сброса состояния ожидания прерываний, а также регистр активных прерываний отображаются на массивы 32-разрядных целых чисел, а именно:

- массив ISER[0] соответствует регистру ISER0;

- массив ICER[0] соответствует регистру ICER0;

- массив ISPR[0] соответствует регистру ISPR0;

- массив ICPR[0] соответствует регистру ICPR0;

- массив IABR[0] соответствует регистру IABR0;

В свою очередь 3-битные поля регистра приоритета прерываний отображаются на массив 3-разрядных целых чисел - массив IP[0]...IP[29], который соответствует регистрам IPR0-IPR7, причем элемент массива IP[n] соответствует приоритету прерывания с номером n.

Каждый бит регистров ISER отвечает за разрешение одного прерывания из таблицы 1. Например, чтобы разрешить прерывание от счётчика Timer1, необходимо записать единицу в 14-й бит (по номеру прерывания) регистра ISER0. Из таблицы видно, что в контроллере реализованы прерывания с номерами [0:31] (причем часть этого множества не используется). Поэтому общий набор регистров ISER[7]:ISER[0] для данного контроллера избыточен и используется только регистр ISER[0] (аналогично ICER[0], ISPR[0], ICPR[0], IABR[0]). Массив регистров IP состоит из 8-битных элементов, и каждый элемент отведён на одно прерывание (при этом из каждых 8-ми бит задействованы только 3).

**Приоритеты прерываний**

Прерываниям может быть присвоен приоритет – значение от 0 до 7. Более высокое значение уровня соответствует меньшему приоритету. Если имеется несколько исключений с одинаковым приоритетом, то больший приоритет имеет исключение с меньшим порядковым номером. Если процессор выполняет обработчик исключения и возникает исключение с большим приоритетом, то происходит переход к обработчику исключения с большим приоритетом. Если при выполнении обработчика произошло исключение с таким же приоритетом, то это исключение будет выполнено по завершению текущего обработчика, несмотря на порядковый номер исключения.

**Работа с контроллером NVIC с использованием CMSIS**

Для упрощения процесса программирования прерываний в библиотеке CMSIS описан набор функций, некоторые из которых представлены в таблице 5.3.

Таблица 5.3 – Функции обработки прерываний библиотек CMSIS

|  |  |
| --- | --- |
| **Подпрограмма** | **Описание** |
| void \_\_WFE(void) | Ожидание события |
| void \_\_WFI(void) | Ожидание прерывания |
| void \_\_disable\_irq(void) | Глобальный запрет прерываний |
| void \_\_enable\_irq(void) | Глобальное разрешение прерываний |
| void NVIC\_EnableIRQ (IRQn\_t IRQn) | Разрешить IRQn |
| void NVIC\_DisableIRQ (IRQn\_t IRQn) | Запретить IRQn |
| uint32\_t NVIC\_GetPendingIRQ (IRQn\_t IRQn) | Вернёт 1, если прерывание  IRQn ожидает обслуживания, 0  – в противном случае |
| void NVIC\_SetPendingIRQ (IRQn\_t IRQn) | Перевести IRQn в состояние  ожидания обслуживания |
| void NVIC\_ClearPendingIRQ (IRQn\_t IRQn) | Сбросить состояние ожидания  обслуживания для IRQn |
| uint32\_t NVIC\_GetActive (IRQn\_t IRQn) | Вернуть номер IRQ текущего  активного прерывания |
| void NVIC\_SetPriority (IRQn\_t IRQn, uint32\_t priority) | Установить приоритет для IRQn |
| uint32\_t NVIC\_GetPriority (IRQn\_t IRQn) | Считать приоритет IRQn |
| void NVIC\_SystemReset (void) | Сбросить систему |

В качестве параметра IRQn\_t IRQn указывается имя прерывания, описанное в файле «MDR32Fx.h» (также представлены в таблице 5.4 в столбце «Имя прерывания»). Имя процедуры обработчика прерывания должно совпадать с именем вектора описанном в файле startup\_MDR32F9Qx.s » (также представлены в таблице 5.4 в столбце «Имя вектора»).

Таблица 5.4 – Векторы прерываний

|  |  |  |  |
| --- | --- | --- | --- |
| **Прерывание** | **Смещение** | **Имя вектора** | **Имя прерывания** |
| Внутренние | | | |
|  | 0x0000 | \_\_initial\_sp | - |
|  | 0x0004 | Reset\_Handler | - |
| -14 | 0x0008 | NMI\_Handler | NonMaskableInt\_IRQn |
| -13 | 0x000C | HardFault\_Handler | HardFault\_IRQn |
| -12 | 0x0010 | MemManage\_Handler | MemoryManagement\_IRQn |
| -11 | 0x0014 | BusFault\_Handler | BusFault\_IRQn |
| -10 | 0x0018 | UsageFault\_Handler | UsageFault\_IRQn |
|  | 0x001C – 0x0028 |  | |
| -5 | 0x002C | SVC\_Handler | SVCall\_IRQn |
| -4 | 0x0030 | DebugMon\_Handler | - |
|  | 0x0034 |  | |
| -2 | 0x0038 | PendSV\_Handler | PendSV\_IRQn |
| -1 | 0x003C | SysTick\_Handler | SysTick\_IRQn |
| Внешние | | | |
| IRQ0 | 0x0040 | CAN1\_IRQHandler | CAN1\_IRQn |
| IRQ1 | 0x0044 | CAN2\_IRQHandler | CAN2\_IRQn |
| IRQ2 | 0x0048 | USB\_IRQHandler | USB\_IRQn |
| IRQ3...IRQ4 | 0x004C – 0x0050 |  | |
| IRQ5 | 0x0054 | DMA\_IRQHandler | DMA\_IRQn |
| IRQ6 | 0x0058 | UART1\_IRQHandler | UART1\_IRQn |
| IRQ7 | 0x005C | UART2\_IRQHandler | UART2\_IRQnr |
| IRQ8 | 0x0060 | SSP1\_IRQHandler | SSP1\_IRQn |
| IRQ9 | 0x0064 |  | |
| IRQ10 | 0x0068 | I2C\_IRQHandler | I2C\_IRQn |
| IRQ11 | 0x006C | POWER\_IRQHandler | POWER\_IRQn |
| IRQ12 | 0x0070 | WWDG\_IRQHandler | WWDG\_IRQn |
| IRQ13 | 0x0074 |  | |
| IRQ14 | 0x0078 | Timer1\_IRQHandler | Timer1\_IRQn |
| IRQ15 | 0x007C | Timer2\_IRQHandler | Timer2\_IRQn |
| IRQ16 | 0x0080 | Timer3\_IRQHandler | Timer3\_IRQn |
| IRQ17 | 0x0084 | ADC\_IRQHandler | ADC\_IRQn |
| IRQ18 | 0x0088 |  | |
| IRQ19 | 0x008C | COMPARATOR\_IRQHandler | COMPARATOR\_IRQn |
| IRQ20 | 0x0090 | SSP2\_IRQHandler | SSP2\_IRQn |
| IRQ21... IRQ26 | 0x0094 – 0x00A8 |  | |
| IRQ27 | 0x00AC | BACKUP\_IRQHandler | BACKUP\_IRQn |
| IRQ28 | 0x00B0 | EXT\_INT1\_IRQHandler | EXT\_INT1\_IRQn |
| IRQ29 | 0x00B4 | EXT\_INT2\_IRQHandler | EXT\_INT2\_IRQn |
| IRQ30 | 0x00B8 | EXT\_INT3\_IRQHandler | EXT\_INT3\_IRQn |
| IRQ31 | 0x00BC | EXT\_INT4\_IRQHandler | EXT\_INT4\_IRQn |

Отдельно стоит отметить функции перевода контроллера в режим ожидания. Инструкции ожидания прерывания WFI (wait for interrupt) и WFE (wait for event) после своего выполнения немедленно переводят процессор в режим пониженного энергопотребления до возникновения любого прерывания, а при выполнении WFE процессор ещё проверяет значение регистра события. Если он равен 0, процессор приостанавливает дальнейшее выполнение команд и переходит в состояние ожидания. Если он равен 1, процессор записывает в регистр события 0 и продолжает нормальную работу без перехода в режим ожидания.

**Особенности программирования**

Для использования прерываний в программе необходимо выполнить следующие действия:

1. инициализация параметров интересующего периферийного устройства микроконтроллера;
2. маскирование событий – разрешение вызывать прерывание конкретным событиям устройства (например, получен байт данных);
3. разрешение прерывания от интересующего периферийного устройства;
4. глобальное разрешение прерываний;
5. разработка процедуры обработчика прерывания, которая будет выполняться при вызове прерывания, имя процедуры должно соответствовать имени вектора (например, void SSP1\_IRQHandler(void)).

**Внешние прерывания EXT\_INT**

В рамках данной работы будут рассмотрены внешние прерывания EXT\_INT. В контроллерах 1986ВЕ9x реализовано 4 таких прерывания, они срабатывают исключительно по высокому уровню и не имеют параметров для настройки. Для использования необходимо: выбрать интересующий вывод контроллера, см. таблицу 120 «Порты ввода-вывода» в техническом описании – раздел «Порты вода вывода MDR\_PORTx» (номер таблицы может отличаться в разных версиях документа) или см. таблицу 5.5; правильно определить режим его работы; разрешить выбранное прерывание; глобально разрешить обработку прерываний.

Информация о выводах, которые могут быть использованы в качестве EXT\_INT, представлена в таблице 5.5.

Таблица 5.5 – Описание выводов прерываний EXT\_INT

|  |  |  |
| --- | --- | --- |
| Порт | Режим (FUNC) | Прерывание |
| PA0 | Альтернативный | EXT\_INT1 |
| PB9 | Переопределенный | EXT\_INT4 |
| PB10 | Альтернативный | EXT\_INT2 |
| PB11 | Альтернативный | EXT\_INT1 |
| PC12 | Альтернативный | EXT\_INT2 |
| PC13 | Альтернативный | EXT\_INT4 |
| PD15 | Переопределенный | EXT\_INT3 |
| PE15 | Альтернативный | EXT\_INT3 |

**Системный таймер SysTick**

Также в рамках работы рассматривается системный таймер контроллера. Системный таймер представляет собой счётчик импульсов, который является составной частью процессора микроконтроллера и формирует внутреннее прерывание. Процессор имеет 24-разрядный системный таймер SysTick, который считает вниз от загруженного в него значения до нуля; перезагрузка (возврат в начало) значения в регистр LOAD происходит по следующему фронту синхросигнала, затем счёт продолжается по последующему фронту. Когда процессор остановлен для отладки, таймер не декрементируется. Данный таймер может быть использован для отсчёта временных интервалов, формирования временных задержек. Обычно используется операционными системами реального времени для распределения времени между задачами.

**Описание регистров системного таймера SysTick**

Таблица 5.6 – Регистры таймера SysTick (Базовый адрес 0xE00E010)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Смещение | Название | Доступ | Значение после  сброса МК | Описание |
| 0x00000000 | CTRL | RW | 0x00000004 | Управление счётчиком |
| 0x00000004 | LOAD | RW | 0x00000000 | Начального значения для счёта |
| 0x00000008 | VAL | RW | 0x00000000 | Текущее значение системного счетчика |
| 0x0000000C | CALIB | RO | 0x00002904 | Калибровочное значение счетчика |

**Регистр CTRL**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 31:17 | 16 | 15:3 | 2 | 1 | 0 |
| - | COUNTFLAG | - | CLKSOURCE | TICKINT | ENABLE |

**Бит 0 (ENABLE)** – включение счётчика: 1 – включен, 0 – выключен.

Когда ENABLE установлен в единицу, таймер загружает значение RELOAD из регистра LOAD и затем начинает декрементироваться. По достижению значения 0 таймер устанавливает бит COUNTFLAG и в зависимости от TCKINT генерирует запрос на прерывание. Затем загружается значение RELOAD и продолжается счёт.

**Бит 1 (TICKINT)** – разрешение запроса на прерывание: 1 – разрешено, 0 – запрещено.

**Бит 2 (CLKSOURCE)** – выбор источника синхросигнала: 0 – LSI, 1 – HCLK (сигнал от блока RST\_CLK для CPU)

**Бит 16 (COUNTFLAG)** возвращает 1, если таймер досчитал до нуля с последнего момента чтения.

**Регистр LOAD**

|  |  |
| --- | --- |
| 31:24 | 23:0 |
| - | RELOAD |

LOAD хранит значение, загружаемое в регистр VAL, когда таймер разрешён и когда достигается значение нуля. Значение RELOAD может быть любым в диапазоне 0x00000001-0x00FFFFFF. Значение 0 допустимо, но не оказывает эффекта, потому что запрос на прерывание и активизация бита COUNTFLAG происходит только при переходе таймера из состояния 1 в 0. Расчёт значения RELOAD происходит в соответствии с использованием таймера:

– для формирования короткого таймера с периодом N процессорных циклов применяется значение RELOAD, равное N-1. Например, если требуется прерывание каждые 100 циклов, то устанавливается значение RELOAD, равное 99;

– для формирования одиночного прерывания после задержки в N тактов процессора используется значение N. Например, если требуется прерывание после 400 тактов процессора, то устанавливается RELOAD, равное 400.

**Регистр VAL**

|  |  |
| --- | --- |
| 31:24 | 23:0 |
| - | VAL |

Чтение возвращает текущее значение системного таймера. Запись любого значения очищает регистр в ноль, и также очищает бит COUNTFLAG регистра CTRL.

**Регистр CALIB**

|  |  |  |  |
| --- | --- | --- | --- |
| 31 | 30 | 29:24 | 23:0 |
| NOREF | SKEW | - | TENMS |

Хранит фиксированное значение для калибровки счётчика.

Для упрощения управления таймером в файле core\_cm3.h описана функция uint32\_t SysTick\_Config(uint32\_t ticks), обеспечивающая инициализацию, включение таймера и разрешение прерывания. В функции указывается значение RELOAD.

**Практическая часть**

**Задание 5.1.** Создайте проект по алгоритму из работы 1.

Добавьте в функцию SystemInit цикл, позволяющий задержать начало выполнения основной программы процессором на 5 секунд (файл system\_MDR32F9Qx.c).

Правильно определите используемый JTAG-разъем в файле MDR32F9Qx\_config.h.

Опишите процедуру инициализации модуля RST\_CLK, обеспечивающую переход процессора на частоту 80МГц.

Опишите процедуру инициализации выводов контроллера для работы со светодиодами: PC0, PC1 для 1986ВЕ92QI или PF0, PF1 для 1986ВЕ93У.

Опишите процедуру инициализации PA0 на ввод информации с подтяжкой вниз в режиме, соответствующем прерыванию EXT\_INT1.

Опишите процедуру обработчика указанного выше прерывания. Общий вид описания процедуры представлен ниже.

void <имя вектора прерывания>(void)

{

//тело подпрограммы обработчика прерывания

}

В тело подпрограммы добавьте код изменения состояния светодиода 1 через полсекунды на противоположное. Для упрощения реализации задержки в полсекунды можно дополнительно описать функцию программной задержки (аналогично примеру из задания 1.1).

В основной программе обеспечьте тактирование используемых портов, инициализацию портов, глобальное разрешение прерываний (см. таблицу 5.3), разрешение прерывания EXT\_INT1(см. таблицы 5.3 и 5.4).

В бесконечном цикле программы добавьте код изменения состояния светодиода 0 через полсекунды на противоположное.

Выполните подключение кнопочного модуля следующим образом: вывод GND модуля подключите к питанию 3.3В – выводы 3-4 групп контактов X26, X27 (для 1986ВЕ93 X24, X25). Вывод K1 подключите к PA0 – вывод 11 контактной группы X27 (для 1986ВЕ93 X25). Подробнее см. документы «Выводы платы 1986ВЕ92QI» и «Выводы платы 1986ВЕ93У».

Выполните сборку проекта. Загрузите программу в контроллер. Проанализируйте обработку прерываний. Зафиксируйте реакцию системы в отчёте.

**Задание 5.2.** В программе из задания 1 сделайте следующие изменения:

Переопределите выводы PC0 (PF0) на PA1, PC1 (PF1) на PA2 в процедуре инициализации, обработчике прерываний, в главной программе, уберите связанные с ними программные задержки, также проинициализируйте на вывод PA3.

Добавьте в подпрограмму main (до бесконечного цикла) процедуру инициализации системного таймера SysTick\_Config(uint32\_t ticks).

Обеспечьте разрешение прерывания SysTick (см. таблицы 5.3 и 5.4).

Опишите процедуру обработчика прерывания SysTick. В теле процедуры обработчика опишите изменение значения PA3 на противоположное.

Подключите логический анализатор к выводам PA1, PA2, PA3.

Выполните сборку проекта. Загрузите программу в контроллер. Проанализируйте обработку прерываний в различных ситуациях с помощью логического анализатора. Зафиксируйте полученные диаграммы различных ситуаций обработки в отчёте.

**Задание 5.3.** В программе, полученной в задании 2, задайте приоритеты прерываниям EXT1 и SysTick, используя функцию void NVIC\_SetPriority (IRQn\_t IRQn, uint32\_t priority). Проанализируйте работу системы с двумя работающими прерываниями без задания приоритетов, с большим приоритетом для SysTick, а затем c большим приоритетом для EXT1. Выполните сборки проекта. Загрузите программы в контроллер. Проанализируйте обработку прерываний при различном порядке возникновения прерываний с помощью логического анализатора. Зафиксируйте полученные результаты в отчёте.

**Задание 5.4.** В главном цикле программы из задания 1 замените фрагмент бесконечного цикла в подпрограмме main на фрагмент, представленный ниже.

|  |
| --- |
| while(1) //основной цикл программы  {  for (int i=0; i<5; i++){  PORT\_SetBits(MDR\_PORTC, PORT\_Pin\_0);  delay (500000);  PORT\_ResetBits(MDR\_PORTC, PORT\_Pin\_0);  delay (500000);  }  \_\_WFI();//режим ожидания до выполнения прерывания  } |

\*Для микроконтроллера 1986ВЕ93У изменить MDR\_PORTC на MDR\_PORTF

Выполните сборку проекта. Загрузите программу в контроллер. Проанализируйте обработку прерываний. Зафиксируйте реакцию системы в отчёте.

**Вопросы для самоконтроля**

1. Дайте определения понятиям: событие, исключение, прерывание.

2. Назовите основные особенности обработки прерываний в МК 1986ВЕ9x.

3. Объясните назначение регистров контроллера NVIC.

4.Какие внутренние исключения обрабатываются в МК 1986ВЕ9x.?

5. Как номер прерывания влияет на порядок обработки?

6. Как задаются приоритеты прерываний?

7. Почему в таблице векторов прерываний смещение адреса соседних векторов отличается на 4?

8. Для чего может использоваться таймер SysTick операционными системами реального времени?

9. В чем заключаются особенности прерываний типа EXT\_INT в 1986ВЕ9x?

10. Какие параметры необходимо задать при инициализации порта для работы с EXT\_INT?