электронный научно-технический журнал
ИНЖЕНЕРНЫЙ ВЕСТНИК
издатель ФГБОУ ВПО «Московский государственный технический университет им. Н.Э. Баумана»

Конфигурирование, настройка и программирование модулей системы на кристалле PSoC

77-48211/424990

06, июнь 2012

Ермохина Н. И., Ибрагимов С. В., Хартов В. Я.

УДК 004. 318

Россия, МГТУ им. Н.Э. Баумана
nadezda.yermokhina@gmail.com
stas.ibragimov@gmail.com
khartovbmstu@gmail.com

В настоящее время возрастающие требования к объему и энергопотреблению электронной аппаратуры стимулируют дальнейшее повышение степени интеграции электронных устройств. Как правило, электронное устройство можно разделить на две части: цифровую и аналоговую. Цифровая часть может состоять из микропроцессора (или микроконтроллера) и дополнительных аппаратных блоков, программная реализация функций которых может оказаться нежелательна. Аналоговая часть содержит устройства и схемы взаимодействия с аналоговыми датчиками.

Программируемые системы на кристалле (PSoC) представляют собой аналогово-цифровые интегральные микросхемы, позволяющие реализовывать все части проекта на системном уровне интеграции. PSoC включают в себя процессорное ядро, аналоговые и цифровые конфигурируемые блоки и программируемую матрицу межсоединений и позволяют обойтись минимумом внешних элементов при разработке изделия, увеличивая его технологичность и значительно сокращая стоимость и время разработки. Широкие возможности конфигурирования PSoC позволяют легко адаптировать проектное решение на их основе к поставленным требованиям. Компания Cypress, известный производитель PSoC, предоставляет три линейки систем в зависимости от функциональных возможностей и процессорного ядра:

- PSoC 1, рекомендуемая для замены 8-битных микроконтроллеров с 8-битным ядром M8,
- PSoC 3 с ядром 8051 и тактовой частотой 67МГц,
- PSoC 5, высокопроизводительная с ядром ARM Cortex-M3.

Функционально PSOC можно разделить на следующие подсистемы.

- Подсистема центрального процессора. Состоит из процессорного ядра ARM, контроллера прерываний, контроллера прямого доступа к памяти и контроллера кэш-памяти.
- Подсистема программирования и отладки. Реализует возможности внутрисхемного программирования и отладки с помощью интерфейсов JTAG и SWD.
- Подсистема памяти. В состав PSOC входит три вида памяти: статическая память с произвольным доступом, флэш-память программ и перепрограммируемое ПЗУ для хранения параметров, настроек и т.д.
- Цифровая подсистема. Состоит из матрицы универсальных цифровых блоков, массива программируемых таймеров/счетчиков и блока аппаратной поддержки USB 2.0. Каждый универсальный цифровой блок соединен с системной шиной посредством пары регистров.
- Аналоговая подсистема. Состоит из аналогово-цифровых и цифрово-аналоговых преобразователей, блока цифровой фильтрации, операционных усилителей, компараторов и универсальных аналоговых блоков. Универсальный аналоговый блок строится на основе операционного усилителя по технологии переключаемых конденсаторов и, в зависимости от конфигурации цепей обратной связи, может выполнять роль усилителя с программируемым коэффициентом усиления, трансимпедансного усилителя, устройства выборки/хранения и т.д.
- Общесистемные ресурсы. Программируемое дерево синхронизации, часы точного времени, система управления энергосбережением, подсистема ввода/вывода и т.д.

Для взаимодействия с внешними устройствами применяется многофункциональные конфигурируемые модули ввода/вывода, связанные с каждым выводом микросхемы. Благодаря этому, режим работы линий ввода/вывода может меняться в процессе работы устройства, обеспечивая как аналоговый, так и цифровой ввод/вывод.

Для связи компонентов PSOC между собой с модулями ввода/вывода применяются две программируемые матрицы межсоединений, учитывающие специфические требования, предъявляемые к передаче аналоговых и цифровых сигналов.

Блок цифровой фильтрации реализован на основе ядра цифрового сигнального процессора и, используя механизм прямого доступа к памяти, позволяет осуществлять фильтрацию оцифрованных сигналов без нагрузки на ЦПУ.

Средства разработки и отладки для PSOC

Для разработки и отладки проектов используют две среды: PSoC Designer (для линейки PSoC 1) и PsoC Creator (для линеек PSoC 3 и 5). Обе среды предоставляют возможность максимально простого конфигурирования и программирования PSOC. Реализация проекта на PSOC предполагает использование отдельных компонентов, как специфических для конкретной задачи, так и библиотечных компонентов общего назначения. Такая модульная архитектура значительно упрощает процесс проектирования и повышает уровень повторного использования кода.

Компонент может включать в себя аппаратную реализацию компонента на основе конфигурируемых ресурсов PSOC, шаблоны программного кода (API) для взаимодействия с аппаратной частью, набор параметров, влияющих на реализацию или функциональные возможности и программу на языке Си для упрощения задания параметров. При компиляции проекта, для каждого экземпляра каждого компонента проекта по шаблонам, созданным разработчиком компонента, генерируются файлы со всем необходимым кодом для взаимодействия с компонентом.

Аппаратная часть компонента реализуется либо с помощью встроенного схемного редактора, либо описывается на языке Verilog. Совместное описание аппаратной и программной части компонента и возможность использовать компонента при построении других компонентов позволяет пользователю рассматривать компонент как "черный ящик", абстрагировавшись от деталей его внутреннего устройства, используя объектно-ориентированный подход при проектировании.

При компиляции проводится синтез сгенерированного HDL описания проекта и анализ полученной реализации для определения временных параметров. Отчеты синтезатора и результаты анализа временных параметров доступны пользователю в удобном для чтения формате.

Модуль отладки и трассировки обеспечивает возможность внутрисхемной отладки через интерфейсы JTAG и SWD. Отладчик уровня исходного теста, включенный в среду разработки, имеет широкие функциональные возможности, в частности чтение и изменение содержимого памяти, пошаговое исполнение кода, установку точек останова и т.д.

Примеры конфигурирования и настройки модулей системы

Пример 1 (платформа PSoC 3 – кристалл CY8C3866AXI-040, отладочный набор CY8CKIT-030). Продемонстрируем генерацию синусоиды, используя ЦАП и прямой доступ к памяти.

Осуществим мигание светодиода с частотой, пропорциональной напряжению, задаваемому пользователем с помощью потенциометра и преобразуемому АЦП в 8-битовый код, а также выведем это значение на LCD-дисплей. Схема аппаратной части тестового проекта приведена на рис.1.

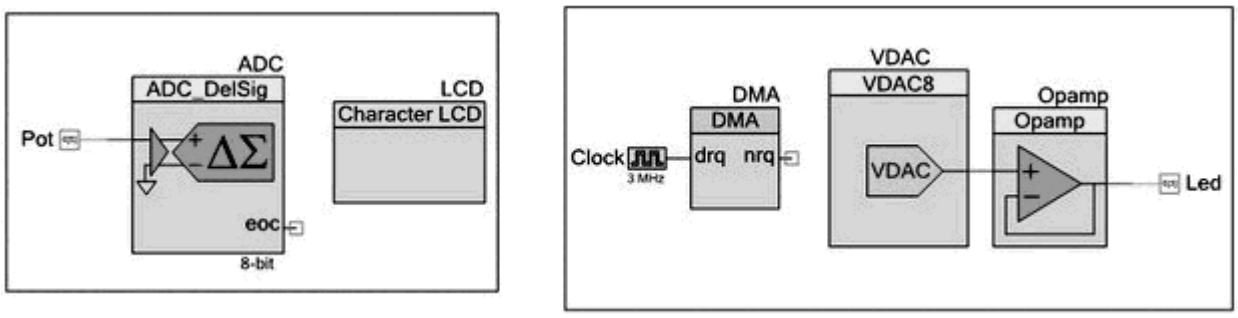
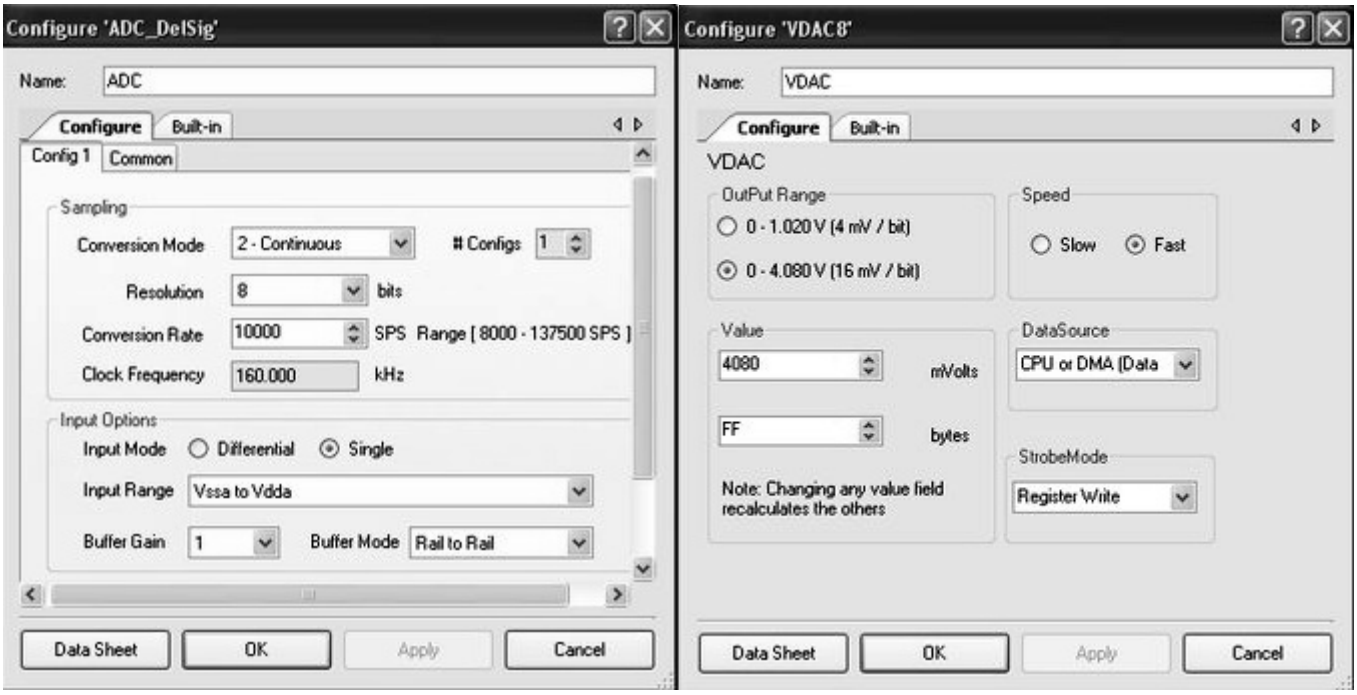


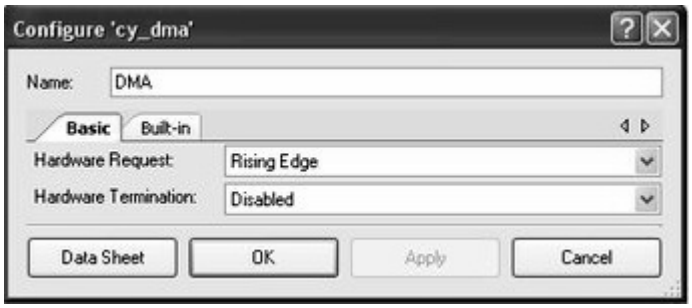
Рис. 1. Аппаратная часть примера 1

В программе (см. листинг программы) задаем значения для синусоидальной волны в интервале от 0x3D до 0x9F (так как это две критичные точки для светодиода – когда он не горит и когда находится в режиме насыщения). Считываемое значение напряжения, заданное потенциометром, поступает на сигма-дельта АЦП (на рис.1 видно, что вывод потенциометра, обозначенный меткой Pot, непосредственно связан с входом АЦП). Полученное значение записывается в переменную voltageRawCount, значение которой выводится на дисплей. В зависимости от этой переменной расчитывается делитель частоты и уже с новой частотой тактирования происходит обращение компонентом прямого доступа к памяти (DMA) к памяти данных, где хранятся значения напряжений синусоидальной волны. Считанные данные заносятся в регистр данных, откуда сразу поступают на ЦАП, который в аналоговом виде выводит их на светодиод (Led) с частотой, пропорциональной значению от АЦП.

Так как значения напряжений, задаваемых потенциометром, должны лежать в интервале 0–255, то необходимо настроить компонент сигма-дельта АЦП и установить параметр разрешения 8 бит (остальные параметры автоматически подстроятся под данное изменение). Компонент DMA должен быть настроен на срабатывание по перепаду из 0 в 1 (Rising Edge). Для компонента ЦАП: источник данных – DMA, Register Write – любые данные, записываемые в регистры данных и сразу пересылаемые в ЦАП. Настройка компонентов приведена на рис.2.



а) б)



в)

Рис. 2. Настройка компонентов сигма-дельта АЦП (а), ЦАП (б) и DMA(в)

Листинг программного кода тестового примера:

```
#include <device.h>
/* Переменные LCD-модуля */
#define ROW_0                   0
#define COLUMN_0                0
#define COLUMN_9                9
#define COLUMN_10               10
#define COLUMN_11               11
/* Очистка десятых и сотых */
#define CLEAR_TENS_HUNDREDS     "    "
/* Очистка сотых */
#define CLEAR_HUNDREDS          "   "
```

```
/* Значения минимального и максимального уровней напряжения */
#define MIN_COUNT               0
#define MAX_COUNT               0xFF
/* Настройка DMA */
#define DMA_BYTES_PER_BURST     1
#define DMA_REQUEST_PER_BURST   1
#define DMA_SRC_BASE          (CYDEV_SRAM_BASE)     /* Адрес источника */
#define DMA_DST_BASE          (CYDEV_PERIPH_BASE)   /* Адрес назначения */
/* Переменные для DMA */
uint8 DMA_Chan;
uint8 DMA_TD[1];
/* Таблица значений напряжений для DMA, посылаемых в ЦАП */
const uint8 voltageWave[] =
{ 0x6D,0x6F,0x71,0x73,0x75,0x77,0x79,0x7B,0x7D,0x7F,0x81,0x83,0x85,0x87,
0x89,0x8B,0x8D,0x8F,0x91,0x93,0x95,0x97,0x99,0x9B,0x9C,0x9D,0x9D,0x9E,0x9E,
0x9F,0x9F,0x9F,0x9E,0x9E,0x9E,0x9C,0x9C,0x9B,0x99,0x97,0x95,0x93,0x91,0x8F,
0x8D,0x8B,0x89,0x87,0x85,0x83,0x81,0x7F,0x7D,0x7B,0x79,0x77,0x75,0x73,0x71,
0x6F,0x6D,0x6B,0x69,0x67,0x65,0x63,0x61,0x5F,0x5D,0x5B,0x59,0x57,0x55,0x53,
0x51,0x4F,0x4D,0x4B,0x49,0x47,0x45,0x43,0x41,0x40,0x40,0x3F,0x3F,0x3D,0x3D,
0x3D,0x3D,0x3D,0x3D,0x3F,0x41,0x43,0x45,0x47,0x49,0x4B,0x4D,0x4F,0x51,0x53,
0x55,0x57,0x59,0x5B,0x5D,0x5F,0x61,0x63,0x65,0x67,0x69,0x6B};
void main()
{ int16 voltageRawCount;
  /* Разрешаем работу АЦП, ЖК-модуля, ЦАП, ОУ */
  ADC_Start();
  LCD_Start();
  VDAC_Start();
  Opamp_Start();
  /* Устанавливаем курсор ЖК-дисплея на строку 0, колонку 0 */
  LCD_Position(ROW_0, COLUMN_0);
  /* Выводим на ЖК-дисплей */
  LCD_PrintString("V Count: ");
  /* Конфигурация DMA */
  DMA_Chan = DMA_DmaInitialize(DMA_BYTES_PER_BURST,DMA_REQUEST_PER_BURST,
  HI16(DMA_SRC_BASE),HI16(DMA_DST_BASE)); /* Инициализировать канал */
  DMA_TD[0] = CyDmaTdAllocate();  /* Выделить дескриптор транзакций */
  CyDmaTdSetConfiguration(DMA_TD[0],116,DMA_INVALID_TD,TD_INC_SRC_ADR);
  /* Настроить дескриптор транзакций */
  CyDmaTdSetAddress(DMA_TD[0],LO16((uint32)voltageWave),
  LO16((uint32)VDAC_Data_PTR));  /* Задать адрес источника и назначения */
```

```
CyDmaChSetInitialTd(DMA_Chan, DMA_TD[0]);
CyDmaChEnable(DMA_Chan, 1); /* Разрешить работу канала */
Clock_Start();       /* Разрешить работу генератора синхросигнала */
ADC_StartConvert();  /* Начать преобразование */
while(1)
/* Ждем окончания преобразования */
{ADC_IsEndConversion(ADC_WAIT_FOR_RESULT);
 voltageRawCount = ADC_GetResult16();   /* Получаем результат */
 /* Минимальная граница */
 if (voltageRawCount < MIN_COUNT)
 {voltageRawCount = MIN_COUNT;
 }
 /* Максимальная граница */
 else if(voltageRawCount > MAX_COUNT)
 {voltageRawCount = MAX_COUNT;
 }
 else {}
 /* Настройка делителя частоты. Светодиод мигает в зависимости от величины
 voltageRawCount. При частоте синхронизации 3 МГц минимальный делитель (при
 напряжении 0В) равен 1000 для мигания с максимальной скоростью,
 наибольший делитель - 52200 для мигания с минимальной скоростью. */
 Clock_SetDivider(((uint32)999*260+voltageRawCount)/ (260-voltageRawCount));
 /* Передвигаем курсор на LCD-дисплее */
 LCD_Position(ROW_0, COLUMN_9);
 LCD_PrintNumber(voltageRawCount);
 if (voltageRawCount < 10)
 {/* Передвигаем курсор на LCD-модуле */
 LCD_Position(ROW_0,COLUMN_10);
 /* Удаляем предыдущие значения переменных */
 LCD_PrintString(CLEAR_TENS_HUNDREDS);
 }
 else if (voltageRawCount < 100)
 {/* Передвигаем курсор на LCD-модуле */
 LCD_Position(ROW_0,COLUMN_11);
 LCD_PrintString(CLEAR_HUNDREDS);
 }
 else
 { }    }    }
```

Пример 2 (платформа PSoC 5 – кристалл CY8C5588AXI-ES1, отладочный набор CY8CKIT-014). Разработаем частотный манипулятор, используя его для передачи сообщения по протоколу UART. Схема аппаратной части проекта приведена на рис.3. Частотный манипулятор реализуем на основе ЦАП с разрешением восемь бит и двух независимых каналов DMA. Выбор используемого канала DMA осуществляется с помощью цифрового мультиплексора. Оба канала DMA настроены на передачу входного кода из массива, расположенного во Flash-памяти, в регистр данных ЦАП, формируя на выходе последнего нужную временную диаграмму. Таким образом, ресурсы центрального процессора не тратятся на генерацию сигнала. Входной сигнал манипулятора снимается с выхода стандартного компонента UART, настроенного на передачу.

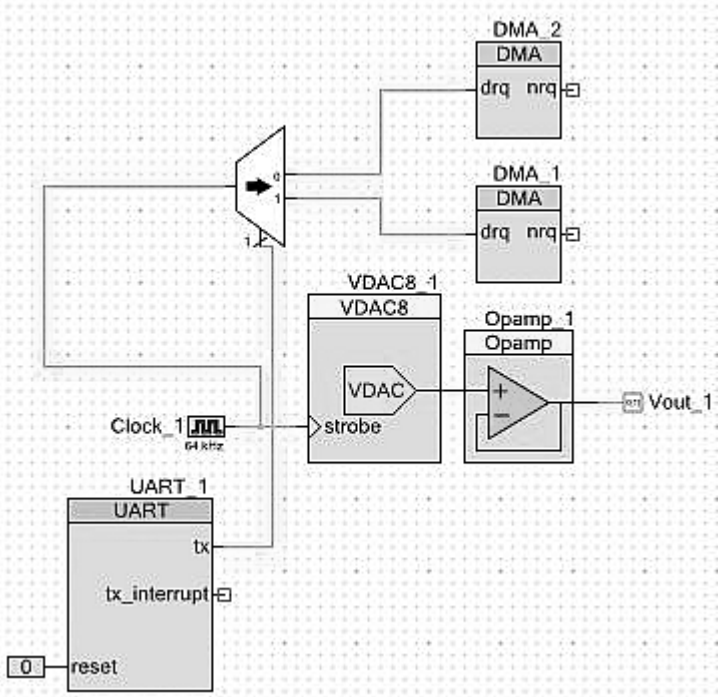


Рис.3. Схема аппаратной части примера 2

Сигнал синхронизации Clock_1 с частотой 64 кГц используется для генерации выходных частот, вычисляемых по формуле:

$$f_{вых} = \frac{f_{clk}}{N},$$

где $f_{вых}$ – частота выходного сигнала,

f_{clk} – частота синхросигнала,

N – количество отсчетов в одном периоде выходного сигнала.

Листинг программного кода проекта:

```
#include <device.h>
#include <VDAC8_1.h>
#include <Clock_1.h>
#include <Opamp_1.h>
#include <DMA_1_dma.h>
#include <DMA_2_dma.h>
#include <UART_1.h>
#define TABLE_LENGTH 32 //Количество отсчетов первой частоты.
#define TABLE_LENGTH1 64     //Количество отсчетов второй частоты.
#define DMA_BYTES_PER_BURST 1      //Передача 1 байта за транзакцию
#define DMA_REQUEST_PER_BURST 1
//Осуществление транзакции только по запросу
CYCODE const uint8 sineTable[TABLE_LENGTH]
={128,152,176,198,218,234,245,253,255,253,245,234,218,198,176,152,127,
103,79,57,37,21,10,2,0,2,10,21,37,57,79,103};
CYCODE const uint8 sineTable1[TABLE_LENGTH1]
={128,140,152,165,176,188,198,208,218,226,234,240,245,250,253,254,255,
254,253,250,245,240,234,226,218,208,198,188,176,165,152,140,127,115,10
3,90,79,67,57,47,37,29,21,15,10,5,2,1,0,1,2,5,10,15,21,29,37,47,57,67,
79,90,103,115};
#define DMA_SRC_BASE (&sineTable[0]) //Адрес источника первого канала
#define DMA_SRC_BASE1 (&sineTable1[0])//Адрес источника второго канала
#define DMA_DST_BASE (CYDEV_PERIPH_BASE)  //Адрес назначения
void main()
{
uint8 DMA_Chan;
uint8 DMA_TD[1];
uint8 DMA_Chan1;
uint8 DMA_TD1[1];
     CyGlobalIntEnable;  //Разрешить прерывания
     Clock_1_Start();    //Разрешить работу генератора синхросигнала
     VDAC8_1_Start();    //Включить ЦАП
     Opamp_1_Start();    //Включить операционный усилитель
//Инициализировать первый канал
DMA_Chan = DMA_1_DmaInitialize(DMA_BYTES_PER_BURST,
DMA_REQUEST_PER_BURST, HI16(DMA_SRC_BASE), HI16(DMA_DST_BASE));
     DMA_TD[0] = CyDmaTdAllocate();//Выделить дескриптор транзакции
CyDmaTdSetConfiguration(DMA_TD[0], TABLE_LENGTH, DMA_TD[0],
TD_INC_SRC_ADR); //Настроить дескриптор транзакции
```

```
CyDmaTdSetAddress(DMA_TD[0],LO16((uint32)sineTable),
LO16((uint32)VDAC8_1_Data_PTR));//Задать адреса источника и назначения
     CyDmaChSetInitialTd(DMA_Chan, DMA_TD[0]);
     CyDmaChEnable(DMA_Chan,1);        //Разрешить работу канала
//Инициализировать второй канал
DMA_Chan1 = DMA_2_DmaInitialize(DMA_BYTES_PER_BURST,
DMA_REQUEST_PER_BURST, HI16(DMA_SRC_BASE1), HI16(DMA_DST_BASE));
     DMA_TD1[0] = CyDmaTdAllocate();//Выделить дескриптор транзакции
     CyDmaTdSetConfiguration(DMA_TD1[0], TABLE_LENGTH1, DMA_TD1[0],
 TD_INC_SRC_ADR); //Настроить дескриптор транзакции
     CyDmaTdSetAddress(DMA_TD1[0], LO16((uint32)sineTable1),
LO16((uint32)VDAC8_1_Data_PTR));//Задать адреса источника и назначения
     CyDmaChSetInitialTd(DMA_Chan1, DMA_TD1[0]);
     CyDmaChEnable(DMA_Chan1, 1);       //Разрешить работу канала
     UART_1_Start();                    //Разрешить работу блока UART
     UART_1_PutString("Hello, World!"); //Передать строку
     for(;;) {}                         //бесконечный цикл
}
```

Выводы. Высокая степень интеграции PSoC, разнообразие конструктивных исполнений и сравнительно низкая цена позволяют применять системы на кристалле для разработки самых разных устройств автоматики, медицины, потребительской электроники, связи и др. PSOC младшей серии (1) могут с успехом применяться вместо традиционных 8-битных микроконтроллеров. Благодаря наличию встроенных аналоговых блоков, количество внешних элементов в схеме устройства удается сократить, а возможность аппаратной реализации цифровых функций с повышенными требованиями к быстродействию нивелирует относительно низкую производительность ЦПУ. Наличие сопроцессора цифровой обработки сигналов и высокопроизводительного ядра ARM Cortex-M3 в серии PSoC 5 дает возможность использовать их в задачах радиосвязи и телекоммуникаций, а наличие конфигурируемых цифровых блоков дает возможность использовать PSoC вместо ПЛИС малой емкости для решения задач специфической обработки цифровых сигналов.

Список использованных источников

1. PSoC 3: CY8C38 Family Data sheet
2. CY8CKIT-030 PSoC 3 Development Kit Guide
3. PSOC 5: CY8C55 Family Data Sheet
4. CY8CKIT-014 PSOC 5 First Touch Starter Kit Guide

5. J. Pardue. C Programming for microcontrollers. 2005
6. PSoC 5 Architecture Technical Reference Manual