ПРИМЕНЕНИЕ ПРОЦЕССОРОВ СЕРИИ «МУЛЬТИКОР»

Обработка прерываний

АО НПЦ «ЭЛВИС»

support@elvees.com, www.multicore.ru

ОГЛАВЛЕНИЕ

1. ВВЕДЕНИЕ

В документе разъясняются принципы работы с прерываниями. Расшифровываются понятия «исключение», «прерывание», «обработчик». Приводится алгоритм обработки и требования к обработчику прерываний. Представлены примеры создания проекта с обработкой прерываний для сред MCStudio 3M и MCStudio 4.

В настоящем документе описание дано на примере процессора 1892ВМ10Я. Документ применим ко всем MIPS32-совместимым процессорам серии «Мультикор» (1892ВМ2Я, 1892ВМ3Т, 1892ВМ5Я, 1892ВМ7Я, 1892ВМ10Я, 1892ВМ12АТ, 1892ВМ15АФ).

2. ОСНОВНЫЕ ПОНЯТИЯ ПРИ РАБОТЕ С ПРЕРЫВАНИЯМИ

2.1 Исключение

В архитектуре MIPS понятие «исключение» (англ. exception) покрывает все виды событий, при которых CPU может прервать поток исполнения основной программы и вызвать программу-обработчик [1]. Например, переполнение в арифметической команде. Исключения обрабатываются единым механизмом, который рассмотрен ниже в настоящем документе.

> По событию исключения аппаратно CPU ничего не сохраняет в стек, не записывает в память, не сохраняет регистров. Если эти действия необходимы, они должны выполняться программно в обработчике.

Ниже перечислены все возможные исключения применительно к процессору 1892ВМ10Я по убыванию приоритета обработки.

2.1.1 Аппаратный сброс Reset

Аппаратный сброс – это событие, которое происходит при установке внешнего сигнала nRST. Когда оно возникает, процессор выполняет полную начальную инициализацию, то есть приводит автоматы к начальному состоянию и переводит процессор в состояние, из которого он может начать запуск команд, находящихся в некэшируемой и неотображаемой области. После возникновения исключения аппаратного сброса состояние процессора не определено, за исключением следующего:

- регистр CP0.Random устанавливается в значение, равное количеству строк TLB -1;
- регистр CP0.Wired устанавливается в 0;
- регистр CP0.Config устанавливается в свое начальное состояние (boot state);
- поля BEV, TS, NMI и ERL регистра CP0.Status устанавливаются в заданные значения;
- в PC загружается значение 0xBFC0_0000 (виртуальный адрес);

Устанавливаются следующие значение регистра CP0.Status:

RP <= 0

BEV <= 1

TS <= 0

NMI <= 0

ERL <= 1

2.1.2 Немаскируемое прерывание NMI

Немаскируемое прерывание (nonmaskable interrupt, NMI) возникает по положительному фронту входного сигнала NMI или при срабатывании сторожевого таймера WDT. Оно не вызывает сброса или другую переинициализацию аппаратных средств. Состояние кэш, памяти, а также другие состояния процессора остаются неизменными. Значения регистров также сохраняются за исключением следующего:

- поля BEV, TS, NMI и ERL регистра CP0.Status принимают заданные значения;
- в регистр CP0.ErrorEPC загружается значение PC - 4, если прерывание произошло на фоне команды в слоте задержки перехода. В противном случае в регистр CP0.ErrorEPC загружается значение PC;
- в PC загружается значение 0xBFC0_0000.

Устанавливаются следующие значение регистра CP0.Status:

BEV <= 1

TS <= 0

NMI <= 1

ERL <= 1

2.1.3 Исключения TLB

В этом пункте перечислены все возможные исключения, возникающие при работе с буфером ассоциативной трансляции (translation lookaside buffer, TLB).

2.1.3.1 Промах буфера ассоциативной трансляции TLB_Ri

Исключение TLB Refill происходит во время выборки команды или доступа к данным, если в TLB нет ни одной строки, соответствующей ссылке к отображенному адресному пространству, и бит EXL в регистре CP0.Status равен 0.

Процессор может хранить несколько последних записей, прочитанных из таблицы страниц в небольшой кэш-памяти, называемой буфером ассоциативной трансляции (TLB). Процессор «заглядывает» в TLB в поисках информации о трансляции прежде чем обратиться к таблице страниц в физической памяти. В реальных программах большинство обращений находят в TLB нужную информацию (в этом случае говорят, что произошло попадание в TLB).

2.1.3.2 TLB_Ii

Исключение TLB Invalid происходит во время выборки команды или доступа к данным в одном из следующих случаев:

- в TLB нет ни одной строки, соответствующей ссылке к отображенному адресному пространству, и бит EXL в регистре CP0.Status равен 1;
- строка TLB соответствует ссылке к отображенному адресу, но ее бит валидности выключен;

2.1.3.3 Промах TLB при загрузке данных TLB_Rd

Исключение (TLB miss) возникает, когда при обращении к TLB в ней нет соответствующего адреса.

2.1.3.4 Попадание в инвалидную страницу TLB (V=0) при загрузке данных TLB_Id

Исключение (TLB invalid exception) возникает, когда адрес найден, но он недействителен.

2.1.3.5 Исключение сохранения в запрещенной области TLB_M

Это исключение (TLB Modified Exception) возникает при обращению по записи данных к отображенному адресу, если выполняется следующее условие: найденная строка TLB действительна, но страница запрещена для записи.

2.1.4 Исключение по ошибке адресации во время доступа к команде или данным AdELi

Исключение AdELi (Address Error Exception – Instruction Fetch/Data Access) возникает при попытке выполнить одно из следующих действий:

- выбрать команду, загрузить или сохранить слово данных, если они не выровнены в границах слова;
- загрузить или сохранить половину слова, если оно не выровнено в границах полуслова;
- обратиться по адресу пространства Kernel при работе в режиме User.

2.1.5 Исключение по аппаратному контролю Mcheck

Данное исключение возникает, если при выполнении команды записи в TLB (TLBWI или TLBWR) обнаруживается, что поле виртуального адреса записываемой строки со-ответствует такому же полю одной из строк, уже хранящихся в TLB.

2.1.6 Исключения исполнения

В архитектуре процессора серии «Мультикор» существует 6 исключений исполнения. Они обладают одинаковым приоритетом и имеют целью быть легко узнаваемыми в процессе создания ПО в безопасном режиме (системные вызовы, условные ловушки и точки останова).

2.1.6.1 Системный вызов System Call

Исключение System Call возникает при исполнении команды SYSCALL.

2.1.6.2 Останов в контрольной точке Breakpoint

Исключение Breakpoint возникает при исполнении команды BREAK.

2.1.6.3 Недоступность процессора CpU

Исключение недоступности сопроцессора (Coprocessor Unusable) вызывается при попытке исполнения команды сопроцессора CP0 в режиме User, или в случае, если происходит обращение к другому сопроцессору, доступ к которому запрещен битами CU[3:1] регистра CP0.Status.

2.1.6.4 Зарезервированная команда RI

Исключение зарезервированной команды (Reserved Instruction). вызывается при исполнении команды с неопределенным старшим кодом операции (major opcode) или полем функции.

2.1.6.5 Целочисленное переполнение (Ov)

Исключение целочисленного переполнения (Integer Overflow) вызывается, когда выбранные целочисленные команды приводят к переполнению в двоичном коде.

2.1.6.6 Trap

Исключение Trap вызывается, если условие команды trap истинно (TRUE).

2.1.7 Ошибка выравнивания адреса при загрузке данных AdELd

Ссылка на адрес режима Kernel при работе в режиме User при загрузке данных.

2.1.8 Ошибка выравнивания адреса при сохранении данных AdES

Попытка сохранения по адресу Kernel в режиме User.

2.1.9 Исключение прерывания (Interrupt Exception)

Об этом виде исключений – в разделе 2.2.

2.2 Прерывание

Прерывание (interrupt exception) – это разновидность исключения. Особенность прерываний в том, что они возникают вне CPU и используются для привлечения внимания CPU к какому-то внешнему событию. У программиста есть возможность заблокировать (замаскировать) возникновение прерывания от периферийного устройства, если это необходимо. Все остальные исключения замаскировать нельзя – они происходят в случае соответствующих событий.

2.3 Обработчик

Обработчик исключений – это код, являющийся, как правило, частью операционной системы, который выясняет причину возникновения исключения и реагирует соответствующим образом (например, чтением клавиатуры в ответ на прерывание). Затем управление возвращается программе, выполнявшейся до возникновения исключения.

Обработчик должен быть размещен по адресу вектора соответствующего исключения. Когда возникает исключение, процессор всегда переходит по этому адресу. Допустимые адреса векторов исключений указаны в соответствующих разделах руководства пользователя на конкретную микросхему.

> Часто по вектору обработчика размещают минимальный обработчик, который выполняет только сохранение контекста и переход на основную функцию, которая располагается в отличных от вектора областях.

2.4 Вектор исключения

Вектор исключения – это адрес, с которого стартует процессор при обработке исключений. По этому адресу должен быть размещен обработчик исключения.

В руководстве пользователя на процессор имеются таблицы, позволяющие определить вектор для каждого вида прерывания (таблицы 3.16, 3.17, 3.18 для процессора 1892ВМ10Я). Значения этого адреса зависят от битов BEV, EXL, IV регистра CP0.Status и состояния бита TR_CRAM системного регистра CSR.

> Векторы исключений аппаратного сброса и NMI всегда находятся по адресу 0xBFC_0000.
>
> Обработчик прерывания от периферийного устройства может быть расположен по адресам 0x8000_0180, 0xBFC0_0380, 0xB800_0180.

3. ОБРАБОТКА ИСКЛЮЧЕНИЙ

Чтобы процессор вошел в прерывание, необходимо их разрешить. Прерывания разрешаются, когда полям регистра CP0.Status присвоены на следующие значения:

- IE = 1;
- EXL = 0 (Exception level);
- ERL = 0.

Программно необходимо установить бит глобального разрешения прерываний IE (interrupt enable) регистра CP0.Status[0]. Запросы на прерывания поступают в регистр CP0.Cause (поле IP[7:0]).

3.1 Обработка вешних и внутренних прерываний

Внешние и внутренние прерывания поступают на вход одного из псевдорегистров QSTR и, если прерывание незамаскировано в регистре MASKR, устанавливается соответствующий бит в поле IP[7:2] регистра CP0.Cause. Поле IP[7:2] показывает, какая линия группы прерываний активна, а поле IM[7:2] – какие группы прерываний разрешены в настоящий момент. То есть, необходимо в поле IM[7:2] регистра CP0.Status (Status[15:10]) разрешить те из прерываний, которые нужны для работы, и внести соответствующее значение в регистр MASKR. После этого прерывания будут доступны.

Для иллюстрации того, какие биты поля IM[7:0] разрешают различные группы прерываний, см. таблицу ниже.

Таблица 3.1

Биты поля IM	Прерывания
IM[7]	Прерывание от таймера COMPARE[1] (встроенного в ядро CPU)
IM[6]	Прерывания от DSP, объединенные по ИЛИ.
IM[4]	Прерывания, объединенные по ИЛИ впсевдорегистре QSTR2.
IM[3]	Прерывания, объединенные по ИЛИ в псевдорегистре QSTR1.

[1]Прерывание COMPARE формируется, когда значение регистра CP0.Count инкрементируется и становится равным значению регистра CP0.Compare. Прерывание снимается записью в регистр CP0.Compare.

Для реализации периодического прерывания обработчик прерываний должен каждый раз увеличивать значение CP0.Compare на фиксированную величину. После установки CP0.Compare необходимо программно проверять, читая значение CP0.Count , не является ли значение, установленное в CP0.Compare, уже пройденным счетчиком Count, так как за время обработки прерывания он отсчитывал такты.

IM[2]	Прерывания, объединенные по ИЛИ в псевдорегистре QSTR0.
IM[1:0]	Запросы программного прерывания.

3.2 Обработка программных прерываний

Программные прерывания поступают в IP[1:0] напрямую, прерывания не маскируются регистрами MASKR и возникают после снятия маски IM[1:0] программой (программа может их сгенерировать записью в биты CP0.Cause[9:8]).

3.3 Последовательность действий CPU при возникновении прерывания

Обнаружив одно из исключений, CPU приостанавливает нормальную последовательность исполнения команд, запускается программа обработчика исключений, расположенную в фиксированных адресах памяти.

Последовательность действий CP0:

1. Устанавливается бит EXL в регистре состояния (CP0.Status), процессор входит в режим Kernel, прерывания отключаются.
2. При возникновении исключения в регистр Exception Program Counter (CP0.EPC) загружается адрес, начиная с которого исполнение команд может возобновиться после завершения обработки исключения. В регистр CP0.EPC помещается адрес команды, вызвавшей исключение или, если команда находилась в слоте задержки перехода, адрес команды перехода, предшествующей слоту задержки. Чтобы различить эти ситуации, программное обеспечение должно проанализировать бит BD (branch delay) в регистре CP0.Cause (регистр причины исключения).
3. Процессор заполняет необходимые регистры CP0 значениями, относящимися к состоянию исключения.
4. Изменяет счетчик команд (PC) на адрес соответствующего вектора обработки исключения.
5. Очищает признаки исключения, относящиеся к более ранним стадиям конвейера.
6. Записывает специальный код в регистре CP0.Cause, сохраняя, таким образом, информацию о причине исключения.

> Некоторые исключения могут произойти одновременно, в этом случае обрабатывается исключение с наивысшим приоритетом. В таблице 3.15 руководства пользователя на микросхему 1892ВМ10Я перечислены все возможные исключения со своими относительными приоритетами, от высшего к низшему.

3.4 Последовательность действий программы-обработчика прерываний

1. Обработчик сохраняет контекст процессора – содержимое счетчика команд, текущий режим процессора и статус разрешения прерываний. Таким образом, контекст может быть восстановлен по завершению обработки исключения.
2. Читается поле ExcCode регистра CP0.Cause:

- если это исключение прерывания (ExcCode==0), то
  - ищется источник прерывания,
  - выполняются действия, соответствующие возникновению данного прерывания (непосредственно обработка, устранение причины прерывания),
  - сбрасывается флаг прерывания;
- если это исключение по другой причине (ExcCode!=0), то необходима обработка в зависимости от конкретной причины исключения. Например, исключение по промаху TLB – штатная ситуация в операционных системах, тогда как исключение по зарезервированной инструкции чаще всего свидетельствует об ошибках в функционировании программы.

3. Восстанавливается контекст.
4. Выполняется инструкция ERET, выполняющая выход из режима обработки исключения и возврат на ту точку, в которой находился процессор во время возникновения исключения.

Ниже представлен пример кода «первичного» обработчика прерывания, сохраняющего контекст и переходящего на «основной» обработчик. После возврата из основного обработчика контекст восстанавливается и выполняется инструкция ERET.

```
.set noat
.text
Interrupt:

/* Сместить указатель стека на 31*4+24 байта вниз */
    addiu      $29,$29,-(31*4+24)
/* Сохранить в стеке регистры 1-28,30,31 */
    sw         $1,(0)($29)
    sw         $2,(4)($29)
    sw         $3,(8)($29)
    sw         $4,(12)($29)
    sw         $5,(16)($29)
    sw         $6,(20)($29)
    sw         $7,(24)($29)
    sw         $8,(28)($29)
    sw         $9,(32)($29)
    sw         $10,(36)($29)
    sw         $11,(40)($29)
    sw         $12,(44)($29)
    sw         $13,(48)($29)
    sw         $14,(52)($29)
    sw         $15,(56)($29)
    sw         $16,(60)($29)
    sw         $17,(64)($29)
    sw         $18,(68)($29)
    sw         $19,(72)($29)
    sw         $20,(76)($29)
    sw         $21,(80)($29)
    sw         $22,(84)($29)
    sw         $23,(88)($29)
    sw         $24,(92)($29)
    sw         $25,(96)($29)
    sw         $26,(100)($29)
```

```
    sw          $27,(104)($29)
    sw          $28,(108)($29)
    sw          $30,(112)($29)
    sw          $31,(116)($29)
    /* Вызов функции основного обработчика */

    la   $26, int_handler
    jalr  $26
    nop

/* Восстановить регистры 1-28,30,31 из стека */
    lw          $1,(0)($29)
    lw          $2,(4)($29)
    lw          $3,(8)($29)
    lw          $4,(12)($29)
    lw          $5,(16)($29)
    lw          $6,(20)($29)
    lw          $7,(24)($29)
    lw          $8,(28)($29)
    lw          $9,(32)($29)
    lw          $10,(36)($29)
    lw          $11,(40)($29)
    lw          $12,(44)($29)
    lw          $13,(48)($29)
    lw          $14,(52)($29)
    lw          $15,(56)($29)
    lw          $16,(60)($29)
    lw          $17,(64)($29)
    lw          $18,(68)($29)
    lw          $19,(72)($29)
    lw          $20,(76)($29)
    lw          $21,(80)($29)
    lw          $22,(84)($29)
```

```
    lw          $23,(88)($29)
    lw          $24,(92)($29)
    lw          $25,(96)($29)
    lw          $26,(100)($29)
    lw          $27,(104)($29)
    lw          $28,(108)($29)
    lw          $30,(112)($29)
    lw          $31,(116)($29)
/* Восстановить указатель стека */
    addiu       $29,$29,31*4+24

/* Возврат из прерывания */
    eret
    nop
```

4. СОЗДАНИЕ ПРОЕКТА С ОБРАБОТКОЙ ПРЕРЫВАНИЙ ДЛЯ MCSTUDIO 3M

В составе сред разработки и отладки программ MCStudio есть примеры проектов с обработкой прерываний:

- MEM_DMA_DSP_int,
- MEM_DMA_int,
- MFBSP_I2S_DMA_int,
- MFBSP_LPORT_DMA_int,
- MFSP_SPI_int,
- WDTimhandler_itimer_mode,
- WDTimer_handler_watchdog_mode
- IT_int_handler,
- ITimer_handler_NVCom02T.

В этом разделе проиллюстрирован процесс создания такого проекта. Описано создание проекта с обработкой прерываний от интервального таймера IT0 (IT_int_handler).

Начало – стандартное для создания проектов.

В главном меню в выбрать File->New Project и дать имя проекту:

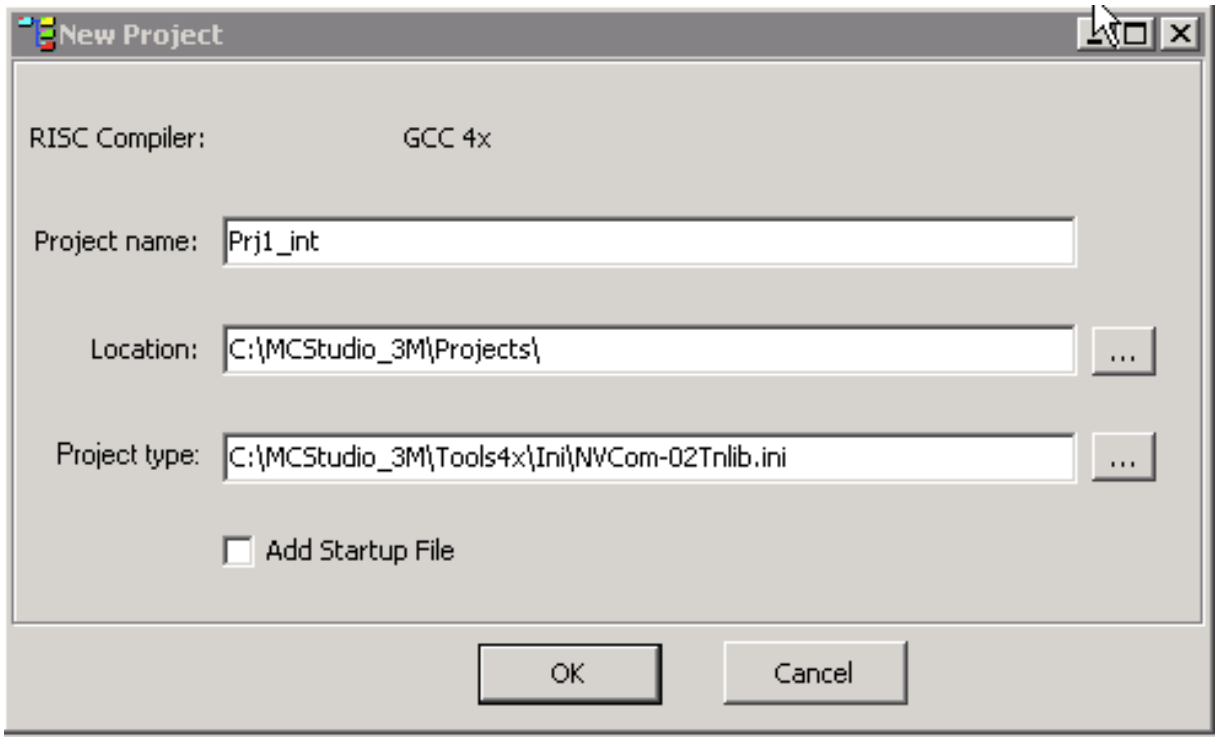

Рисунок 4.1

В строке Project type выбрать библиотеку для процессора, под который создается проект, см. **Рисунок 4.2.**

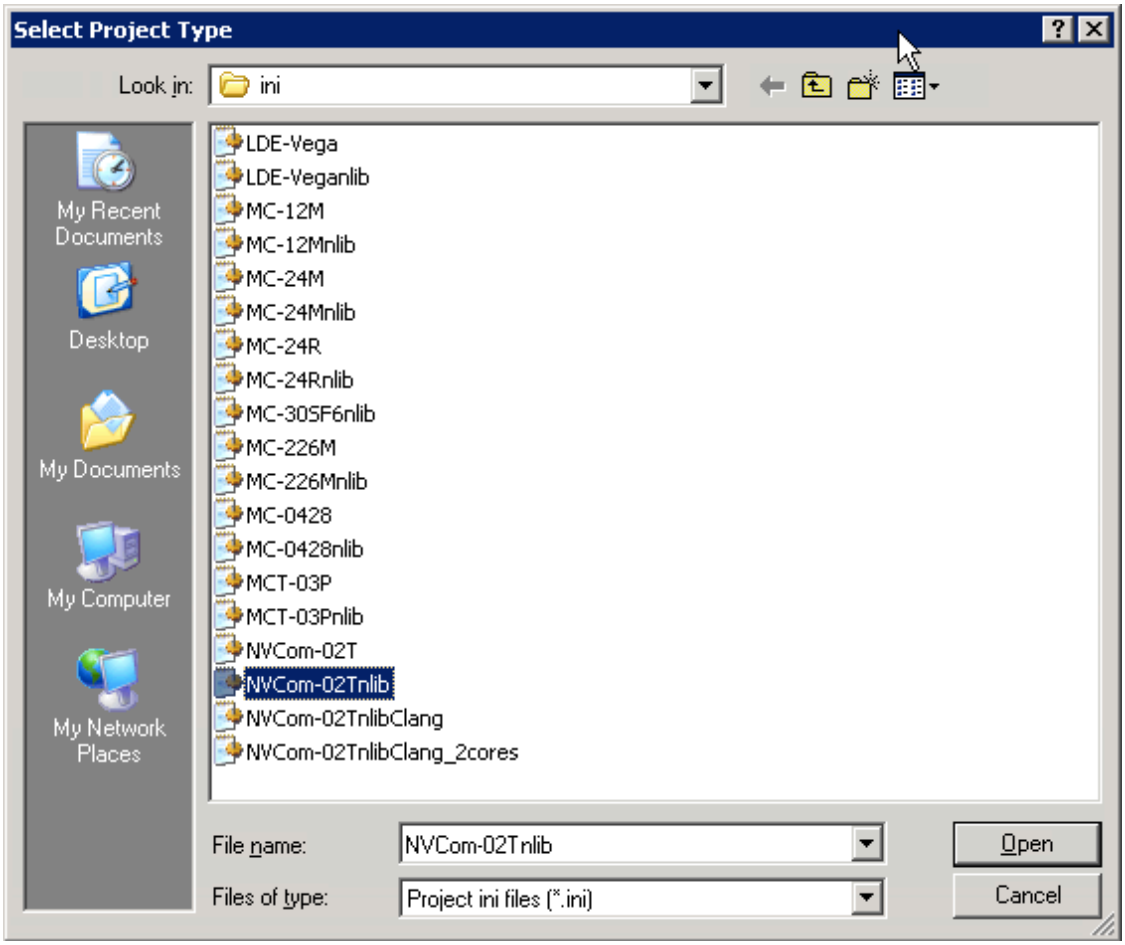

Рисунок 4.2

Для состава проекта с обработкой прерывания необходимый минимум – файл с основной программой (main.c) и файл-обработчик прерывания (handler.s). Следовательно, далее правой клавишей мыши выбрать выпадающее меню и прибавить файлы к проекту:

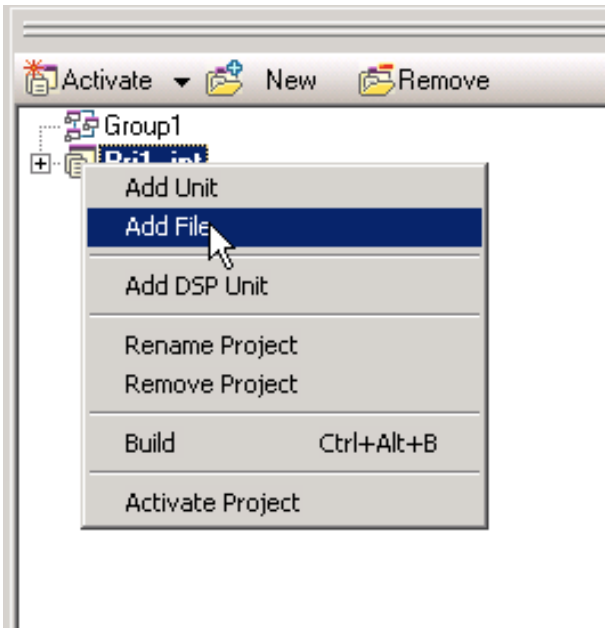

Рисунок 4.3

В файл handler.s необходимо вставить текст первичного обработчика, код которого приведен в разделе 3.4.

Первичный обработчик из файла handler.s переходит на основной (строка «la $26, int_handler»). Основной обработчик создается в файле main.c (void int_handler()). В нем опрашиваются регистры QSTR0 и MASKR0 на предмет возникновения прерывания от таймера и регистр ITCSR0 на прерывание от интервального таймера 0, и если оно произошло, флаг прерывания сбрасывается. Далее происходит возврат на первичный обработчик. Восстанавливается контекст. Выполняется инструкция ERET, выполняющая выход из режима обработки исключения и возврат на ту точку, в которой находился процессор во время возникновения исключения. Прерывание обработано. Когда таймер снова досчитает до определенного значения и прерывание опять произойдет, CP0 снова запустит первичный обработчик, изменив счетчик команд (PC) на адрес соответствующего вектора обработки исключения.

Функция основного обработчика, ее необходимо добавить в основную программу:

```
void int_handler() {
unsigned int ActiveIRQ;
    ActiveIRQ = QSTR0 & MASKR0;
    // Если это прерывание таймера
    if ( (ActiveIRQ & (1<<22)) != 0 ) {
    // Если это прерывание от интервального таймера 0
        if ( (ITCSR0 & 2) != 0 )
//сбрасываем флаг прерывания
                ITCSR0 = 1;

        }
}
```

В основную программу необходимо добавить первичные настройки – код, инициализирующий регистры CP0.Status и MASKR:

```
MASKR0 |=(1<<22); Разрешаем прерывание от таймера IT0
//Разрешаем прерывания от внутренних устройств микросхемы
    asm("or $4,$0,$0"); //0->GPR#4
    asm("li $4,0x401");
    asm("mtc0 $4,$12"); //GPR#4 -> CP0.Status
```

Затем в настройках проекта выбрать вкладку секций данных (главное меню ->Options->Sections) и вписать в ячейки таблицы адреса сегментов данных и адреса сегментов кода файлов (main.c и handler.s), см. Рисунок 4.4. Название секции .text, соответствующее обработчику, необходимо переименовать, например в «handler». Это нужно для того, чтобы среда разработки позволила разместить эту секцию в адресах, отличных от адресов секции .data обработчика. Обработчик должен быть размещен по адресу, соответствующему таблице векторов прерываний для конкретного случая, как указано в руководстве пользователя (таблица 3.18 для 1892ВМ10Я). В данном случае представлен пример расположения во внутренней памяти (CRAM).

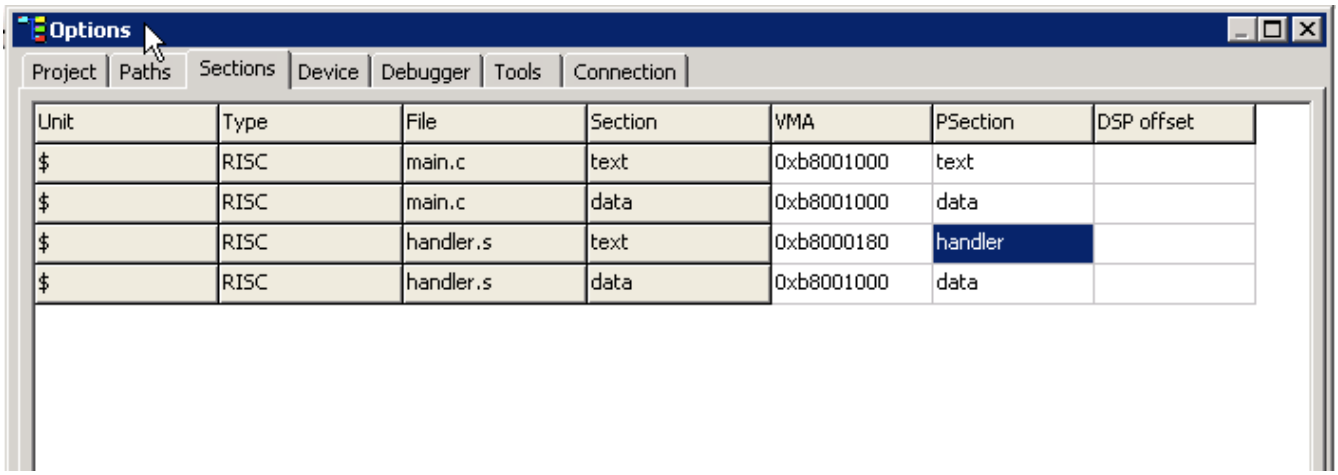
Options
Project
Paths
Sections
Device
Debugger
Tools
Connection
Unit
Type
File
Section
VMA
PSection
DSP offset
$
RISC
main.c
text
0xb8001000
text
$
RISC
main.c
data
0xb8001000
data
$
RISC
handler.s
text
0xb8000180
handler
$
RISC
handler.s
data
0xb8001000
data

Рисунок 4.4

5. ПРОЦЕСС СОЗДАНИЯ ПРОЕКТА С ОБРАБОТКОЙ ПРЕРЫВАНИЙ ДЛЯ MC STUDIO 4

Ниже проиллюстрирован процесс создания проекта с обработкой прерываний от интервального таймера (IT1) для процессора 1892ВМ10Я в среде MC Studio 4 (IT_int_handler).

Начало – стандартное для создания проектов.

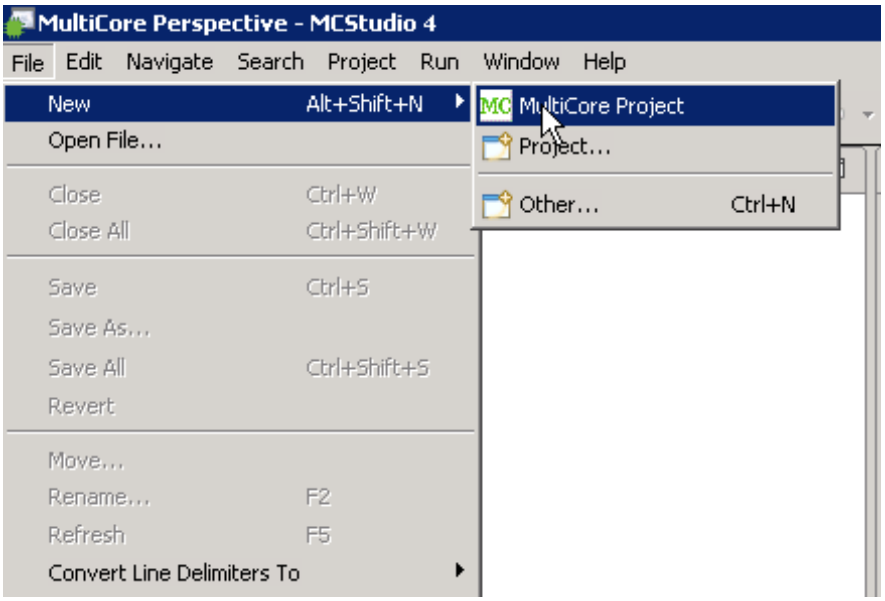

Рисунок 5.1

Задать название проекту, выбрать процессор и тип проекта – с библиотекой Newlib, нажать кнопку Next на нижней панели окна:

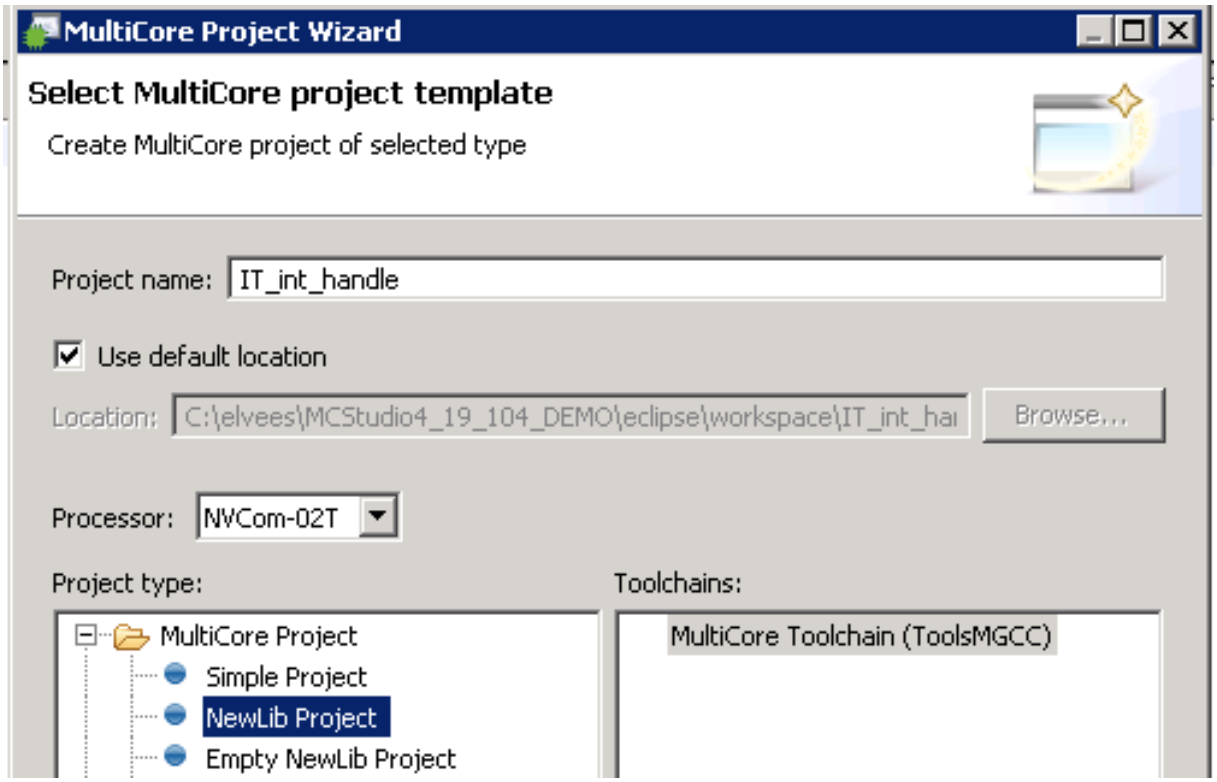

Рисунок 5.2

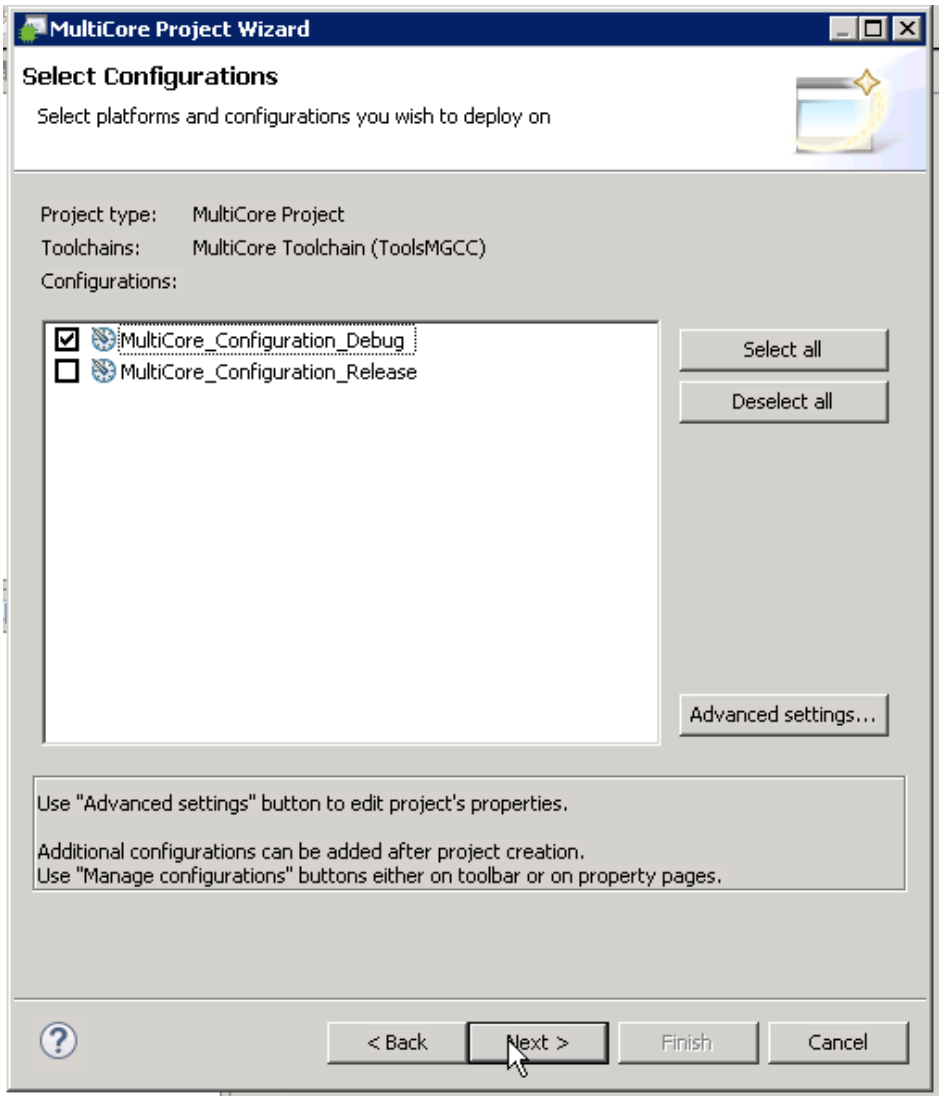

Рисунок 5.3

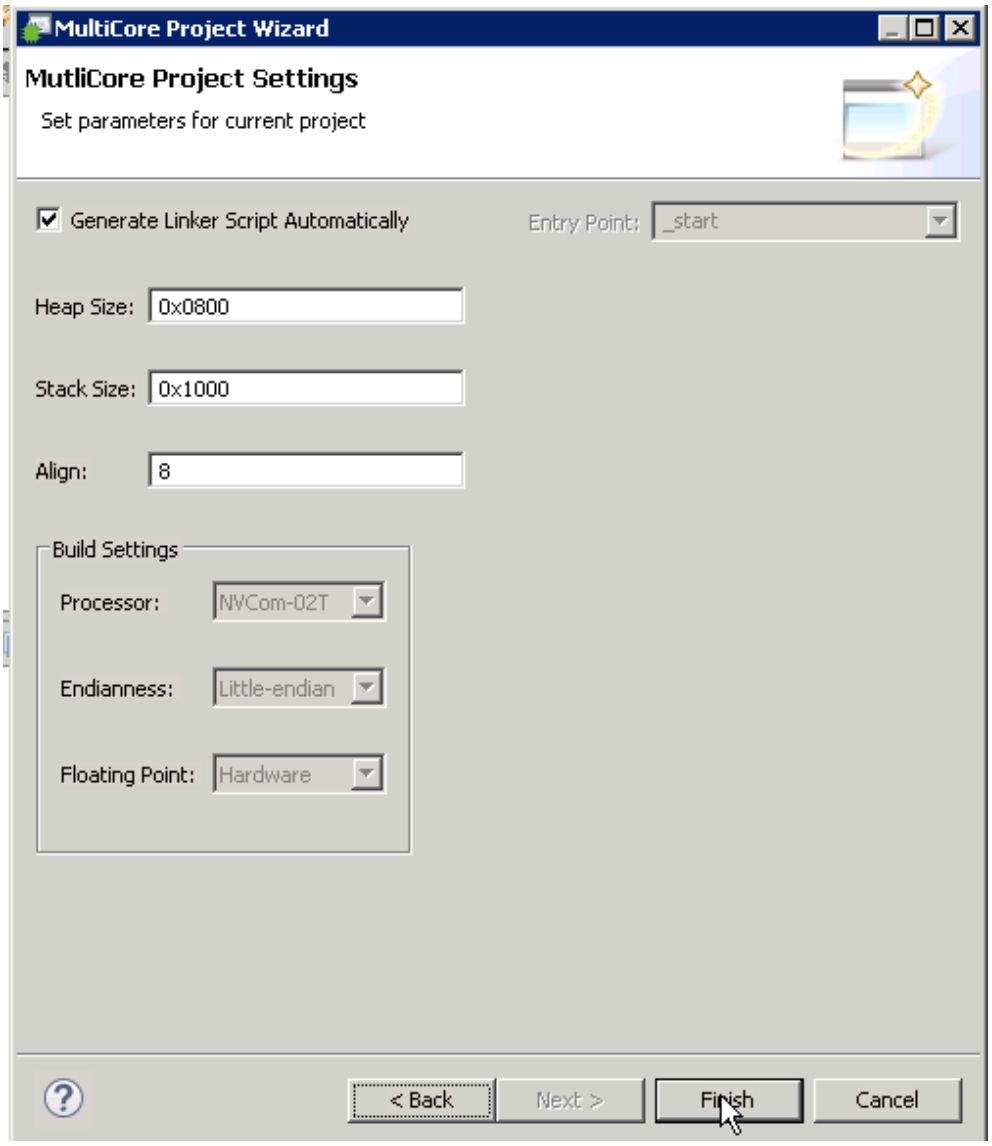

Рисунок 5.4

Далее необходимо добавить новый файл к проекту – handler.s. Файл main.c создается автоматически вместе с проектом.

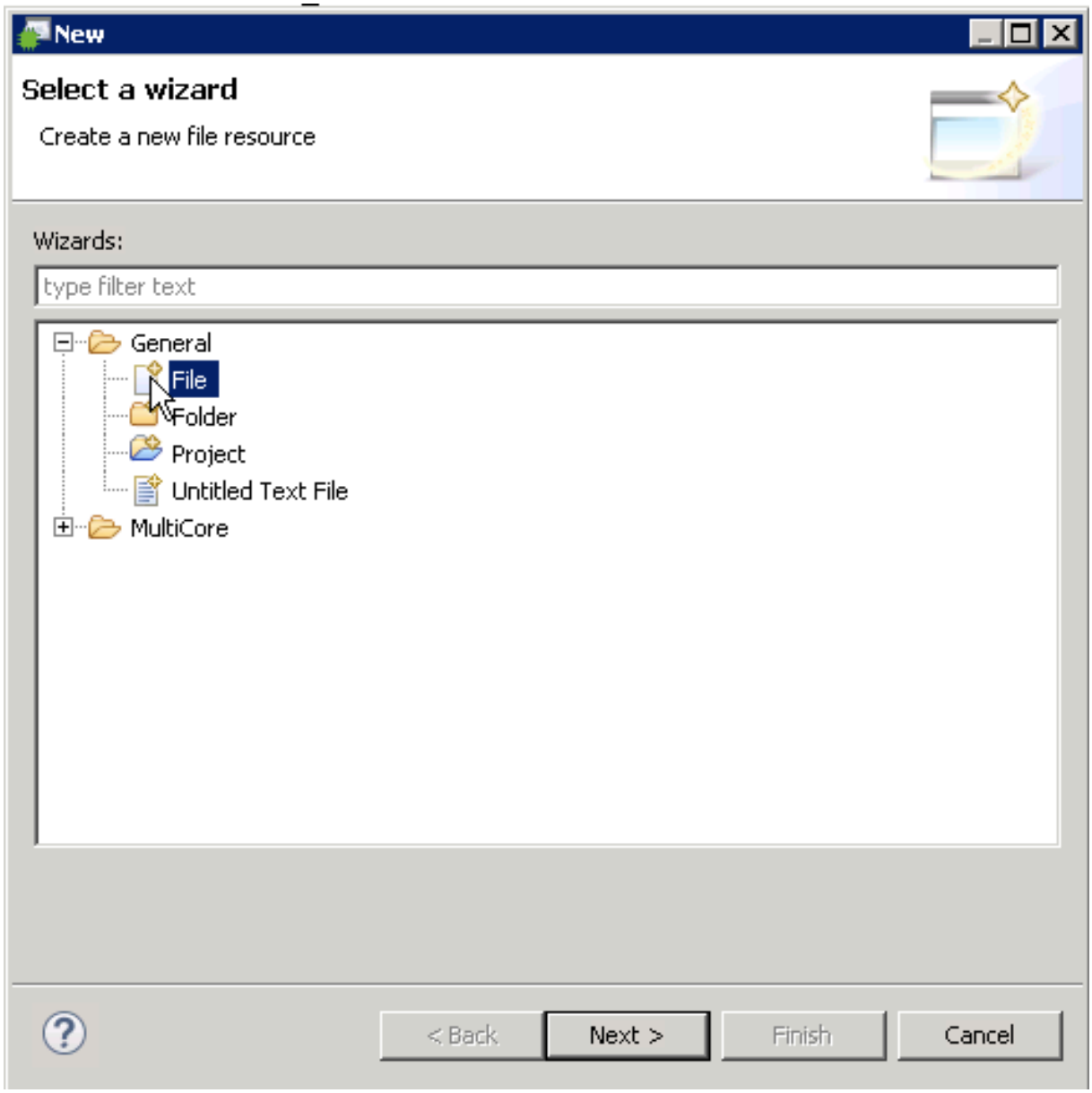
New
Select a wizard
Create a new file resource
Wizards:
type filter text
General
File
Folder
Project
Untitled Text File
MultiCore
< Back
Next >
Finish
Cancel

Рисунок 5.5

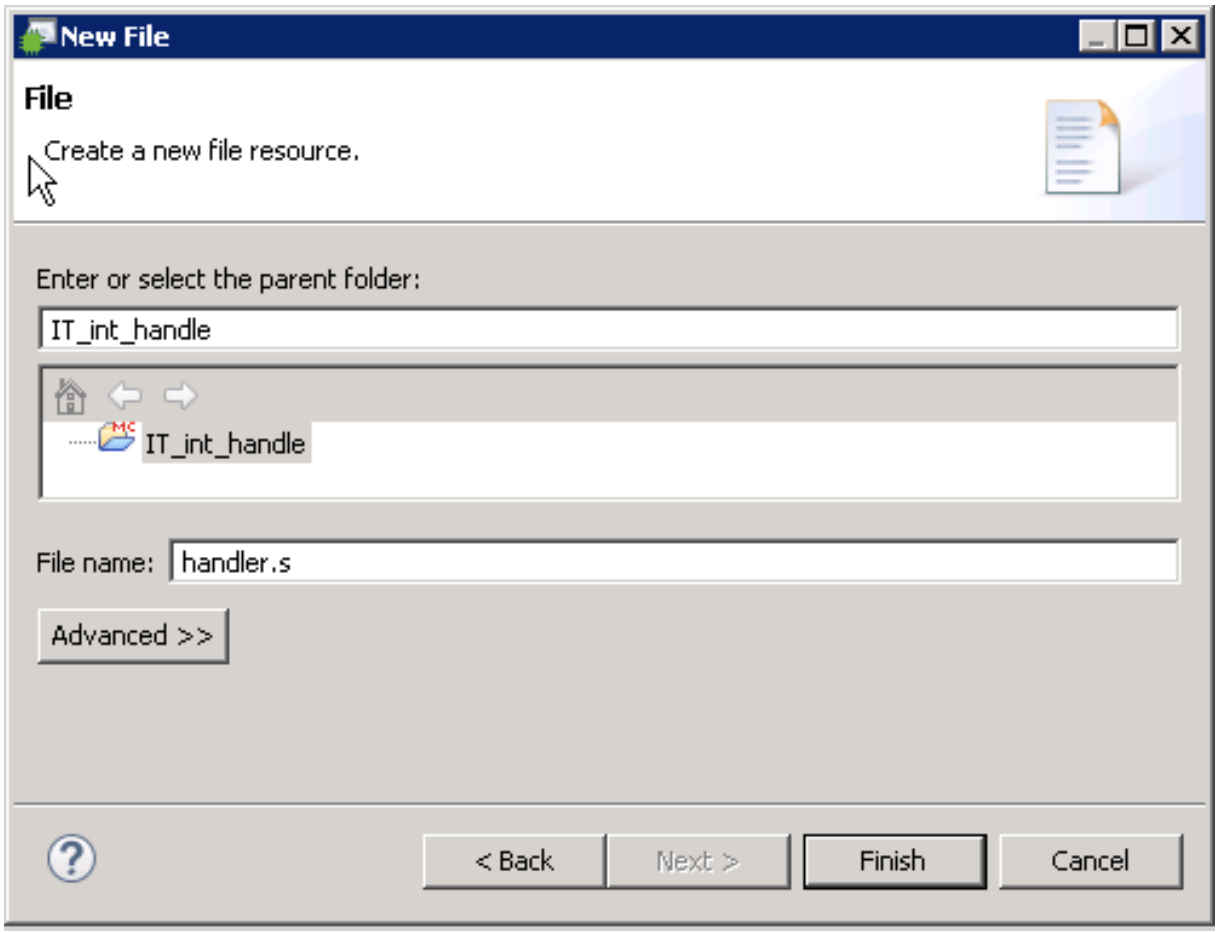

Рисунок 5.6

Файл появился в составе проекта:

Рисунок 5.7

В файл handler.s необходимо вставить текст первичного обработчика, код которого приведен в разделе 3.4.

Первичный обработчик из файла handler.s переходит на основной (строка «la $26, int_handler»). Основной обработчик создается в файле main.c (void int_handler()). В нем опрашиваются регистры QSTR0 и MASKR0 на предмет возникновения прерывания от таймера и регистр ITCSR0 на прерывание от интервального таймера 0, и если оно произошло, флаг прерывания

сбрасывается и интервальный таймер заново запускается. Далее происходит возврат на первичный обработчик. Восстанавливается контекст. Выполняется инструкция ERET, выполняющая выход из режима обработки исключения и возврат на ту точку, в которой находился процессор во время возникновения исключения. Прерывание обработано. Когда таймер снова досчитает до определенного значения и прерывание опять произойдет, CP0 снова запустит первичный обработчик, изменив счетчик команд (PC) на адрес соответствующего вектора обработки исключения.

Ниже – код основного обработчика.

```
void int_handler() {


unsigned int ActiveIRQ;
    ActiveIRQ = SYS_REG.QSTR0.data & SYS_REG.MASKR0.data;
    // Если это прерывание от интервального таймера
    if ( (ActiveIRQ & (1<<21 )) != 0 ) {
        // Если это прерывание от интервального таймера
            i++;
            tmp ^= 0xFFFF;
            MFBSP0.GPIO_DR.data = tmp;
            // сбрасываем флаг прерывания
            IT1.ITCSR.bits.INT = 0;
            // снова запускаем таймер
            IT1.ITCSR.bits.EN = 1;
            //  флаг завершения обработки прерывания
            flag = 1;
        }
}
```

В основную программу необходимо добавить первичные настройки – код, инициализирующий регистры CP0.Status и MASKR:

```
// Разрешаем прерывания вообще (CP0.Status[0]) и
// прерывания в QSTR0 в частности (CP0.Status[10])
      SetCP0_Status(0x401);


    // Разрешаем прерывания от интервального таймера 0
    SYS_REG.MASKR0.data |= 1<<21;
```

Следующим этапом необходимо перейти в свойства проекта, чтобы задать адреса секций кода и данных.

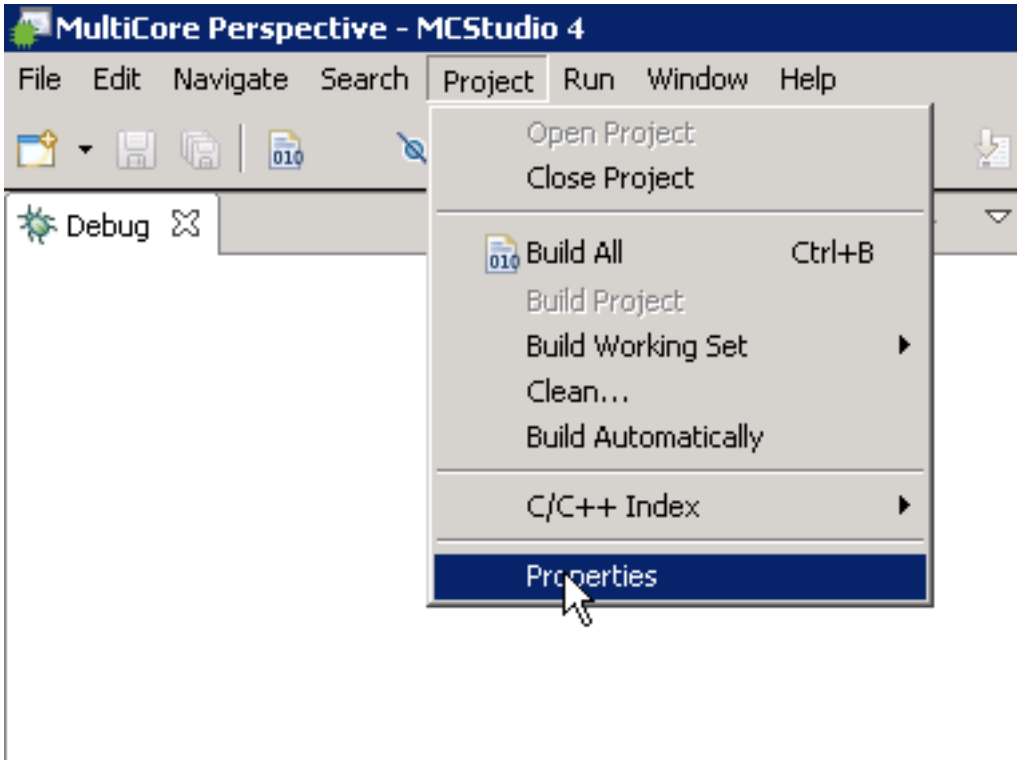

Рисунок 5.8

Далее, в настройках проекта выбрать вкладку секций данных (главное меню ->Options->Sections) и вписать в ячейки таблицы адреса сегментов данных и адреса сегментов кода файлов (main.c и handler.s), см. Рисунок 5.9. Название секции .text, соответствующее обработчику, необходимо переименовать, например в «handler». Это нужно для того, чтобы среда разработки позволила разместить эту секцию в адресах, отличных от адресов секции .text остальной программы. Обработчик должен быть размещен по адресу, соответствующему таблице векторов прерываний для конкретного случая, как указано в руководстве пользователя (таблица 3.18 для 1892ВМ10Я). В данном случае представлен пример расположения во внутренней памяти (CRAM).

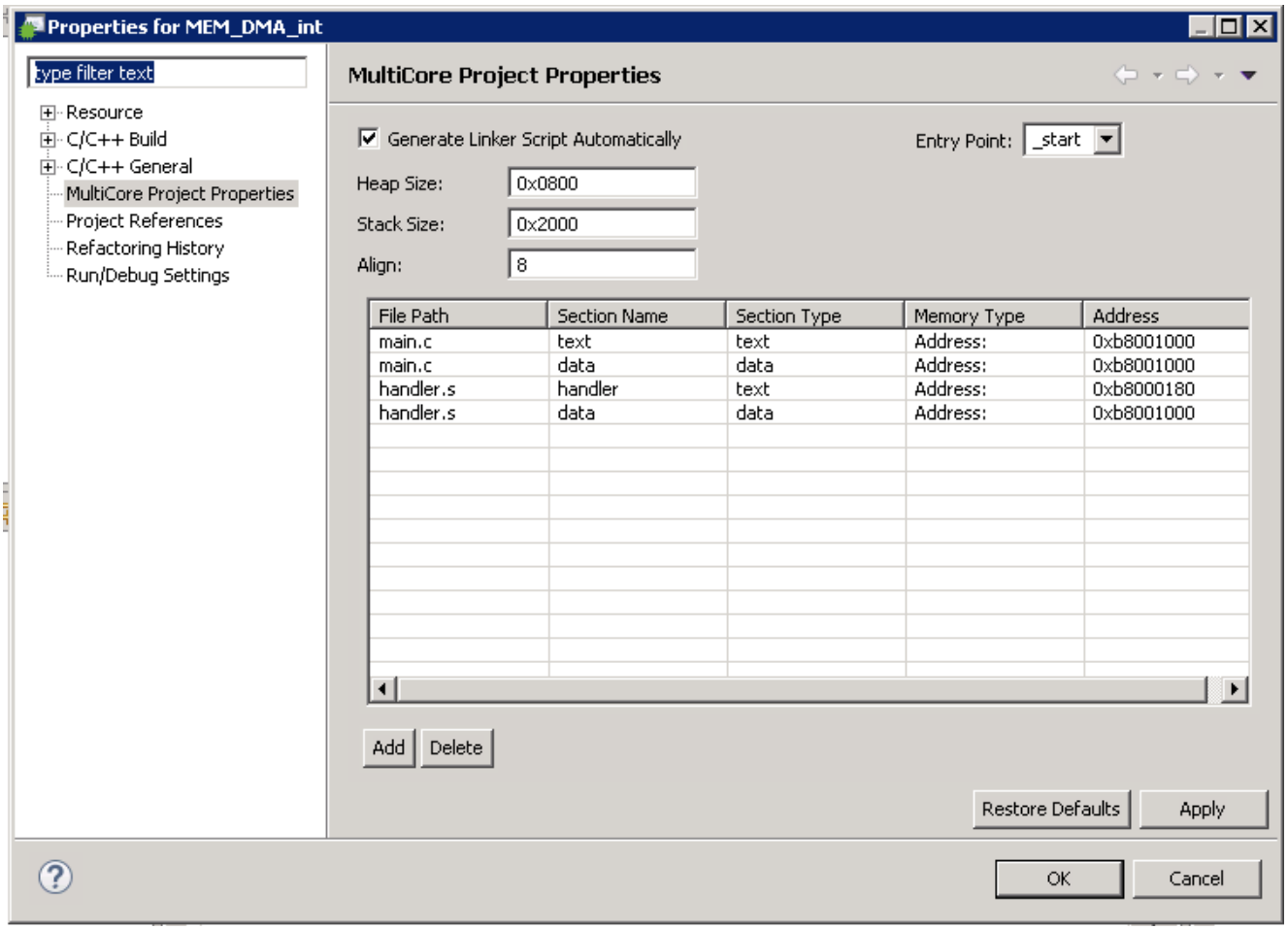
Properties for MEM_DMA_int
type filter text
Resource
C/C++ Build
C/C++ General
MultiCore Project Properties
Project References
Refactoring History
Run/Debug Settings
MultiCore Project Properties
Generate Linker Script Automatically
Entry Point: _start
Heap Size: 0x0800
Stack Size: 0x2000
Align: 8
File Path | Section Name | Section Type | Memory Type | Address
main.c | text | text | Address: | 0xb8001000
main.c | data | data | Address: | 0xb8001000
handler.s | handler | text | Address: | 0xb8000180
handler.s | data | data | Address: | 0xb8001000
Add
Delete
Restore Defaults
Apply
OK
Cancel

Рисунок 5.9

6. СПИСОК РЕКОМЕНДУЕМОЙ ЛИТЕРАТУРЫ

Ниже приведен список источников, в которых описывается работа с прерываниями в MIPS-совместимых процессорах:

1. Dominic Sweetman. See MIPS® Run. Second Edition. - Morgan Kaufmann Publishers, 2007.
2. Д.М. Харрис и С.Л. Харрис. Цифровая схемотехника и архитектура компьютера. Второе издание -, Morgan Kaufman, 2013.

7. ИСТОРИЯ ИЗМЕНЕНИЙ

16 февраля 2017

- В главу 2.2 добавлен список исключений, которые обрабатывает процессор 1892ВМ10Я.
- Скорректировано введение главы 3, добавлена таблица 3.1 со сноской.