Разработка проекта микроконтроллера 8051s на основе IP-ядер корпорации Microsemi.
Часть 4. Использование прерываний

Дмитрий ИОФФЕ
support@actel.ru
Андрей МАКСИМОВ
maksimov@actel.ru

Это четвертая статья из цикла, посвященного применению микропроцессорного IP-ядра 8051s для ПЛИС корпорации Microsemi. В первой части статьи [1] было рассмотрено построение аппаратной части системы на основе 8051s с использованием IP-ядер, поставляемых в составе САПР Libero. Во второй части [2] приведено описание ядра 8051s для программиста. В третьей части [3] мы написали и отладили простейшую программу для микроконтроллера. Сегодня мы рассмотрим работу с прерываниями и подключение сторожевого таймера.

Таймер

Источником прерываний у нас будет таймер. Он реализован как IP-ядро CoreTimer [4] из каталога Libero, которое мы включили в состав нашего проекта во время формирования аппаратной части [1]. Его устройство показано на рис. 1.

Перед началом работы следует загрузить в вычитающий счетчик нужное число. Таймер может работать в двух режимах: непрерывном (continuous) и однократном (one-shot). Если задан непрерывный режим (он включен по умолчанию), то по достижении нулевого значения вычитающего счетчика отсчет начинается сначала. В однократном режиме при достижении нуля счетчик останавливается. Если разрешена генерация прерываний, то при достижении нуля CoreTimer вырабатывает сигнал прерывания. Чтобы сбросить флаг прерывания, нужно записать любое число в регистр очистки прерывания (Interrupt Clear Register). В любой момент времени число в вычитающем счетчике можно прочитать из регистра текущего значения (Current Value Register). Работу таймера можно запретить или разрешить через бит в регистре управления таймером (Timer Control Register).

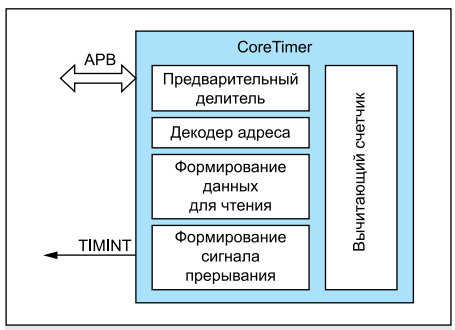

Рис. 1. Устройство IP-ядра CoreTimer

Можно маскировать прерывание, записывая ноль в бит разрешения прерывания (Interrupt Enable Bit) в регистре управления таймером. При этом текущее состояние прерывания (без учета маскирования) можно прочитать из регистра состояния TimerRIS.

Вычитающий счетчик тактируется импульсами тактовой частоты шины APB (PCLK). Сигнал разрешения счета вырабатывается модулем предварительного деления (prescale unit). Коэффициент деления задается через регистр управления таймером и может принимать одно из следующих значений: 2, 4, 8, 16, 32, 64, 128, 256, 512 или 1024.

После сброса CoreTimer находится в следующем состоянии:
- работа счетчика запрещена;
- флаг прерывания сброшен;
- в регистр счетчика загружен ноль;
- установлен непрерывный режим;
- предварительный делитель делит входную частоту на два.

Таблица 1. Регистры IP-ядра CoreTimer

Смещение	Чтение или запись	Разрядность	Значение после сброса	Имя	Описание
0x00	Чтение и запись	16 или 32	0x0000 или 0x00000000	TimerLoad	Загружаемое значение
0x04	Чтение	16 или 32	0xFFFF или 0xFFFFFFFF	TimerValue	Текущее значение
0x08	Чтение и запись	3	0x0	TimerControl	Регистр управления
0x0C	Чтение и запись	4	0x0	TimerPrescale	Установка предварительного делителя
0x10	Запись	–	–	TimerIntClr	Очистка прерывания
0x14	Чтение	1	0x0	TimerRIS	Реальное состояние прерывания
0x18	Чтение	1	0x0	TimerMIS	Замаскированное состояние прерывания

Таблица 2. Биты регистра управления

Биты	Имя	Чтение или запись	Назначение
31:3	–	–	Не используются, читаются нули
2	Режим таймера	Чтение и запись	0 — непрерывная работа (по умолчанию) 1 — однократный запуск
1	Разрешение прерывания	Чтение и запись	0 — прерывание от таймера запрещено (по умолчанию) 1 — прерывание разрешено
0	Разрешение работы таймера	Чтение и запись	0 — работа запрещена (по умолчанию) 1 — работа разрешена

Таблица 3. Регистр установки предварительного делителя

Биты	Имя	Чтение или запись	Назначение
31:4	–	–	Не используются, читаются нули
3:0	Предварительный делитель	Чтение и запись	Поле задания коэффициента деления. Значения: 0000 — 2 (по умолчанию) 0001 — 4 0010 — 8 0011 — 16 0100 — 32 0101 — 64 0110 — 128 0111 — 256 1000 — 512 1001 и все остальные значения — 1024

Разрядность вычитающего счетчика задается при конфигурировании IP-ядра и может быть равна 16 или 32 битам.

Регистр реального состояния прерывания без учета маскирования (Raw Interrupt Status Register, TimerRIS) содержит «1», если прерывание запрашивается, и «0» — если не запрашивается. Аналогично ведет себя регистр прерывания с учетом маскирования (Interrupt Status Register, TimerMIS).

Доработка программы микроконтроллера

Добавим в нашу программу возможность работы с прерываниями от таймера. Для этого нам нужно будет в начале программы инициализировать таймер, а в основном цикле программы опрашивать флаг прерывания. Обнаружив запрос прерывания, мы должны будем зажечь или погасить выбранный светодиод и затем сбросить запрос прерывания.

Сначала, как и раньше, мы будем моделировать работу нашей системы в Modelsim. Поэтому сделаем пока период работы таймера небольшим, чтобы сократить время моделирования. Мы легко сможем установить минимальный коэффициент предварительного делителя, записав ноль в четырехразрядный регистр TimerPrescale. А вот перед загрузкой 32-разрядного вычитающего счетчика нам нужно будет вспомнить порядок записи 32-разрядных данных на шину APB, описанный в [2]. Поскольку 8051s — восьмиразрядный процессор, непосредственно работать с 32-разрядными числами он не может. Для решения этой проблемы разработчики IP-ядра 8051s добавили к регистрам общего назначения 8051 так называемые X-регистры: три для чтения и три для записи. X-регистры для записи называются XWB1, XWB2 и XWB3. Для записи 32-разрядного числа сначала нужно записать три его старших байта в X-регистры, а затем записать младший байт по соответствующему адресу на шине APB. После этого на шине выходных данных APB появится 32-разрядный код. Временно запишем в вычитающий счетчик небольшое число, например 200.

После задания периода работы таймер можно включать. Для этого нужно записать единицу в бит 0 регистра управления таймера. В его бит 1 также запишем единицу, чтобы разрешить прерывания, а в бите 2 оставим 0, чтобы задать непрерывный режим работы таймера.

Итак, нам нужно сделать в программе следующие изменения.

1. Описать регистры IP-ядра CoreTimer:
- __xdata at TimerAddr unsigned char TimerLoad;
- __xdata at TimerAddr + 0x08 unsigned char TimerControl;
- __xdata at TimerAddr + 0x0C unsigned char TimerPrescale;
- __xdata at TimerAddr + 0x10 unsigned char TimerIntClr;
- __xdata at TimerAddr + 0x14 unsigned char TimerRIS;

Эти записи при помощи ключевого слова xdata сообщают компилятору, что регистры шины APB лежат во внешней памяти данных.

2. Описать X-регистры:
- __data at 0x9A unsigned char XWB1;
- __data at 0x9B unsigned char XWB2;
- __data at 0x9C unsigned char XWB3;

```
#include "..\System8051s_hw_platform.h"

#define APB_base 0xF000U //начало адресного пространства APB

/*Работа с GPIO:*/
#define GPIO_addr APB_base + COREGPIO_0 //адрес блока GPIO
__xdata at GPIO_addr + 0xA0 unsigned char dataout;
__xdata at GPIO_addr + 0x90 unsigned char datain;
#define jumper 0x01 //вход опроса переключателя
#define LED_0 0x01 //нулевой светодиод
#define LED_1 0x02 //первый светодиод
unsigned char inreg;
unsigned char outreg;
unsigned char temp;

/*Работа с таймером*/
#define TimerAddr APB_base + CORETIMER_0 //адрес таймера
__xdata at TimerAddr unsigned char TimerLoad;
__xdata at TimerAddr + 0x08 unsigned char TimerControl;
__xdata at TimerAddr + 0x0C unsigned char TimerPrescale;
__xdata at TimerAddr + 0x10 unsigned char TimerIntClr;
__xdata at TimerAddr + 0x14 unsigned char TimerRIS;

/*X-регистры для записи в старшие 24 разряда шины APB*/
__data at 0x9A unsigned char XWB1;
__data at 0x9B unsigned char XWB2;
__data at 0x9C unsigned char XWB3;

void main ()
{
   outreg = 0x00;
//Инициализация таймера:
   XWB3 = 0;
   XWB2 = 0;
   XWB1 = 0;
   TimerLoad = 200; //записали 32-разрядное число
   TimerPrescale = 0;
   TimerControl = 0x03;
//Основной цикл программы:
   for (;;) {
   //Опрос флага прерывания:
      if (TimerRIS == 1)
         {temp = outreg & LED_1;
         if (temp == 0) outreg = outreg | LED_1;
         else outreg = outreg & ~LED_1;
         dataout = outreg;
         TimerIntClr = 0; /*сброс прерывания*/};
   //Отслеживание джампера:
      inreg = datain;
      if ((inreg & jumper) == 0)
         outreg = outreg & ~LED_0;
      else
         outreg = outreg | LED_0;
      dataout = outreg;
   }
}
```

Листинг. Программа, реализующая все учебные планы по управлению светодиодами

Ключевое слово data сообщает компилятору, что X-регистры находятся в непосредственно адресуемой внутренней памяти данных. К сожалению, непосредственно использовать файл reg51.h не получится.

3. Добавить в начале программы код инициализации таймера.
4. Написать в основном цикле программы опрос флага прерывания и реакцию на прерывание.

Листинг содержит полный текст программы, которая зажигает и гасит один светодиод вслед за движениями микропереключателя, а другой — по сигналу таймера.

Моделирование работы со светодиодами

Откомпилируем нашу программу в среде SoftConsole, преобразуем полученный шестнадцатеричный код в текст модуля ПЗУ на VHDL, а затем выполним генерацию проекта, как описано в третьей части статьи [3]. Перед генерацией проекта следует исправить ошибку, допущенную авторами в [1]: в IP-ядре CoreGPIO необходимо включить все 32 линии ввода/вывода, но вывести из системы по-прежнему только те, что нужны для работы. В противном случае CoreGPIO поведет себя при моделировании непредсказуемо.

Файл тестбенча оставим без изменений, так как новых внешних сигналов у нас не появилось. Запустим среду Modelsim и выберем из поля Instance для отображения следующие сигналы:
- сигналы тестбенча (корневой узел tb_inout): sysclk, nsysreset и jumper;
- входы и выходы нашего проекта (system8051s_0):
  - gpio_in — вход микропереключателя;
  - gpio_out(0) — выход светодиода, управляемого микропереключателем;
  - gpio_out(1) — выход светодиода, управляемого от таймера;
- сигналы таймера (разворачиваем system8051s_0 и выбираем сигналы группы coretimer_0):
  - countiszero — момент появления нуля в вычитающем счетчике;
  - intenable — сигнал разрешения прерываний;
  - timint — флаг запроса прерываний.

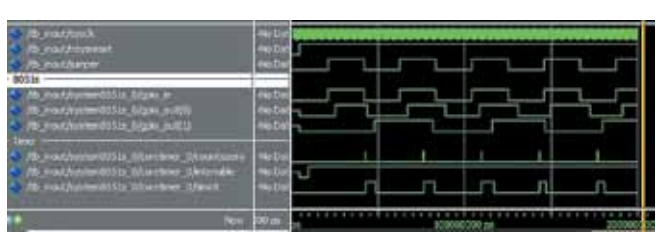

Рис. 2. Результат моделирования управления светодиодами

Таблица 4. Регистры IP-ядра CoreWatchdog

Смещение	Чтение или запись	Разрядность	Значение после сброса	Имя	Описание
0x00	Чтение и запись	16 или 32	0x0000 или 0x00000000	WdogLoad	Значение, загружаемое в счетчик
0x04	Чтение	16 или 32	0xFFFF или 0xFFFFFFFF	WdogValue	Текущее значение счетчика
0x08	Чтение и запись	5	0x00	WdogControl	Управляющий регистр
0x0C	Запись	–	–	WdogRefresh	Регистр обновления

Не забудем сохранить полученный набор сигналов в отдельном файле, например, под именем wave_timer.do.

В результате моделирования, если у нас все получилось правильно, мы увидим примерно такую картину (рис. 2).

На рис. 2 видно, что управление светодиодом от микропереключателя после всех доработок не пострадало: входной сигнал jumper попадает на вход gpio_in модуля GPIO, и затем с некоторой задержкой переключается выход gpio_out(0), к которому подключен светодиод.

В группе временных диаграмм Timer мы можем наблюдать работу таймера. Окончание инициализации заметно по разрешению прерываний intenable, и после этого мы видим появление периодического сигнала countiszero, который принимает высокий уровень при нулевом значении вычитающего счетчика. Этот сигнал устанавливает флаг запроса прерывания timint, который через некоторое время сбрасывается программно. С некоторой задержкой после установки этого флага переключается сигнал gpio_out(1), который управляет вторым светодиодом.

Программирование ПЛИС и проверка управления светодиодами

Перед тем как использовать наш проект для прошивки ПЛИС, нам осталось увеличить период работы таймера так, чтобы мы могли наблюдать мигание светодиода. Пусть период свечения светодиода будет равен 1 с, тогда период работы таймера должен составлять половину секунды. Мы знаем, что системная тактовая частота равна 48 МГц: она делится на четыре, и результат используется для тактирования ядра 8051s и шины APB. В свою очередь, вычитающий счетчик переключается с частотой вдвое меньшей, если мы не изменим коэффициент предварительного деления. Получается, что нам нужно загружать в вычитающий счетчик число 3 000 000. В шестнадцатеричном виде оно будет выглядеть как 0x2DC6C0. Тогда участок программы, который загружает код в вычитающий счетчик, будет иметь вид:

```
XWB3 = 0;
XWB2 = 0x2D;
XWB1 = 0xC6;
TimerLoad = 0xC0;   // записали 32-разрядное число
```

Компилируем проект и загружаем его в ПЛИС, как описано в [3].

Авторы желают вам, уважаемый читатель, чтобы у вас получилось так же: один светодиод мигает с частотой 1 Гц, а другой загорается и гаснет согласно движению микропереключателя.

Сторожевой таймер

Для реализации сторожевого таймера корпорация Microsemi предлагает IP-ядро CoreWatchdog [5]. Как и в только что рассмотренном CoreTimer, основой ядра является загружаемый вычитающий счетчик, разрядность которого (32 или 16 разрядов) выбирается при конфигурировании. Здесь мы также найдем предварительный делитель, который управляется точно так же, теми же кодами.

Если не обновить содержимое вычитающего счетчика, пока он не досчитал до нуля, на выходе CoreWatchdog появляется сигнал WATCHDOGRES, который сбросит микропроцессорное ядро 8051s, и выполнение программы начнется сначала (рис. 3). Для обновления достаточно записать любое число в регистр WdogRefresh. В таблицах 4 и 5 приведена информация, достаточная для работы с CoreWatchdog.

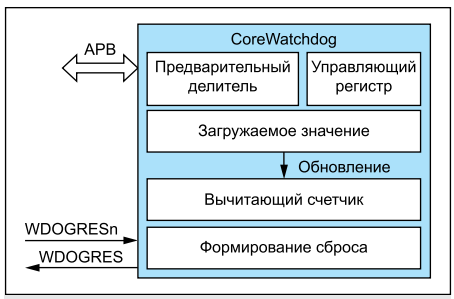

Рис. 3. Функциональная схема IP-ядра CoreWatchdog

Таблица 5. Биты регистра управления

Биты	Имя	Чтение или запись	Назначение
31:5	–	–	Не используются
4	Watchdog Enable	Чтение и запись	0 — сторожевой таймер выключен (по умолчанию) 1 — включен
3:0	Предварительный делитель	Чтение и запись	Поле задания коэффициента деления. Значения: 0000 — 2 (по умолчанию) 0001 — 4 0010 — 8 0011 — 16 0100 — 32 0101 — 64 0110 — 128 0111 — 256 1000 — 512 1001 и все остальные значения — 1024

Управление сторожевым таймером

Отладим работу со сторожевым таймером при помощи Modelsim. Для этого временно опять будем загружать в вычитающий счетчик CoreTimer число 200.

Введем в наш код описание регистров сторожевого таймера:

```
#define WatchdogAddr APB_base + COREWATCHDOG_0 //адрес таймера
__xdata at WatchdogAddr unsigned char WatchdogLoad;
__xdata at WatchdogAddr + 0x08 unsigned char WatchdogControl;
__xdata at WatchdogAddr + 0x0C unsigned char WatchdogRefresh;
```

После сброса сторожевой таймер выключен. Добавим в начале программы код инициализации сторожевого таймера:

```
XWB3 = 0;
XWB2 = 0x00;
XWB1 = 0x00;
WatchdogLoad = 0xFF;
WatchdogControl = 0x10;
```

А обновление сторожевого таймера пока делать не будем. Запустим моделирование и добавим отображение временных диаграмм сигналов CoreWatchdog: состояние вычитающего счетчика count и выходной сигнал сброса wdogres (рис. 4).

Мы видим, что без обновления сторожевой таймер вырабатывает сигнал сброса

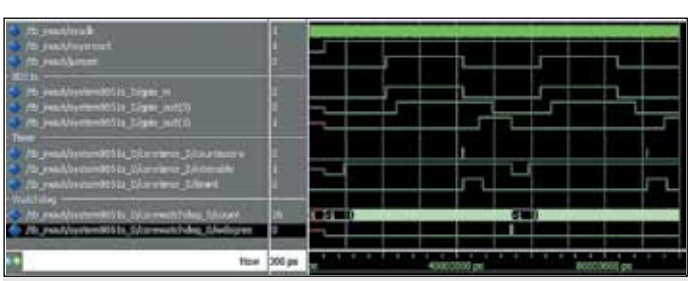

Рис. 4. Моделирование работы сторожевого таймера без обновления

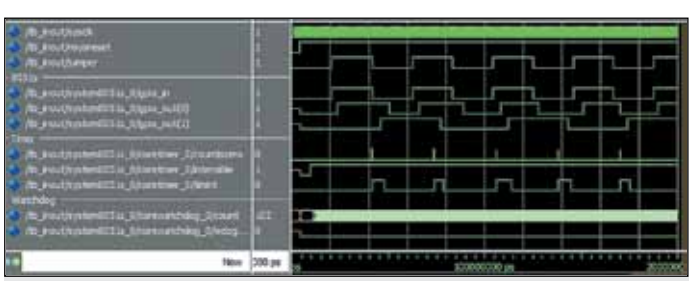

Рис. 5. Моделирование работы сторожевого таймера с обновлением

wdogres, и наша система перезапускается по этому сигналу. Заметим, что во время инициализации мы загрузили в вычитающий счетчик сторожевого таймера число 255 (0xFF), которое несколько больше числа 200 для вычитающего счетчика CoreTimer. Добавим в код обработки прерывания строку для обновления сторожевого таймера:

```
WatchdogRefresh = 0x00;
```

Результат моделирования представлен на рис. 5.

Здесь сигнал wdogres не вырабатывается, и наша система работает так же, как и раньше.

Если мы теперь приведем в соответствие коды, загружаемые в вычитающие счетчики IP-ядер CoreTimer и CoreWatchdog, то наши светодиоды будут вести себя по-прежнему и ничего нового мы не увидим. Предлагаем читателю проверить это самостоятельно.

Выводы

В четвертой части статьи мы освоили работу с IP-ядрами таймера CoreTimer и сторожевого таймера CoreWatchdog корпорации Microsemi, а также использование прерываний от таймера.

Файлы учебного проекта можно найти в Интернете по адресу [6].

Литература

1. Иоффе Д., Максимов А. Разработка проекта микроконтроллера 8051s на основе IP-ядер корпорации Microsemi. Ч. 1 // Компоненты и технологии. 2014. № 1.
2. Иоффе Д., Максимов А. Разработка проекта микроконтроллера 8051s на основе IP-ядер корпорации Microsemi. Ч. 2. IP-ядро 8051s для программиста // Компоненты и технологии. 2014. № 3.
3. Иоффе Д., Максимов А. Разработка проекта микроконтроллера 8051s на основе IP-ядер корпорации Microsemi. Ч. 3. Первая программа для микроконтроллера // Компоненты и технологии. 2014. № 5.
4. CoreTimer — www.microsemi.com
5. CoreWatchdog — www.microsemi.com
6. http://www.actel.ru/catalog/8051s-part4.zip