KiT#59(6).qxd 6/27/2006 1:42 PM Page 56

Продолжение. Начало в № 4`2006

Проектирование на программируемых системах на кристалле PSoC Cypress.

Часть 2. Цифровая конфигурируемая периферия

Дмитрий КИЛОЧЕК
Dmitry.Kilochek@macrogroup.ru

Продолжая рассказ, посвященный проектированию на PSoC, рассмотрим работу с цифровой конфигурируемой периферией.

Цифровая подсистема PSoC позволяет реализовать необходимые разработчику периферийные функции, задав конфигурацию цифровых блоков PSoC и их взаимосвязь.

Конфигурируемая цифровая подсистема позволяет реализовать счетчики, таймеры, ШИМ, генераторы случайных последовательностей, блоки CRC, а также контроллеры интерфейса SPI, инфракрасный приемопередатчик и UART. К цифровой периферии PSoC относятся также контроллер шины I²C, умножитель и блок прореживания (Decimator), используемый для создания АЦП. Однако так как эти блоки являются фиксированными аппаратными модулями, то в структуре PSoC они отнесены к системным ресурсам, а не к цифровой подсистеме.

Цифровая подсистема состоит из четырех компонентов (рис. 1):

- глобальные цифровые связи (Global Digital Interconnect, GDI);
- связи в массиве цифровых блоков (Array Digital Interconnect, ADI);
- связи внутри ряда цифровых блоков (Row Digital Interconnect, RDI);
- цифровые базовые (DBB) и коммуникационные (DCB) блоки.

Глобальные цифровые связи состоят из четырех 8-разрядных шин — двух входных и двух выходных. Входные и выходные шины подразделяются на четные (GIE, GOE) и нечетные (GIO, GOO), в зависимости от номеров портов ввода-вывода, к которым эти шины подсоединены. Глобальные шины предназначены для передачи сигналов от контактов портов в цифровые блоки, либо для выдачи на контакты сигналов от цифровых блоков или от источников глобальных частот. Входные и выходные шины связаны друг с другом через двунаправленные тристабильные буферы (рис. 2).

Таким образом, есть возможность внутри микросхемы передавать входные сигналы на выход и наоборот, сигналы с выходных шин передавать на входные. Каждая линия глобальной шины имеет специальный триггер-хранитель (keeper). Этот хранитель после сброса системы устанавливает состояние линии глобальной шины в '1', а в процессе работы сохраняет на линии последнее поданное на нее значение, что предотвращает возникновение неопределенного состояния.

За подсоединение глобальных шин отвечают регистры **PRTxGS**. Выдачей сигналов глобальных частот на выходную шину GOE управляет регистр **OSC_GO_EN**. Буферами, связывающими входные и выходные шины,

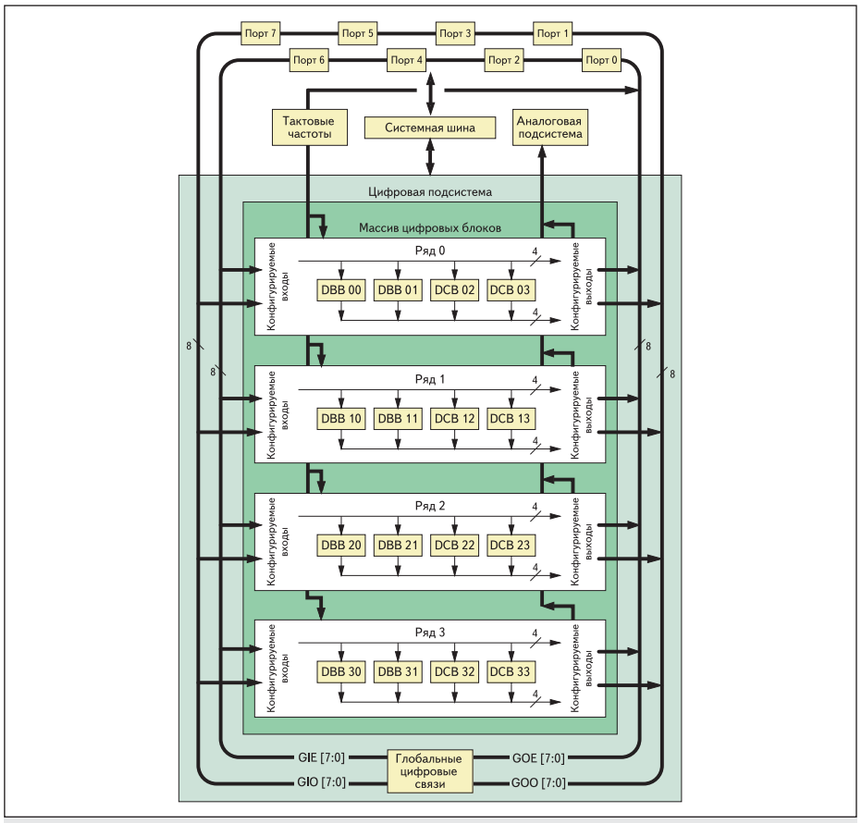

Рис. 1. Блок-схема цифровой подсистемы PSoC

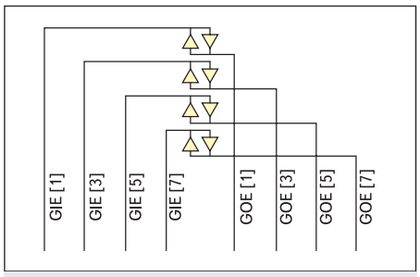

Рис. 2. Входные и выходные шины

KiT#59(6).qxd 6/27/2006 1:42 PM Page 57

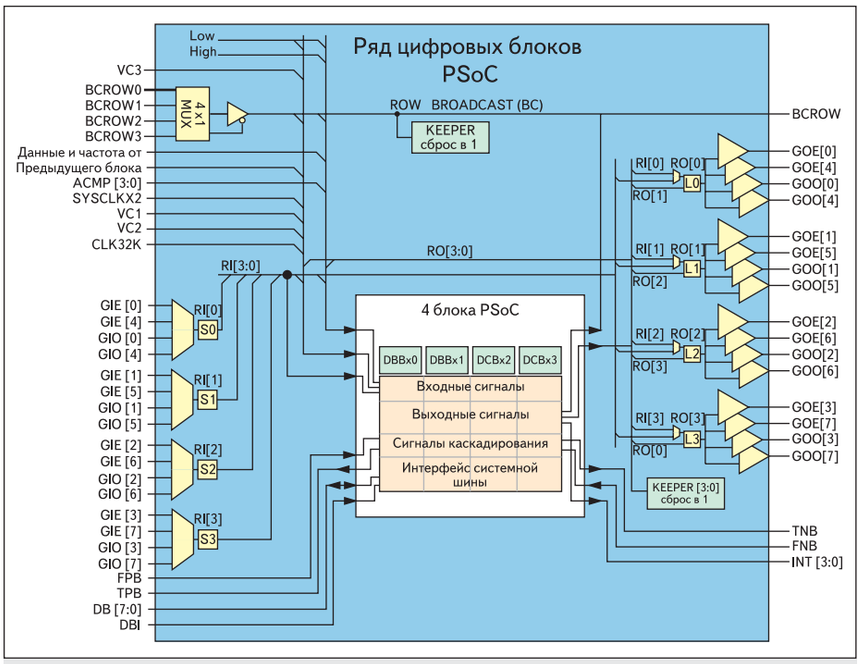

Рис. 3. Ряд цифровых блоков PSoC

управляют регистры **GDI_x_IN** (соединение вход-выход) и **GDI_x_OU** (соединение выход-вход).

Под связями в массиве цифровых блоков подразумеваются следующие линии:

- сигналы глобальных частот для тактирования блоков;
- шины адреса и данных для доступа к регистрам цифровых блоков;
- запросы на прерывание от цифровых блоков;
- соединение broadcast-линий имеющихся в каждом ряду цифровых блоков;
- сигналы для каскадирования блоков.

Каждый ряд цифровых блоков (рис. 3) содержит четыре блока. Два первых блока являются базовыми (basic, DBB), два других — коммуникационными (communication, DCB). Каждый ряд содержит четыре входных RI[3..0] и четыре выходных линии RO[3..0], а также линию «широковещательной связи» (broadcast) — BCROW.

Мультиплексоры, соединяющие входные шины ряда с глобальными входными шинами, управляются регистрами **RDIxRI**. Регистры **RDIxSYN** управляют синхронизацией входных сигналов (блоки S0–S3). Входные сигналы могут быть привязаны к системной частоте SYSCLK, либо проходить напрямую, то есть быть асинхронными по отношению к цифровым блокам.

Выходные линии и линия BCROW, так же как и линии глобальной шины, имеют специальный триггер-хранитель (keeper), сохраняющий последнее состояние линии. Каждая выходная линия имеет таблицу преобразования (блоки L0–L3), которая позволяет ввести дополнительную логическую функцию при формировании выхода. Функция таблицы преобразования для каждой выходной линии ряда задается в регистрах **RDIxLTx** (рис. 4).

LUTx[3:0]	
0h: 0000:	FALSE
1h: 0001:	A .AND. B
2h: 0010:	A .AND. $\overline{B}$
3h: 0011:	A
4h: 0100:	$\overline{A}$ .AND. B
5h: 0101:	B
6h: 0110:	A .XOR. B
7h: 0111:	A .OR. B
8h: 1000:	A .NOR. B
9h: 1001:	A .XNOR. B
Ah: 1010:	$\overline{B}$
Bh: 1011:	A .OR. $\overline{B}$
Ch: 1100:	$\overline{A}$
Dh: 1101:	$\overline{A}$ .OR. B
Eh: 1110:	A .NAND. B
Fh: 1111:	TRUE

Рис. 4. Функции таблицы преобразования

Регистры **RDIxIS** управляют мультиплексором на входе «A» в таблицах преобразования, а также задают связь broadcast-линии данного ряда с broadcast-линией другого ряда (линии BCROW[3..0]).

Регистры **RDIxROx** управляют буферами, подсоединяющими выходы таблиц преобразования каждой из выходных линий ряда к линиям глобальных выходных шин.

Цифровые блоки PSoC (рис. 5) подразделяются на два типа — базовые (Digital Basic Blocks, DBB) и коммуникационные (Digital Communication Blocks, DCB). Первый тип блоков позволяет реализовать такие периферийные модули, как таймеры, счетчики и ШИМ, а также модуль контроля с помощью циклического избыточного кода (CRC) и генератор псевдослучайной последовательности (PRS). Коммуникационные блоки, помимо указанных функций, позволяют создавать контроллеры интерфейса SPI или полнодуплексный UART.

Каждый цифровой блок построен на базе семи 8-битных регистров — трех конфигурационных, определяющих тип выполняемой блоком функции и конфигурацию входов и выходов блока, а также трех регистров данных и регистра управления функцией. Назначение бит в регистрах данных DxBxxDR0, DxBxxDR1, DxBxxDR2 и регистре статуса-управления DxBxxCR0 зависит от выполняемой блоком функции, которая определяется регистром DxBxxFN (табл. 1).

За конфигурацию входных сигналов отвечает регистр DxBxxIN, который управляет мультиплексорами, выбирающими сигнал входных данных и сигнал тактовой частоты для блока. Смысловое назначение входных сигналов для каждой функции блока приведено в таблице 2.

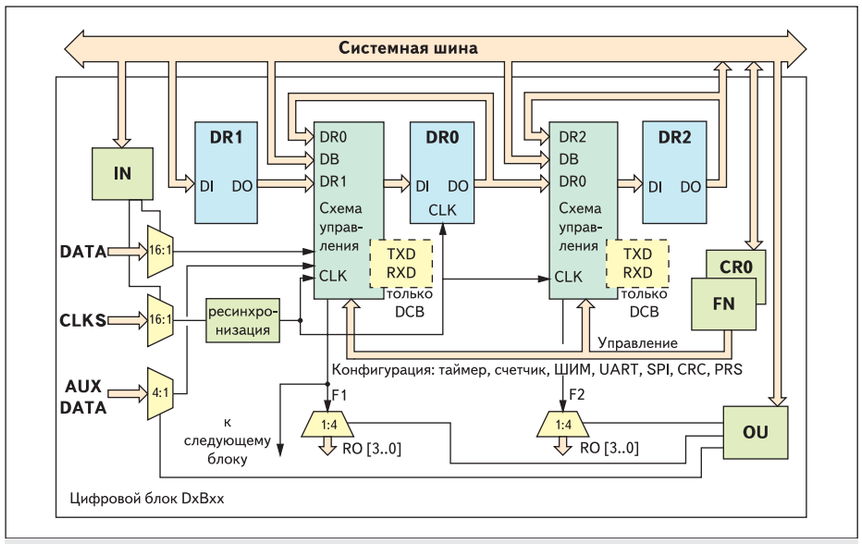

Рис. 5. Структура цифрового блока PSoC

KiT#59(6).qxd 6/27/2006 1:42 PM Page 58

Таблица 1. Назначение битов в регистрах данных DxBxxDR0, DxBxxDR1, DxBxxDR2 и регистре статуса-управления DxBxxCR0

Тип функции	DxBxxDR0		DxBxxDR1		DxBxxDR2		DxBxxCR0	
	функция	доступ	функция	доступ	функция	доступ	функция	доступ
Таймер	вычитающий счетчик	чтение[2]	период	запись	захват/сравнение	чтение/запись	управление	чтение/запись
Счетчик	вычитающий счетчик	чтение[2]	период	запись	сравнение	чтение/запись	управление	чтение/запись
Dead Band	вычитающий счетчик	чтение[2]	период	запись	нет	нет	управление	чтение/запись
CRC, PRS	регистр LFSR[1]	чтение[2]	полином	запись	начальное число	чтение/запись	управление	чтение/запись
SPI мастер	сдвигающий регистр	Нет	буфер передатчика	запись	буфер приемника	чтение	управление статус	чтение/запись[3]
SPI периферия	сдвигающий регистр	Нет	буфер передатчика	запись	буфер приемника	чтение	управление статус	чтение/запись[3]
TXUART	сдвигающий регистр	Нет	буфер передатчика	запись	нет	нет	управление статус	чтение/запись[3]
RXUART	сдвигающий регистр	Нет	нет	нет	буфер приемника	чтение	управление статус	чтение/запись[3]

1) Сдвиговый регистр с обратной связью.
2) При чтении из регистра DR0 на шину данных выдается 00h, а содержимое DR0 копируется в DR2.
3) Управляющие биты доступны для чтения-записи, статусные только для чтения.

Таблица 2. Смысловое назначение входных сигналов для каждой функции блока

Тип функции	Входы		
	DATA	CLK	AUX
Таймер	захват	CLK	Нет
Счетчик	разрешение работы	CLK	Нет
ШИМ с «мертвыми зонами» (Dead Band)	очистка	CLK	опорный сигнал*
CRC, PRS	Последовательные данные**	CLK	Нет
SPI мастер	MISO	CLK	Нет
SPI периферия	MOSI	SCLK	SS_***
TXUART	Нет	8×Бод/с CLK	Нет
RXUART	RXD	8×Бод/с CLK	Нет

*) мультиплексор не используется, сигнал жестко привязан к предыдущему блоку;
**) для CRC — последовательные данные, для PRS должен быть '0';
***) управляется регистром DxBxxOU.

Таблица 3. Смысловое назначение выходных сигналов для каждой функции блока

Тип функции	Выходы		
	главный	дополнительный	прерывание
Таймер	предел счета	сравнение	предел или сравнение
Счетчик	сравнение	предел счета	предел или сравнение
ШИМ с «мертвыми зонами» (Dead Band)	фаза 1	фаза 2	фаза 1
CRC, PRS	MSB	сравнение	сравнение
SPI мастер	MOSI	SCLK	TX буфер пуст или SPI транзакция завершена
SPI периферия	MISO	Нет*	TX буфер пуст или SPI транзакция завершена
TXUART	TXD	SCLK**	TX буфер пуст или передача завершена
RXUART	RXD	SCLK**	буфер RX полон

*) В данном случае регистр DxBxxOU управляет входным сигналом SS_;
**) UART может генерировать сигналы тактирования в моменты приема или передачи байта.

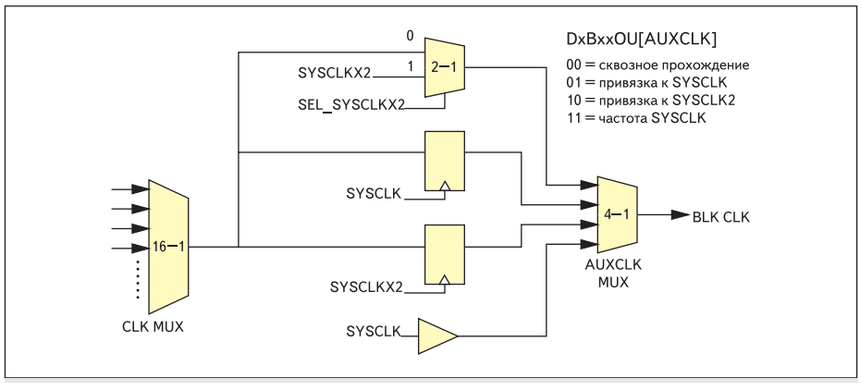

Рис. 6. Блок ресинхронизации

Регистр DxBxxOU управляет демультиплексором выходных сигналов, а также блоком ресинхронизации входной тактовой частоты (рис. 6). В таблице 3 приведено смысловое назначение выходных сигналов для каждой функции блока.

Ресинхронизация необходима для устранения возможной рассинхронизации блоков. Если тактовая частота образована делением системной частоты SYSCLK или удвоенной системной частоты SYSCLKx2 (например, частоты VC1, VC2, VC3), то необходимо привязать ее к соответствующей частоте. Так же следует проступать, если для тактирования блока используется выход другого блока, либо внешний сигнал. Использовать сквозное прохождение тактирующего сигнала не рекомендуется из-за возможного возникновения ошибок при выполнении операций чтения-записи регистров блока. Использовать этот режим следует, если требуется обеспечить работу блока в спящем режиме процессора (в этом случае частота либо должна быть образована от источника частоты 32K, либо должна являться внешним сигналом с контакта микросхемы) или если общение блока с процессором будет заключаться лишь в генерации прерывания.

Каждый цифровой блок имеет специальные линии для каскадирования блоков (сигналы переноса, сравнения, захвата и т. п.), что позволяет создавать периферийные модули с разрядностью большей, чем 8 бит. Модули пользователя в библиотеке PSoC Designer, имеющие разрядность больше 8, имеют уже правильно настроенные сигналы каскадирования, поэтому рекомендуется использовать готовые библиотечные модули.

Каждый блок в массиве цифровых блоков имеет свой выход запроса на прерывание. Приоритет этого запроса определяется порядковым номером блока в массиве, то есть блок с номером 0 в ряду 0 (блок DBB00) имеет наивысший приоритет, а блок DCB33 (если микросхема 29-й серии) имеет самый низший приоритет. За разрешение прерываний от цифровых блоков отвечают регистры маскирования INT_MSK1 и INT_MSK2 контроллера прерываний.

Подробно функции, реализуемые цифровыми блоками, рассмотрены в руководстве пользователя [1] в главе 17 “Digital blocks”. Все рассмотренные регистры могут либо программироваться непосредственно в тексте программы, либо настраиваться в режиме расположения модулей и задания связей в PSoC Designer (Interconnect View).

В качестве примера, демонстрирующего работу с конфигурируемой цифровой периферией, рассмотрим несложный «проект», который реализуем на известной уже нам демо-плате из комплекта CY3210-Miniprog1.

KiT#59(6).qxd 6/27/2006 1:42 PM Page 59

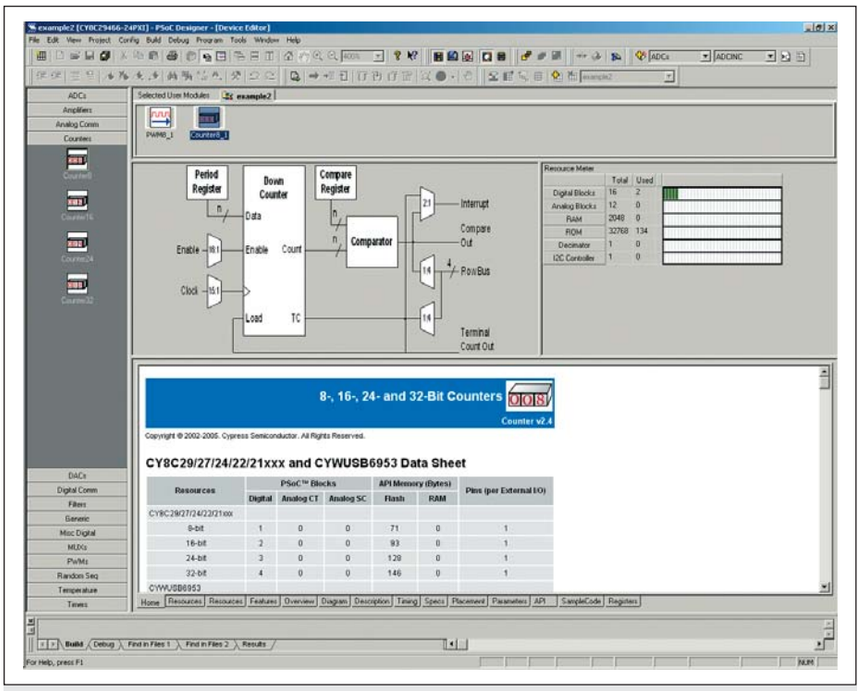

Рис. 7. Выбор модулей пользователя

Будем использовать счетчик и ШИМ для реализации поочередного плавного зажигания и гашения светодиодов. Яркость свечения светодиода в данном случае будет определяться скважностью импульсов, поступающих с выхода ШИМ, а счетчик будет использоваться для генерации прерывания, по которому происходит изменение скважности. Период счетчика, таким образом, будет определять скорость изменения яркости. Сигнал от микрокнопки, установленной на плате, будет являться сигналом разрешения работы.

Выполним все те действия по созданию нового проекта, которые были рассмотрены в первой части статьи. В редакторе системы (Device Editor,) в режиме выбора модулей пользователя (User Module Selection View,) выберем необходимые нам модули счетчика (Counter) и ШИМ (PWM) — рис. 7. При выборе модулей им присваивается имя, отражающее тип модуля и его порядковый номер, например PWM8_1. Это назначенное по умолчанию имя может быть изменено пользователем по его желанию.

Далее перейдем в режим задания связей (Interconnect View,) и зададим соединение выходных линий (RO0[3..0]) нулевого ряда цифровых блоков с контактами, которые управляют светодиодами (рис. 8).

Для обеспечения последовательного переключения светодиодов будем переключать в нашей программе выход модуля ШИМ на соответствующую линию RO.

Следующий этап — размещение модулей пользователя на массиве цифровых блоков и задание их параметров. Разместим модули счетчика и ШИМ на блоках DBB00 и DBB01 соответственно (рис. 9), хотя можно выбрать и другое место в пределах ряда. Определение нужного нам места размещения для выбранного в данный момент модуля пользователя происходит нажатием на кнопку Next Allowed Placement , а его размещение на блоке (или блоках) происходит при нажатии на кнопку Place User Module . Удалить выбранный модуль из массива блоков можно кнопкой Undo Place User Module .

После размещения компонентов зададим их параметры (рис. 10, 11).

Для обеспечения эффекта плавного изменения яркости светодиода частота его переключения должна быть не менее 25 Гц. Для обеспечения полного диапазона изменения яркости для модуля ШИМ зададим период, равный 256 тактам входной частоты. Соответственно настроим делители, формирующие глобальные частоты, следующим образом: VC1=SYSCLK/16, VC2=VC1/15, VC3=VC2/12. При системной частоте, по умолчанию равной 24 МГц, период ШИМ получается равным примерно 32,5 Гц.

Выход внутреннего компаратора, сравнивающего значение внутреннего счетчика ШИМ и значение ширины импульса, соединим с линией RO0[0] (это будет начальным значением, то есть когда выбран светодиод на контакте P20).

Для счетчика выберем период, равный 100 импульсам частоты VC3. Вход разрешения работы соединим со входной линией RI0[0], которую в свою очередь соединим с контактом, связанным с микрокнопкой (рис. 12). В качестве источника прерывания от счетчика выберем факт переполнения (Terminal Count). В обработчике прерывания будем производить изменение ширины импульса ШИМ, что и даст эффект плавного изменения яркости.

После задания всех настроек и подключений генерируем проект (кнопкой Generate Application,) и переходим в режим редактора программы (Application Editor,).

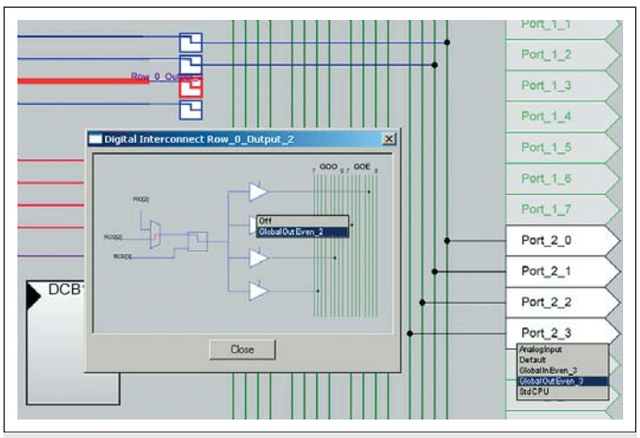

Рис. 8. Режим задания связей

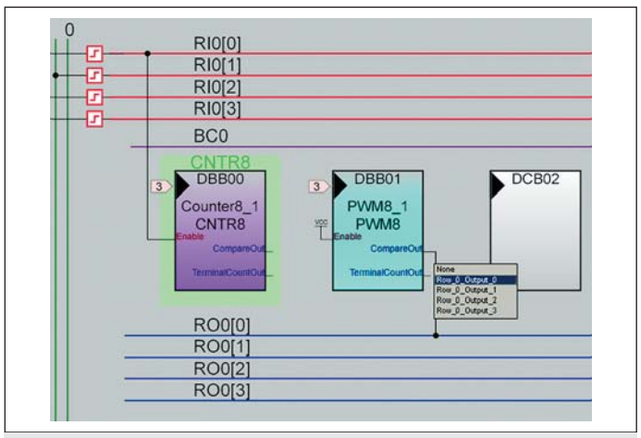

Рис. 9. Размещение модулей пользователя на массиве цифровых блоков

KiT#59(6).qxd 6/27/2006 1:42 PM Page 60

Counter8_1

User Module Parameters	Value
Clock	VC3
Enable	Row_0_Input_0
CompareOut	None
TerminalCountOut	None
Period	99
CompareValue	0
CompareType	Less Than
InterruptType	Terminal Count
ClockSync	Sync to SysClk
InvertEnable	Normal

Рис. 10. Параметры модуля счетчика

PWM8_1

User Module Parameters	Value
Clock	VC3
Enable	High
CompareOut	Row_0_Output_0
TerminalCountOut	None
Period	255
PulseWidth	0
CompareType	Less Than
InterruptType	Compare True
ClockSync	Sync to SysClk
InvertEnable	Normal

Рис. 11. Параметры модуля ШИМ

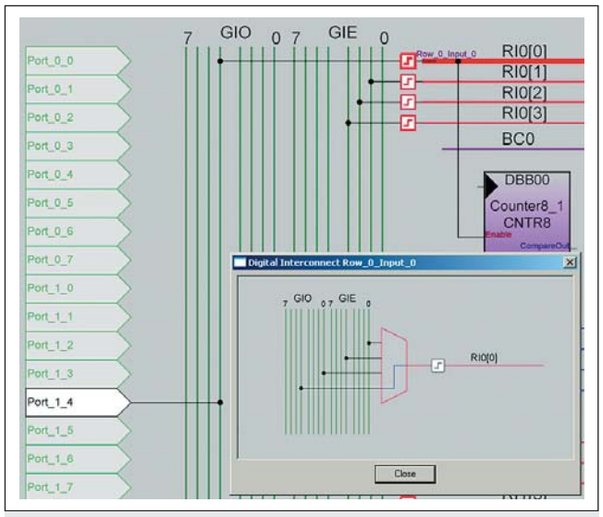

Рис. 12. Подсоединение линии RI0[0] к микрокнопке

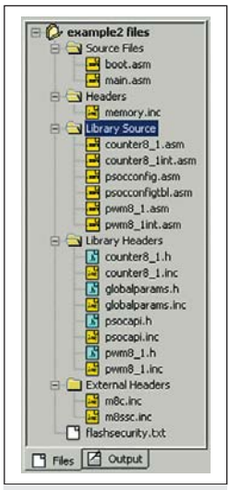

Рис. 13. Программная структура проекта

После генерации проекта в структуре проекта появятся автоматически сгенерированные файлы, содержащие функции программного интерфейса (API) для наших модулей (рис. 13).

В нашем проекте это файлы, имя которых начинается на counter8_1 и pwm8_1. Файлы counter8_1int.asm и pwm8_1int.asm представляют собой шаблоны обработчиков прерываний от соответствующего модуля. Сгенерированный файл boot.asm уже содержит переходы на соответствующие функции обработчиков для данных векторов прерываний, поэтому нет необходимости его редактировать. Так как мы будем использовать только прерывание от счетчика, то редактировать необходимо только файл counter8_1int.asm. Добавим в этот файл необходимую нам обработку — обновление ширины импульса ШИМ, смену направления изменения яркости (зажигание или гашение), а также переключение ШИМ на следующий светодиод. Текст программы, реализующей эти действия, приведен в листинге файла counter8_1int.asm. Обратите внимание, что добавляемый пользователем текст должен находиться между специальными маркерами «;@PSoC_UserCode_INIT@» или «;@PSoC_UserCode_BODY@» и «;@PSoC_UserCode_END@». В противном случае текст будет удален, например, при изменении параметров проекта и его повторной генерации.

Листинг файла counter8_1int.asm:

```
;;*****************************************************************
;;*****************************************************************
;; FILENAME: Counter8_1INT.asm
;; Version: 2.4, Updated on 2005/10/05 at 10:23:46
;; Generated by PSoC Designer ver 4.2 b1013 : 02 September, 2004
;;
;; DESCRIPTION: Counter8 Interrupt Service Routine
;;-----------------------------------------------------------------
;; Copyright (c) Cypress MicroSystems 2000-2004. All Rights Reserved.
;;*****************************************************************
;;*****************************************************************

include «m8c.inc»
include «memory.inc»
include «Counter8_1.inc»


;-----------------------------------------------
; Global Symbols
;-----------------------------------------------
export _Counter8_1_ISR


AREA InterruptRAM (RAM,REL,CON)

;@PSoC_UserCode_INIT@ (Do not change this line.)
;-----------------------------------------------
; Insert your custom declarations below this banner
;-----------------------------------------------

;-----------------------
; Includes
;-----------------------

include «pwm8_1.inc»

;-----------------------
; Constant Definitions
;-----------------------

;-----------------------
; Variable Allocation
;-----------------------

pulse_w : blk 1 ;ширина импульса
up_down : blk 1 ;направление изменения ширину импульса
led_select : blk 1 ;выбор светодиода (линии RO[3..0])

;-----------------------------------------------
; Insert your custom declarations above this banner
;-----------------------------------------------
;@PSoC_UserCode_END@ (Do not change this line.)


AREA UserModules (ROM, REL)

;-----------------------------------------------------------------
; FUNCTION NAME: _Counter8_1_ISR
;
; DESCRIPTION: Unless modified, this implements only a null handler stub.
;
;-----------------------------------------------------------------
;

_Counter8_1_ISR:

;@PSoC_UserCode_BODY@ (Do not change this line.)
;-----------------------------------------------
; Insert your custom code below this banner
;-----------------------------------------------
; NOTE: interrupt service routines must preserve
; the values of the A and X CPU registers.

    push A                          ; сохраняем регистр-
                                    ; аккумулятор в стеке

    mov A, [pulse_w]
    call PWM8_1_WritePulseWidth     ; обновляем значение
                                    ; ширины импульса
                                    ; в модуле PWM

    mov A, [up_down]                ;оцениваем направление
    jnz .down_count

    inc [pulse_w]                   ; увеличиваем ширину
                                    ; импульса
    jnc .exit
    mov [pulse_w], 255
    mov [up_down], 1                ; если переполнение —
                                    ; меняем направление
    jmp .exit

.down_count:
    dec [pulse_w]                   ; уменьшаем ширину
                                    ; импульса
    jnc .exit
    mov [pulse_w], 0
    mov [up_down], 0                ; если переполнение —
                                    ; меняем направление

    inc [led_select]                ;переключаем светодиод
    and [led_select], 0b00000011

    M8C_SetBank1
    mov A, reg[PWM8_1_OUTPUT_REG]
    and A, 0b11111100
    or A, [led_select]
    mov reg[PWM8_1_OUTPUT_REG], A   ; подсоединяем PWM
                                    ; к другой линии RO
    M8C_SetBank0

.exit:
    pop A
;-----------------------------------------------
; Insert your custom code above this banner
;-----------------------------------------------
;@PSoC_UserCode_END@ (Do not change this line.)

reti

; end of file Counter8_1INT.asm
```

Имя регистра PWM8_1_OUTPUT_REG в данном случае — всего лишь переопределенное имя регистра DxBxxOU, который управляет подключением выходов цифрового блока. Два младших бита этого регистра определяют линию RO, к которой подключен блок.

KiT#59(6).qxd 6/27/2006 1:42 PM Page 61

И, наконец, необходимо добавить несколько вызовов функций API наших модулей для разрешения их работы и разрешения генерации прерывания от счетчика (листинг файла main.asm).

Листинг файла main.asm:

```
;-----------------------------------------------------------------
; Assembly main line
;-----------------------------------------------------------------

include «m8c.inc» ; part specific constants and macros
include «memory.inc» ; Constants & macros for SMM/LMM and
Compiler
include «PSoCAPI.inc» ; PSoC API definitions for all User Modules

export _main

_main:

 ; Insert your main assembly code here.

call PWM8_1_Start
call Counter8_1_EnableInt
call Counter8_1_Start
M8C_EnableGInt

.terminate:
 jmp .terminate
```

После компиляции программы и загрузки прошивки в PSoC можно запустить программу, нажав на кнопку на демо-плате. Обратите внимание, что после подачи питания горят все светодиоды, хотя никакого значения в порты записано не было. Это происходит благодаря наличию триггеров-хранителей, которые после включения устанавливают на линиях RO и GOE высокий уровень.

В следующей статье будет рассмотрена аналоговая конфигурируемая периферия PSoC, являющаяся характерной особенностью данной системы на кристалле.

Литература

1. PSoC Mixed Signal Array Technical Reference Manual. Version 2.10.
2. PSoC Designer IDE User Guide.
3. Assembly Language User Guide.
4. 8-, 16-, 24- and 32-Bit Counters. CY8C29/27/24/22/21xxx and CYWUSB6953 Data Sheet.
5. 8- and 16-Bit Pulse Width Modulators. CY8C29/27/24/22/21xxx, CY7C6xxxx, and CYWUSB Data Sheet.