Быстрое погружение
с «черными плавниками»

Андрей САВИЧЕВ
avisv@yandex.ru
Олег РОМАНОВ
oleg.rom@eltech.spb.ru

При освоении современных высоких технологий нередко возникает психологический барьер, связанный как с их объективной сложностью, возможностью скрытых ошибок и несоответствий с документацией, так и с субъективными ощущениями дискомфорта в необычной среде разработки. Метод быстрого погружения поможет разрешить эту проблему. Фирма Analog Devices, разработчик архитектуры Blackfin, самим ее наименованием словно подталкивает нас к этому. Давайте вместе попробуем осуществить быстрое погружение с «черным плавником», используя предоставленные нам возможности.

С чего начинать? Разумеется, с выбора экипировки и техники погружения. Простейшим способом пользовались во все времена ловцы жемчуга: камень в руках и тренированные легкие — минимум затрат и максимум доступности «экипировки». Аналогично разработчики могут использовать наиболее дешевую микросхему семейства, самостоятельно спроектировать и изготовить для нее макетную плату, а также (при достаточном опыте) создать собственные средства разработки. Простота, однако, оборачивается в обоих случаях примерно одним и тем же — риском не достичь желаемого результата за разумное время. А ведь на рынке для дайверов имеются не только ласты, маски и акваланги, но даже миниатюрные подводные лодки. Рынок инструментальных средств разработки не менее разнообразен по возможностям представленных на нем продуктов — от простейших демонстрационных плат с бесплатным программным обеспечением до сложных систем проектирования, симуляции и эмуляции, снабженных сопровождаемым коммерческим инструментальным софтом.

Мы ограничимся рассмотрением двух устройств: ADSP-BF533 STAMP и ADSP-BF533 EZ-KIT Lite, поскольку они дают хорошее представление о диапазоне возможностей инструментальных средств для данной архитектуры. Начнем, пожалуй, с последнего, как наиболее широко используемого российскими разработчиками в настоящее время. ADSP-BF533 STAMP рассмотрим в следующих статьях, посвященных освоению цифровых сигнальных процессоров Blackfin. Не касаясь вопроса выбора самой архитектуры Micro Signal Architecture (MSA), анонсированной совместно Intel и Analog Devices в декабре 2000 года (представителем которой и является семейство ADSP-BF5xx), предположим, что ее высокая эффективность, как в универсальных встроенных приложениях, так и в цифровой обработке сигналов, утвердили вас в правильности этого выбора.

Итак, ADSP-BF533 EZ-KIT Lite (рис. 1). Возьмем на себя смелость уподобить это устройство миниатюрной подводной лодке. Это касается не столько цены, сколько возможностей. Приобретая его, вы сможете получить осязаемые результаты буквально в течение одного рабочего дня. Звучит весьма заманчиво. Так ли это на самом деле? Чудо-амфибия для дайвера, конечно, тоже весьма привлекательна, но ведь вопрос в том, насколько сложно научиться ей управлять? Для начала отметим, что рассматриваемое устройство все же намного проще, чем миниатюрную субмарину, вывести из строя. Это связано, конечно, с тем, что «на борту» устройства в большом количестве имеются компоненты, которые не выносят статического электричества. Соблюдая несложные правила предосторожности, вы избежите поломки. Переносите устройство в антистатической упаковке. Перед извлечением устройства обеспечьте свою электронейтральность (при отсутствии заземляющего браслета дотроньтесь рукой до массивного металлического предмета, до батареи отопления, водопроводной или газопроводной трубы, корпуса заземленного прибора). Старайтесь при работе не делать резких движений и без необходимости не прикасайтесь к компонентам устройства. По окончании работы храните устройство в антистатической упаковке.

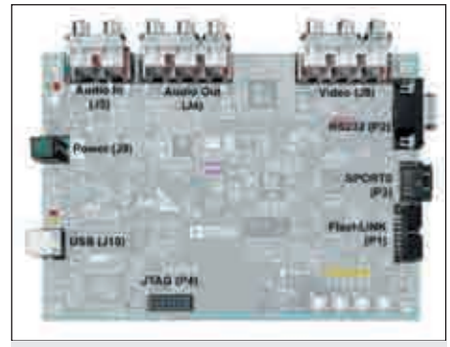

Рис. 1. Отладочная плата ADSP-BF533 EZ-KIT Lite (выделены разъемы для подключения внешних устройств)

В комплекте с устройством (в коробке) находится: кабель для подключения к порту USB 2.0 персонального компьютера, внешний источник питания, компакт-диск с драйверами для ОС Windows 98, Windows XP и с полнофункциональной демонстрационной версией инструментальной среды VisualDSP++. Ограничение лицензии этой версии состоит в одном — размер кода отлаживаемой программы, которую можно загрузить во Flash-память устройства, не может превышать 20 кбайт. Для начала работы этого более чем достаточно. В дальнейшем можно либо подумать о покупке полнофункциональной рабочей версии у дистрибьюторов Analog Devices, либо использовать бесплатную ОС uCLinux for the ADSP-BF53x Processor и такие же бесплатные средства разработки и отладки, предназначенные для написания программ для этой операционной системы. uCLinux — это адаптированная, переработанная с учетом особенностей DSP Blackfin (так называемая «портированная») ОС Linux. Более подробную информацию можно найти на сайте **ht tp:/ /blackfin.uclinux.o rg**. Также можно пообщаться с другими разработчиками на форумах **ht tp:/ /w ww .telesys.r u/ww wboards**. Словом, особых проблем с инструментальными средствами разработчика для данной архитектуры у вас возникнуть принципиально не должно.

Перед работой с устройством установите на своем персональном компьютере VisualDSP++. Ваш компьютер должен иметь разъемы USB 2.0 и около 600 Мбайт свободного места на жестком диске (для версии 4.0 с примерами).

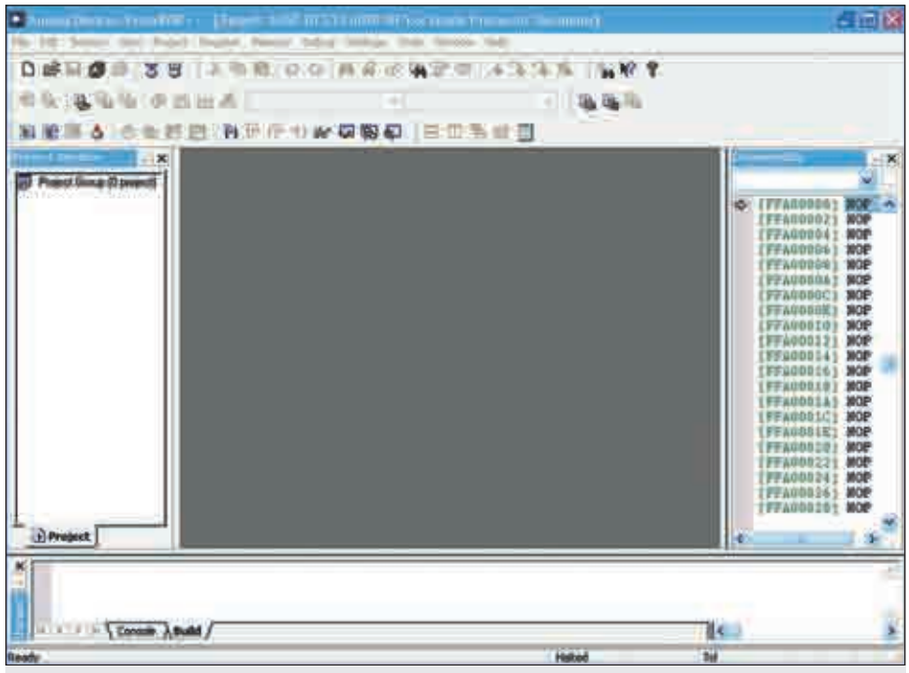

Рис. 2. Внешний вид интегрированной среды разработки VisualDSP++ при запуске в режиме симуляции

Процессор может быть даже Pentium III 500 МГц. Впрочем, некоторые пользователи весьма успешно использовали версию 3.5 даже на Pentium II. Если вы задумали работать с Visual DSP++ в режиме симуляции, о чем далее пойдет речь, то более мощный процессор и большее количество оперативной памяти обеспечат вам лучшее быстродействие. Но здесь многое зависит также и от решаемых задач. Не будем забывать, что процессоры семейства Blackfin работают на частотах до 756 МГц, имеют в своей основе высокопроизводительное RISC-ядро (или даже два у моделей ADSP-BF56x). В режиме симуляции невозможно работать в реальном времени с аналоговыми сигналами и цифровыми интерфейсами, и поэтому мы коснемся этого режима лишь вскользь.

Установка VisualDSP++ на компьютере не должна доставлять особых трудностей.

Рекомендуется производить ее при отсутствии других активных приложений, в особенности других инсталляторов драйверов. После успешного завершения и запуска программы вы должны увидеть картинку, показанную на рис. 2.

Как видим из надписи в заголовке окна, программа запущена в режиме симуляции. Уже сейчас вы имеете возможность работать с инструментальной системой и изучать особенности архитектуры Blackfin. Для этого нет необходимости приобретать какое-либо устройство. Достаточно посетить сайт Analog Devices и скачать там демонстрационную версию — так называемый «тест драйв». Хоть мы и называем эту версию VisualDSP++ демонстрационной, но она ничем не отличается от полнофункциональной, за исключением ограничения по времени (90 дней). Но поскольку у нас речь идет о ADSP-BF533 EZ-KIT Lite, мы проследуем далее, поскольку подробное изучение архитектуры Blackfin не является предметом рассмотрения данной статьи. Режим работы VisualDSP++ можно оперативно менять без переустановки программы через пункт меню Session -> Select Session. А теперь давайте установим для нашего устройства драйвер, поскольку без этого работа в режиме эмуляции, то есть с реальным внешним устройством ADSP-BF533 EZ-KIT Lite попросту невозможна (рис. 3).

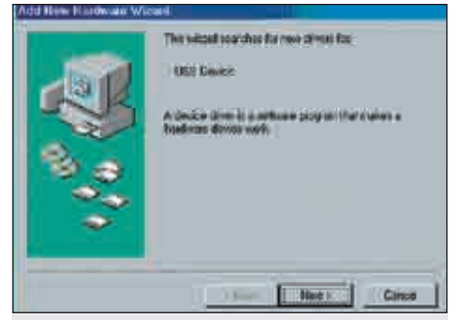

Рис. 3. Одна из фаз автоматической установки драйвера для ADSP-BF533 EZ-KIT Lite

USB обладает приятной для пользователя особенностью — поддержкой технологии plug and play (подключи и работай). Поэтому мы не станем приводить инструкции по установке драйвера. Она достаточно подробно и понятно рассмотрена в руководстве по установке отладочного комплекта ADSP-BF533 EZ-KIT Lite (рис. 4). Достаточно подключить устройство с включенным источником питания к настольному компьютеру или ноутбуку,

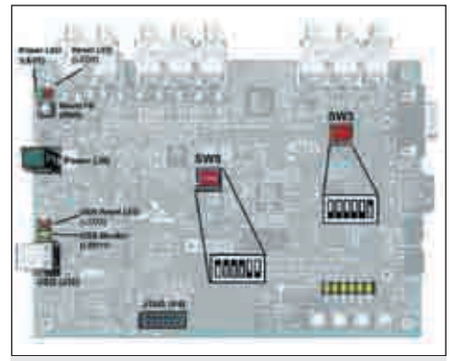

Рис. 4. Отладочная плата ADSP-BF533 EZ-KIT Lite (выделены разъемы питания и USB 2.0, а также устанавливаемые пользователем переключатели режимов и индикаторы)

как все произойдет автоматически. От вас потребуется только адекватная реакция на появляющиеся диалоговые окна с кнопками и другими элементами управления. Наконец, когда драйвер будет успешно установлен, а устройство и хост-компьютер свяжутся по USB, вы сможете судить о загрузке вашей программы по индикатору USB Monitor.

Для успешного завершения первого «быстрого погружения» нам остается немного. Во-первых, не надо отвлекаться на «детали» (ведь на плате много всего интересного, но доверьтесь поставщику, все установлено как раз так, чтобы у вас не возникло проблем, не нужно только ничего сейчас «трогать руками»). Предположим, вы владеете в каком-то минимальном объеме языком программирования Си. Тогда находите в директории <Путь установки>\Blackfin\EZ-KITs\ADSP-BF533\Examples\Blink\C готовый проект BF533 Flags C.dpj (файл проекта) и открывайте его в Visual DSP++ через меню File -> Open -> Project. Двойным щелчком в окне Project на файле main.c загрузите исходный текст главного модуля проекта main.c. Аналогично можно вызвать для редактирования и другие модули проекта: Initialization.c и ISRs.c, а также заголовочный файл BF533 Flags.h. Давайте запустим проект на построение командой меню Project->Build Project. При этом модули проекта компилируются, затем связываются, размещаются и загружаются в память устройства ADSP-BF533 EZ-KIT Lite. Остается запустить программу на выполнение. Мы в двух шагах от сокровищ затонувшего галеона! Выберите пункт меню Debug. Как можно заметить, инструмент позволяет нам многое, но сейчас мы спешим увидеть результат, поэтому выбираем банальное Run. И, о чудо, наша плата оживает! На ней переливаются огни. Но это еще не все. Как и положено в таких случаях, мы можем не только видеть, но и нажимать на кнопочки. Направление бегущих огней при каждом нажатии меняется. Попробуем теперь разобраться с программой. Это совсем несложно, ведь она у нас под рукой.

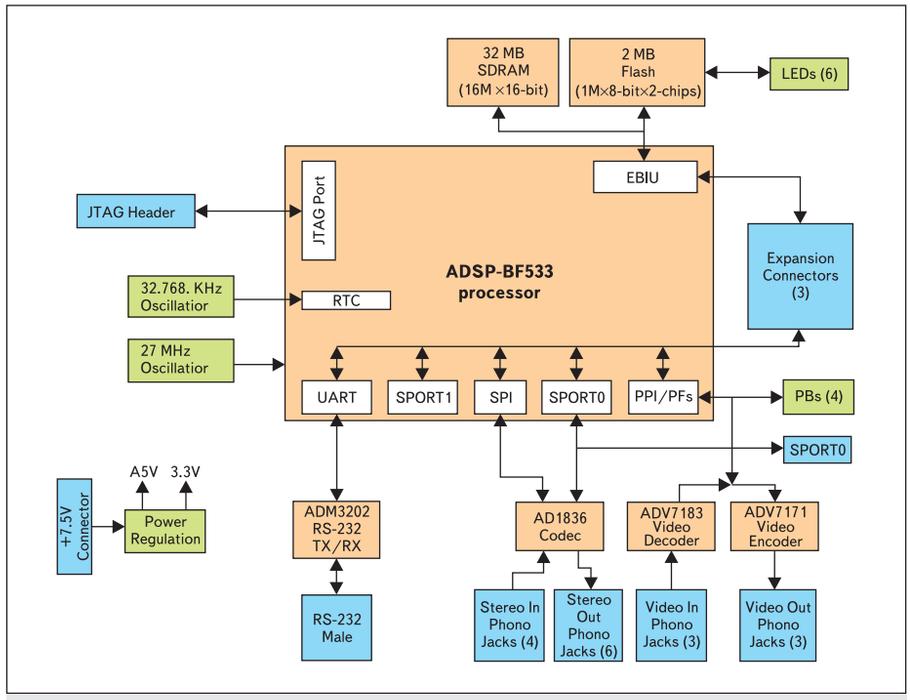

Рис. 5. Функциональная схема платы ADSP-BF533 EZ-KIT Lite

Программы, как серьезные книги, можно изучать постепенно. Но если для понимания сложных текстов могут потребоваться комментарии и обращения к другим книгам, то для понимания программ, связанных с новой архитектурой, всегда необходимы справочные документы от производителей. В нашем случае, поскольку мы имеем дело с платой и с процессором, это:

1. ADSP-BF533 EZ-KIT Lite Evalution System Manual
   файл 179226691ADSP_BF533_EZ_KIT_Lite_Manual_Rev_2.0.pdf;
2. ADSP-BF533 Blackfin Processor Hardware Reference
   файл 61478483BF533_online.pdf.

В первом из них мы почерпнем нужные сведения о регистрах внешних по отношению к ADSP-BF533 устройств, используемых программой, и их адресах, а во втором — то же применительно к внутренним регистрам процессора. Но прежде давайте разберемся со структурой нашей программы. Она состоит из следующих модулей:

```
BF533 Flags.h
Initialization.c
ISRs.c
main.c
```

Само название main.c говорит нам, что это главный программный модуль.

```
#include «BF533 Flags.h»
#include «ccblkfn.h»
#include <sysreg.h>

// flag indicating direction of moving light (toggled in FlagA ISR)
short sLight_Move_Direction = 0;
void main(void)
{
        sysreg_write(reg_SYSCFG, 0x32);  //Initialize System
                                         //Configuration Register
        Init_Flags();
        Init_Timers();
        Init_EBIU();
        Init_Flash();
        Init_Interrupts();
        while(1);
}
```

На первый взгляд, программа примитивна. В конце ее процессор находится в бесконечном цикле, а все наблюдаемые нами полезные действия осуществляются по прерываниям. Поскольку мы наблюдаем строго упорядоченные во времени изменения состояний светодиодов, то можем предположить, что они синхронизированы с аппаратным таймером. Естественное любопытство вызывает reg_SYSCFG, тем более что безликое 0x32 свидетельствует о плохом стиле авторов программы. Но, с другой стороны, они очень эффектно вынесли в верхнюю часть sLight_Move_Direction. Эта переменная, скорее всего, и есть тот флаг, который задает направление перемещения «бегущих огней» Init_Timers() и Init_Interrupts() — это понятно и хорошо согласовывается с нашими предположениями относительно таймера и прерываний, но что такое EBIU, и для чего Init_Flags? Ведь флага достаточно одного! Не так уж все просто, как кажется на первый взгляд, и нам сейчас придется «включить голову». Да, мы с вами быстро погрузились, но ведь это не было самоцелью? Самое интересное — это как раз то, что скрывают морские глубины, и теперь мы начинаем их исследовать всерьез. Попытаемся действовать простым, но эффективным методом «разделяй и властвуй». Выясним, где проходит граница между ADSP-BF533 и остальными, используемыми нашей программой устройствами.

Рассматривая рис. 5, с удивлением обнаруживаем, что наши светодиоды подключены не напрямую к портам вывода ADSP-BF533, а через Flash-память! А со стороны самого процессора они поддерживаются все еще пока для нас загадочным EBIU. Но зато становится ясно, почему разнесены в разные функции Init_EBIU() и Init_Flash(). Совсем пока неочевидно, что кнопка, изменяющая направление «бегущих огней», имеет связь с зеленым прямоугольником в правой части PBs (4). Но тогда взгляните еще раз на саму плату (рис. 6) с выделенными светодиодами и кнопками и на таблицу 1.

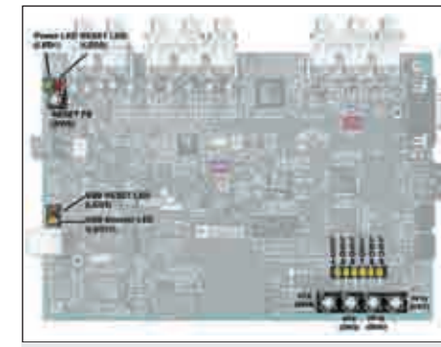

Рис. 6. Светодиоды и кнопки на ADSP-BF533 EZ-KIT Lite

Таблица 1. Программируемые флаги-переключатели

Processor Programmable Flag Pin	Push Button Reference Designator
PF8	SW4
PF9	SW5
PF10	SW6
PF11	SW7

Почему они так странно называются? Для этого надо быть знатоком истории создания Blackfin, а это скорее вопрос к разработчикам. Для нас лишь важно, что это действительно часть самого процессора, а не других устройств платы, и то, главным образом, в связи с обработкой прерываний, но об этом далее. Теперь самое время заглянуть в файл Initialization.c. Там мы находим реализацию интересующей нас функции Init_Flags():

```
void Init_Flags(void)
{
        *pFIO_INEN          = 0x0100;
        *pFIO_DIR           = 0x0000;
        *pFIO_EDGE          = 0x0100;
        *pFIO_MASKA_D       = 0x0100;
}
```

Хочется ругать этого безымянного автора программы за его аптекарское взвешивание меда и дегтя в своем продукте (первого, к сожалению, не бочка, а второго — отнюдь не ложка).

Опять «безликие константы» в правых частях операторов присваивания. Кстати, по-

чему собственно это все же вывод в регистры, а не в ячейки обычной памяти? Ответ хотя и не лежит на поверхности, но очевиден, так как сами указатели определены в файле cdefBF532.h, например:

```
#define PFIO_INEN (volatile unsigned short *) FIO_INEN
```

Поиски же FIO_INEN приведут нас к файлу defBF532.h, а в нем:

```
#define FIO_INEN      0xFFC0073C      // Flag Input Enable Reg
```

Теперь, обратившись к ADSP-BF533 Blackfin Processor Hardware Reference, находим биты регистра разрешения входов флагов FIO_INEN (рис. 7).

То есть, 0x0100 — это PF8 Input Enable. Остальные 3 кнопки SW5-SW7 в нашей программе не действуют, так что все хорошо согласуется с опытом. Быть может, авторы этой программы даже преследовали цель обучения. Ведь запоминается лучше то, что дается с трудом.

Мы можем проверить теперь, правильный ли адрес. Но это, конечно, излишне, поскольку софт все же коммерческий. Лучше перейти к выяснению того, за что отвечают другие строки в Init_Flags. В defBF532.h находим:

```
#define FIO_DIR              0xFFC00730 // Peripheral Flag Direction
#define FIO_EDGE             0xFFC00738 // Flag sources Sensitivity
Register
#define FIO_MASKA_D          0xFFC00738 // Flag Mask Interrupt D
Register
```

Очевидно первые два отвечают за направление (то есть ввод или вывод через бит порта) и срабатывание по уровню или перепаду прерывания. А вот последний отвечает за разрешение прерывания. С портами ввода-вывода, то есть флагами, туман немного рассеялся. Давайте теперь обратимся к загадочному обитателю EBIU. Можно было бы пойти тем же путем, но есть и более простое решение: «Правка->Найти» в MS Word. Впрочем, то, что EBIU означает External Bus Interface Unit, проницательный читатель мог уяснить из комментария в начале реализации функции void Init_EBIU(void) в файле Initialization.c, а вот уяснение принципов работы интерфейса внешней шины заняло бы довольно много времени.

Давайте сосредоточимся на системе прерывания. Это намного важнее, поскольку без этих знаний невозможно написание собственных драйверов ввода–вывода. А для чего они нужны, специалистам в области DSP объяснять, кажется, излишне. Ведь основная цель использования ADSP-BF533 — именно задачи цифровой обработки сигналов, а прочие задачи — фоновые. Фоновые задачи должны выполняться по прерываниям. Программы-обработчики прерываний (хандлеры) должны каким-то образом быть привязаны к векторам прерываний.

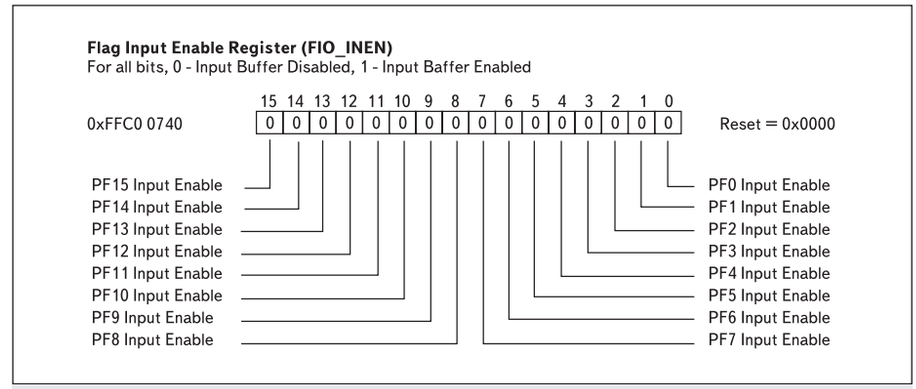

Рис. 7. Биты регистра разрешения входов флагов FIO_INEN

Файл ISRs.c имеет весьма красноречивое название (Interrupt Service Routines). И действительно, в нем сосредоточены все обработчики прерываний нашей программы: EX_INTERRUPT_HANDLER(Timer0_ISR) — обработчик прерывания от таймера и EX_INTERRUPT_HANDLER(FlagA_ISR) — обработчик прерывания от кнопки. Где еще встречаются Timer0_ISR и FlagA_ISR в нашей программе? Конечно, в реализации void Init_Interrupts(void) в файле Initialization.c. Выглядит это так:

```
// assign ISRs to interrupt vectors
 register_handler(ik_ivg11, Timer0_ISR);     // Timer0 ISR -> IVG 11
 register_handler(ik_ivg12, FlagA_ISR);      // FlagA ISR -> IVG 12
```

Полная абракадабра, неправда ли? Что такое IVG? И откуда взялось и что означает register_handler? В наших руках есть очень убедительное исследовательское орудие — торпедный аппарат (ведь мы же на мини-субмарине). А если серьезно, все та же функция поиска дает нам:

```
ex_handler_fn register_handler (interrupt_kind, ex_handler_fn)
```

Это указатель на функцию, обрабатывающую прерывание заданного типа, то есть вектор прерывания в таблице векторов событий EVT (Events Vector Table). Почему не таблица векторов прерываний? Потому что ADSP-BF53x — на самом деле довольно сложная архитектура, поддерживающая ОС реального времени (РВ), в том числе защиту памяти и несколько режимов работы ядра. Но не будем сейчас углубляться в эту тему. Достаточно принять к сведению, что понятие events шире, чем interrupts, и контроллер прерываний, точнее System Interrupt Controller (SIC), является частью Core Event Controller (CEC), а прерывания — лишь один из возможных типов событий ядра. Для прерываний от периферийных устройств выделены семь векторов IVG13-IVG7. Периферийных устройств и источников прерываний значительно больше, поэтому возникает необходимость динамической привязки к вектору. Кроме того, наивысший приоритет всегда имеет IVG0, а наименьший — IVG15. В нашем случае таймер имеет более высокий приоритет, чем кнопка (попробуйте держать ее нажатой, и убедитесь, что интервалы переключения светодиодов сохраняются неизменными). Следы такой привязки мы наблюдаем в строке:

```
*pSIC_IAR2 = 0xffff5ff4;
```

А почему 5 вместо 12, и 4 вместо 11 (рис. 8)? Все в соответствии с таблицей 2.

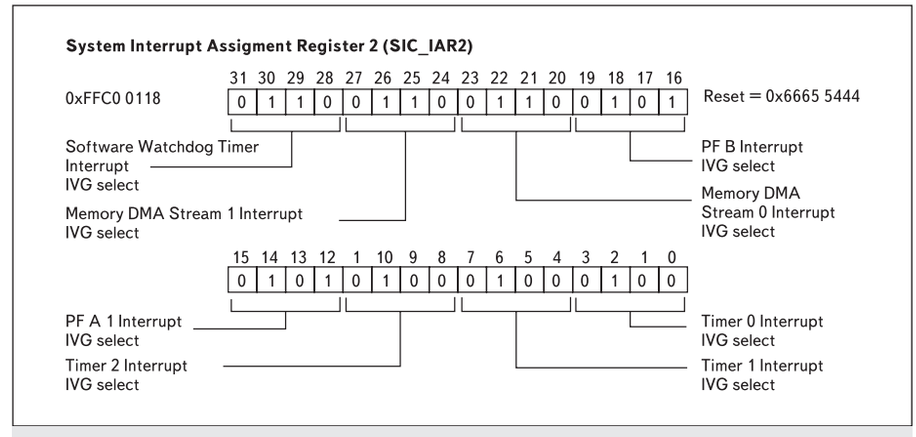

Рис. 8. Биты регистра назначенных прерываний SIC_IAR2

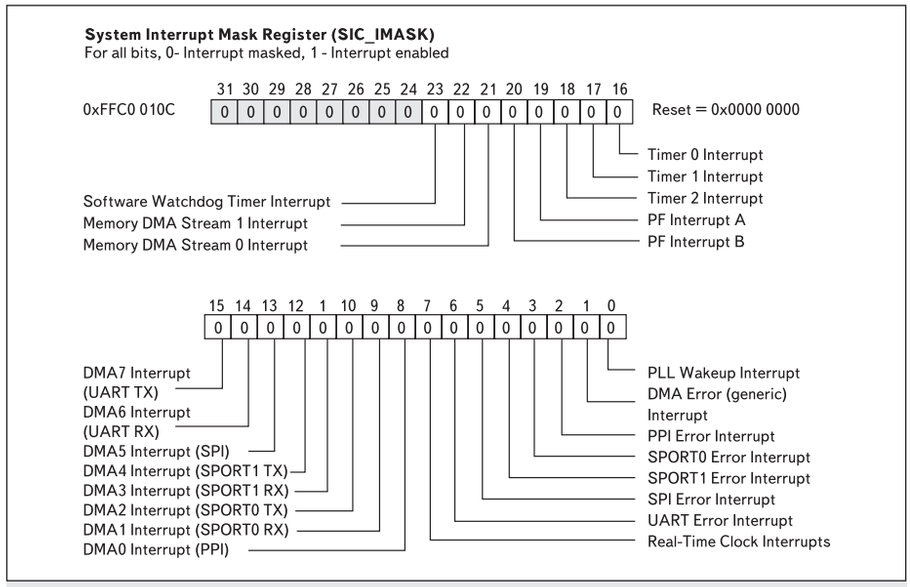

Рис. 9. Биты регистра маскирования системных прерываний SIC_IMASK

И, наконец, прерывания должны быть разрешены не только глобально, но и индивидуально для каждого вектора:

```
*pSIC_IMASK = 0x00090000;
```

Как видно из рис. 9, разрешены как раз Timer 0 Interrupt и PF Interrupt A.

Теперь мы можем попытаться написать наш первый драйвер для работы с последовательным портом по прерываниям. Поскольку процесс будет описан весьма подробно, оставим это для следующей статьи. Если у вас возникли вопросы, замечания, уточнения или пожелания, направляйте их авторам по адресам электронной почты, указанным в начале статьи.

Таблица 2. Идентификаторы прерываний в регистре SIC_IAR

General-purpose Interrupt	Value in SIC_IAR
IVG7	0
IVG8	1
IVG9	2
IVG10	3
IVG11	4
IVG12	5
IVG13	6
IVG14	7
IVG15	8

Литература

1. ADSP-BF533 EZ-KIT Lite Evaluation System Manual. **ww w .analog.c om/UploadedFiles/Associated_Docs/68863602ADSP_BF533_EZ_KIT_Lite_Manual_Rev_3.0.pdf**
2. ADSP-BF533 Blackfin Processor Hardware Reference. **w ww .analog.c om/UploadedFiles/Associated_Docs/56894137133580215**