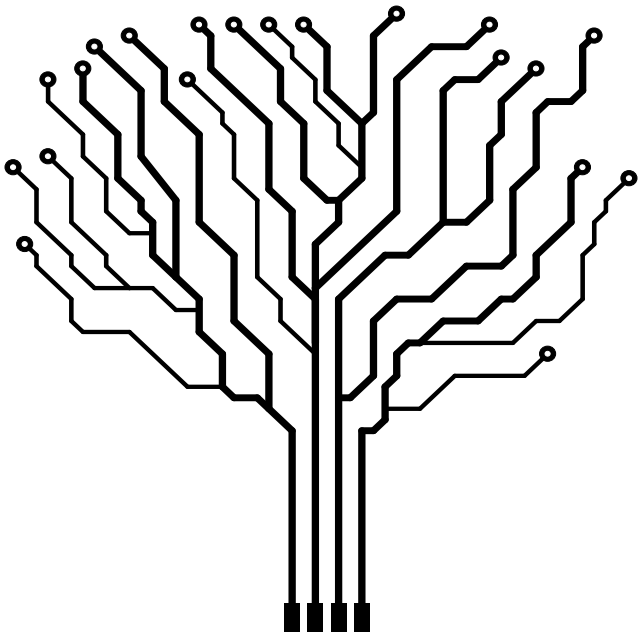
УНИВЕРСИТЕТ ИТМО

М.Я. АФАНАСЬЕВ Ю.В. ФЕДОСОВ

ВСТРОЕННЫЕ КОМПЬЮТЕРНЫЕ СИСТЕМЫ

Методические рекомендации
по выполнению лабораторных работ



Санкт-Петербург
2016

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

УНИВЕРСИТЕТ ИТМО

М. Я. Афанасьев Ю. В. Федосов

ВСТРОЕННЫЕ КОМПЬЮТЕРНЫЕ СИСТЕМЫ

Методические рекомендации по выполнению лабораторных работ

УНИВЕРСИТЕТ ИТМО

Санкт-Петербург

2016

Афанасьев М. Я., Федосов Ю. В. Встроенные компьютерные системы. Методические рекомендации по выполнению лабораторных работ — СПб: Университет ИТМО, 2016. — 51 с.

В учебно-методическом пособии приведены методические рекомендации для выполнения лабораторных работ по дисциплине «Встроенные компьютерные системы».

Пособие предназначено для студентов высших учебных заведений, обучающихся по направлению подготовки 12.04.01 «Приборостроение».

Рекомендовано к печати учёным советом факультета компьютерных технологий и управления, протокол № XX от XX xxxxxx 2016 г.

УНИВЕРСИТЕТ ИТМО

Университет ИТМО — ведущий вуз России в области информационных и фотонных технологий, один из немногих российских вузов, получивших в 2009 году статус национального исследовательского университета. С 2013 года Университет ИТМО — участник программы повышения конкурентоспособности российских университетов среди ведущих мировых научно-образовательных центров, известной как проект «5 в 100». Цель Университета ИТМО — становление исследовательского университета мирового уровня, предпринимательского по типу, ориентированного на интернационализацию всех направлений деятельности.

ОГЛАВЛЕНИЕ

В теории, теория
и практика неразделимы.
На практике это не так.

— Йоги Берра

ВВЕДЕНИЕ

Встроенные вычислительные системы, являясь одной из фундаментальных основ современного промышленного производства, позволяют решать самые разнообразные и специфические производственные задачи. Объём использования встроенных систем в промышленности огромен.

Несмотря на внушительный объём производства, потребность во встроенных вычислительных системах неуклонно растёт. Широко развивается производство станков с ЧПУ, интеллектуального производственного оборудования, автоматизированных производственных линий, промышленных контроллеров, и иных встраиваемых систем промышленного применения.

Наряду с иностранными производителями, увеличивают долю на рынке и производители отечественной электронной техники. Отечественные предприятия потребляют всё больше встроенных ЭВМ и систем, причем эта доля увеличивается.

Для того чтобы исключить отставание по ключевым направлениям промышленного производства, и, в частности, специализированной электроники, была принята программа «Развитие промышленности и повышение её конкурентоспособности», утвержденная постановлением Правительства Российской Федерации от 15 апреля 2014 г.

В программе одним из пунктов обозначается недопущение научного и технического отставания Российской Федерации от признаваемого мирового уровня. В соответствии с программой, вводится стимулирование производства высокотехнологичной аппаратуры, в частности — сверхбольших интегральных схем, персональных электронно-вычислительных машин и электронных приборов на их основе.

Особенно важным направлением, на которое нацелена программа, является повышение эффективности

использования производственного потенциала оборонно-промышленного комплекса для обеспечения разработки и производства новых видов вооружения и военной техники.

Поставленные задачи требуют планового подхода и комплексной работы. Одним из основных условий их решения является четкое понимание особенностей работы программного комплекса, предназначенного для выполнения технологических задач.

Известно, что разработка программного обеспечения является в десятки раз более трудоёмкой и дорогостоящей в сравнении с разработкой аппаратной части изделия. Ключевым отличием разработки программы является возможность её переноса с одного оборудования на другое (портирование) с изменением функциональности, или без него.

Это повышает гибкость производственных процессов, облегчает наладку оборудования, служит основой для типизации и унификации вычислительных систем. Использование стандартизированных языков программирования, таких, как, например, C/C++, позволяет разрабатывать мультиплатформенные решения, легко масштабируемые под различные типы оборудования. Наряду с прочими преимуществами, это позволяет ускорить разработку, используя имеющийся научно-технический задел.

Желаем Вам успехов в нелегком, но чрезвычайно увлекательном деле — освоении сложной вычислительной техники.

ОБЗОР АППАРАТНОГО ОБЕСПЕЧЕНИЯ

Для выполнения лабораторных работ будет использоваться отладочный комплект MSP430 Launchpad, созданный компанией Texas Instruments. В состав комплекта входит отладочная плата MSP-EXP430G2 и два шестнадцатибитных микроконтроллера архитектуры MSP430.

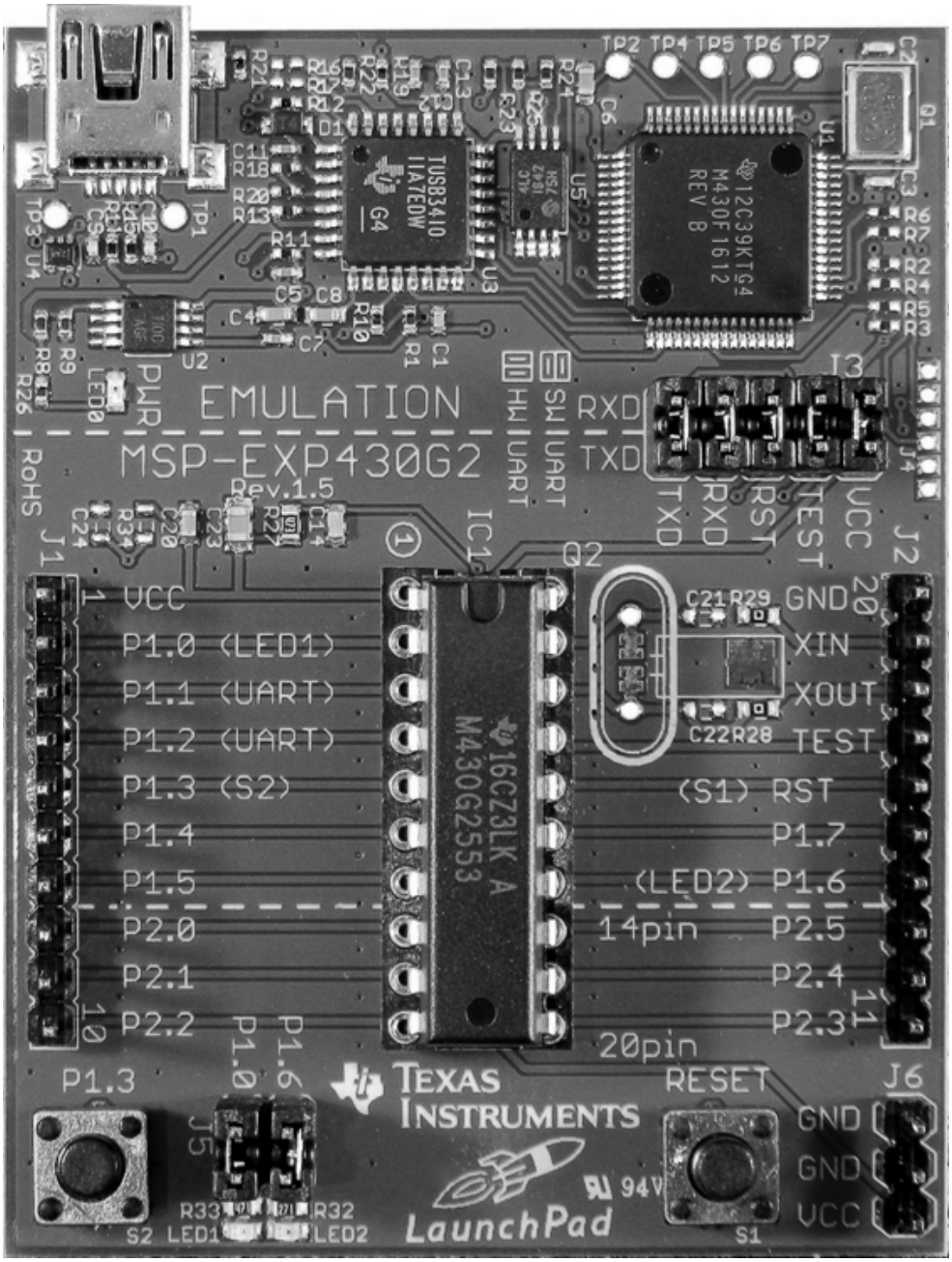


Рисунок 1. Отладочная плата MSP-EXP430G2

Отладочная плата (рис. 1) состоит из двух частей: верхняя часть (на рис. 1 обозначена как EMULATOR) — программатор, нижняя — целевое устройство со слотом для установки микроконтроллера и минимальным набором периферийных устройств.

В качестве программатора используется система внутрисхемной отладки и эмуляции по протоколу JTAG[1] в двухпроводной реализации от Texas Instruments, именуемой Spy-Bi-Wire [1]. Программатор позволяет работать как с целевым микроконтроллером, так и с любым другим внешним устройством.

В слот целевого устройства может быть установлен любой микроконтроллер серии MSP430G2 Value Line. В комплекте поставляются чипы MSP430G2553N20 и MSP430G2452N20. Все выводы данных микроконтроллеров разведены на штыревые разъемы, размещающиеся по бокам платы (на рис. 1 отмечены P1.X и P2.X). К выводам P1.0 и P1.6 подключены индикаторные светодиоды красного и зеленого цветов соответственно. Вывод P1.3 соединен с тактовой нормально разомкнутой кнопкой. Также реализована схема сброса микроконтроллера (кнопка RESET целевого устройства) и возможность подключения внешнего кварцевого резонатора с частотой 32768 Гц (на рис. 1 отмечено как Q2).

Для подключения к персональному компьютеру используется интерфейс USB, позволяющий взаимодействовать с программатором, а также задействовать возможности двухсторонней передачи данных между прикладными программами и микроконтроллером целевого устройства.

Стандартное напряжение питания программатора и целевого устройства должно быть в диапазоне от 1,8 В до 3,6 В. Питание отладочной платы осуществляется по интерфейсу USB, выдающему 5 В, которые преобразуются в необходимые

[1] сокр. от англ. *Joint Test Action Group*; произносится «джей-та́г» — интерфейс предназначен для подключения сложных цифровых микросхем или устройств уровня печатной платы к стандартной аппаратуре тестирования и отладки [2].

для стабильной работы микроконтроллеров 3,6 В средствами установленного регулятора питания.

В табл. 1 представлены основные характеристики используемых микроконтроллеров.

Таблица 1

Характеристики микроконтроллеров серии Value Line

Характеристика	**MSP430G2553**	**MSP430G2452**
Рабочая частота, МГц	16	16
Энергонезависимая память, КБ	16	8
Оперативная память, КБ	0,5	0,25
Порты ввода вывода общего назначения	24	16
I^2C[2]	1	1
SPI[3]	1	1
UART[4]	1	0
Количество компараторов	8	8
Количество таймеров (16 бит)	2	1
BSL[5]	UART	нет
Энергопотребление в активном режиме, мкА/МГц	330	320
Энергопотребление в дежурном режиме, мкА/МГц	0,7	0,7

[2] сокр. от англ. *Inter-Integrated Circuit*; произносится «ай-ту-си/и-два-цэ/и-два-си» — последовательная шина данных для связи интегральных схем, использующая две двунаправленные линии связи [3].

[3] сокр. от англ. *Serial Peripheral Interface*, SPI bus; последовательный периферийный интерфейс, шина SPI — последовательный синхронный стандарт передачи данных в режиме полного дуплекса, предназначенный для обеспечения простого и недорогого сопряжения микроконтроллеров и периферии [4].

[4] сокр. от англ. *Universal Asynchronous Receiver-Transmitter*; УАПП, универсальный асинхронный приёмопередатчик — узел вычислительных устройств, предназначенный для организации связи с другими цифровыми устройствами. Преобразует передаваемые данные в последовательный вид так, чтобы было возможно передать их по цифровой линии другому устройству [5].

[5] сокр. от англ. *BootStrap Loader*, bootloader; произносится «бутлодер» — программа, размещающаяся в энергонезависимой памяти микроконтроллера и автоматически активируемая при подаче питания. Чаще всего используется для обновления прошивки микроконтроллера без использования специального программатора [6].

Окончание табл. 1

Характеристика	MSP430G2553	MSP430G2452
Время перехода из дежурного в активный режим, мкс	1,5	1,5
Дополнительные функции	Сторожевой таймер, температурный датчик, сброс/перезапуск по скачку питания	Сторожевой таймер, температурный датчик, сброс/перезапуск по скачку питания
Специальные порты ввода/вывода	Возможность подключения сенсорных датчиков	Возможность подключения сенсорных датчиков

ИНТЕГРИРОВАННАЯ СРЕДА РАЗРАБОТКИ

Для работы с отладочным комплектом MSP430 Launchpad используется интегрированная среда разработки Code Composer Studio. Данная среда включает в себя набор инструментов, позволяющих разрабатывать и отлаживать программное обеспечение встроенных систем. Она включает в себя оптимизирующий компилятор языков C/C++, редактор исходного кода, систему автоматической сборки, отладчик, профайлер и многие другие инструменты.

Среда доступна на официальном сайте компании Texas Instruments. В процессе установки необходимо постоянное подключение к сети Интернет. Для установки необходимо выполнить следующие действия:

1) Зарегистрироваться на портале http://www.ti.com/. Для этого в правом вернем углу окна нажать на ссылку Login/Register, на открывшейся странице заполнить все поля формы регистрации, получить письмо с подтверждением аккаунта, после чего зайти на портал со своими учетными данными.

2) После этого перейти на страницу Code Composer Studio по адресу http://www.ti.com/tool/CCSTUDIO и в нижней части страницы в разделе Downloads нажать кнопку с надписью Windows.

3) На открывшейся странице подтвердить, что вы не будете использовать данное программное обеспечение для военных целей. Для этого ответить на вопрос «What end-equipment/application will you use this file for», выбрав только пункт «Civil», а в самом низу страницы, рядом с надписью «I CERTIFY ALL THE ABOVE IS TRUE», установить переключатель в положение «Yes». Затем выбрать пункт «Remember me for next 6 months», после чего нажать кнопку «Submit» (рис. 2).

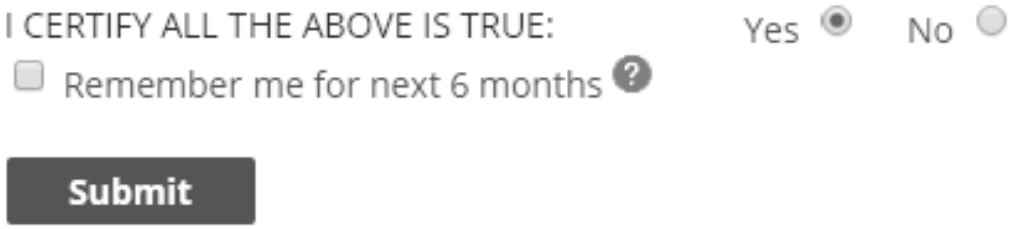


Рисунок 2. Форма подтверждения загрузки программного обеспечения Code Composer Studio

4) На открывшейся странице нажать кнопку с надписью «Download» и дождаться загрузки файла «ccs_setup_win32.exe», после чего запустить его.

5) В открывшемся диалоге согласиться с условиями пользовательского соглашения, выбрав пункт «I accept the terms of the license agreement», после чего нажать кнопку «Next >».

6) В следующем окне диалога выбрать место, куда будет установлено программное обеспечение, либо оставить значение по умолчанию (корень логического диска «C», директория «ti»), после чего нажать кнопку «Next >».

7) В следующем окне выбрать пункт списка «MSP Ultra Low Power MCUs», после чего нажать кнопку «Next >».

8) В следующем окне выбрать пункт списка «MSP430 USB FET», после чего нажать кнопку «Next >».

9) В следующем окне выбрать пункт вне списка «Select All», после чего нажать кнопку «Finish».

10) Дождаться окончания процесса установки. Процесс может занять от десяти минут до получаса, будьте терпеливы.

ЛАБОРАТОРНАЯ РАБОТА 1
Порты ввода-вывода общего назначения

Порты ввода-вывода общего назначения представляют собой сгруппированные выводы микроконтроллера, с помощью которых он может взаимодействовать с другими компонентами встроенных систем. Как следует из названия, каждый вывод порта может быть сконфигурирован на выход (для вывода сигнала) или на вход (для получения сигнала от других устройств). Порты ввода-вывода общего назначения оперируют логическими уровнями, так для микроконтроллеров архитектуры MSP430, напряжение на выводе в диапазоне V_{SS}[6] + 0,3 В обозначает «логический ноль», а в диапазоне V_{CC}[7] – – 0,3 В — «логическую единицу».

В данной лабораторной работе будет показано, как настроить порты ввода-вывода микроконтроллера MSP430G2553 для работы с внешней периферией: индикаторным светодиодом и тактовой кнопкой.

Последовательность действий при выполнении лабораторной работы:

Шаг 1. Откройте интегрированную среду разработки Code Composer Studio. Для операционной системы Windows 7: *Пуск → Все программы → Texas Instruments → Code Composer Studio 6.x.x → Code Composer Studio 6.x.x*. После запуска будет показано окно выбора пути рабочего окружения (директории, в которой будут храниться все проекты с программным кодом). Оставьте поле «Workspace» без изменений, подтвердите выбор кнопкой «ОК», откроется главное окно системы (рис. 3). Среда Code Composer Studio построена на базе системы разработки Eclipse,

[6] Условный потенциал нуля или «земля» (наименьшее отрицательное напряжение).

[7] Плюс питания (наибольшее положительное напряжение).

поэтому на устаревших компьютерах запуск может занять некоторое время.

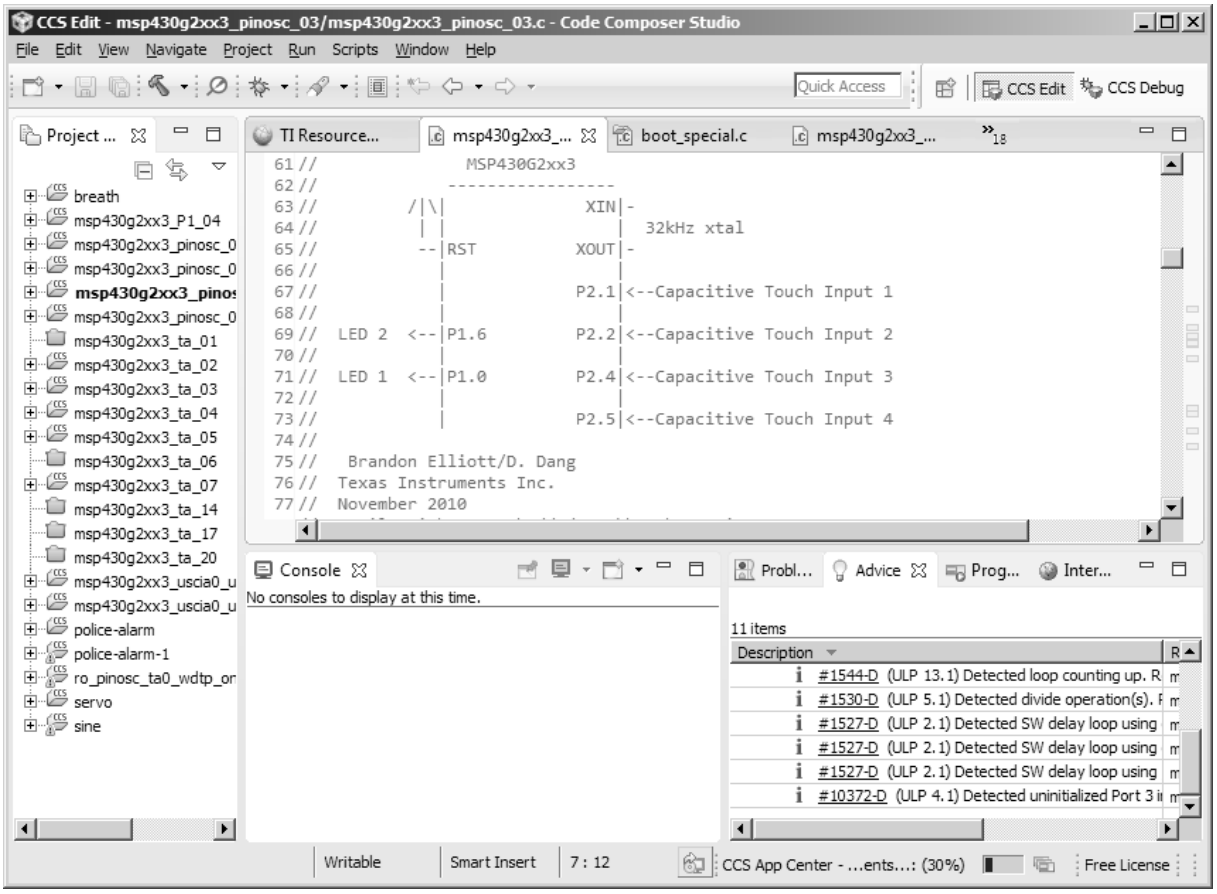


Рисунок 3. Главное окно среды Code Composer Studio

Шаг 2. Подключите отладочную плату к USB порту компьютера, должен загореться зеленый индикаторный светодиод с подписью «PWR».

Шаг 3. Создайте новый проект, выбрав следующие пункты главного меню: *File → New → CCS Project*, откроется окно настройки параметров проекта (рис. 4). В левом выпадающем меню «Target» выберите пункт «MSP430GXXX», а в правом — «MSP430G2553». В поле «Project name» введите произвольное имя проекта, например «hello-world», остальные поля оставьте без изменений. После этого подтвердите свои действия кнопкой «Finish». В левой части главного окна, в списке «Project Explorer» появиться новый проект.

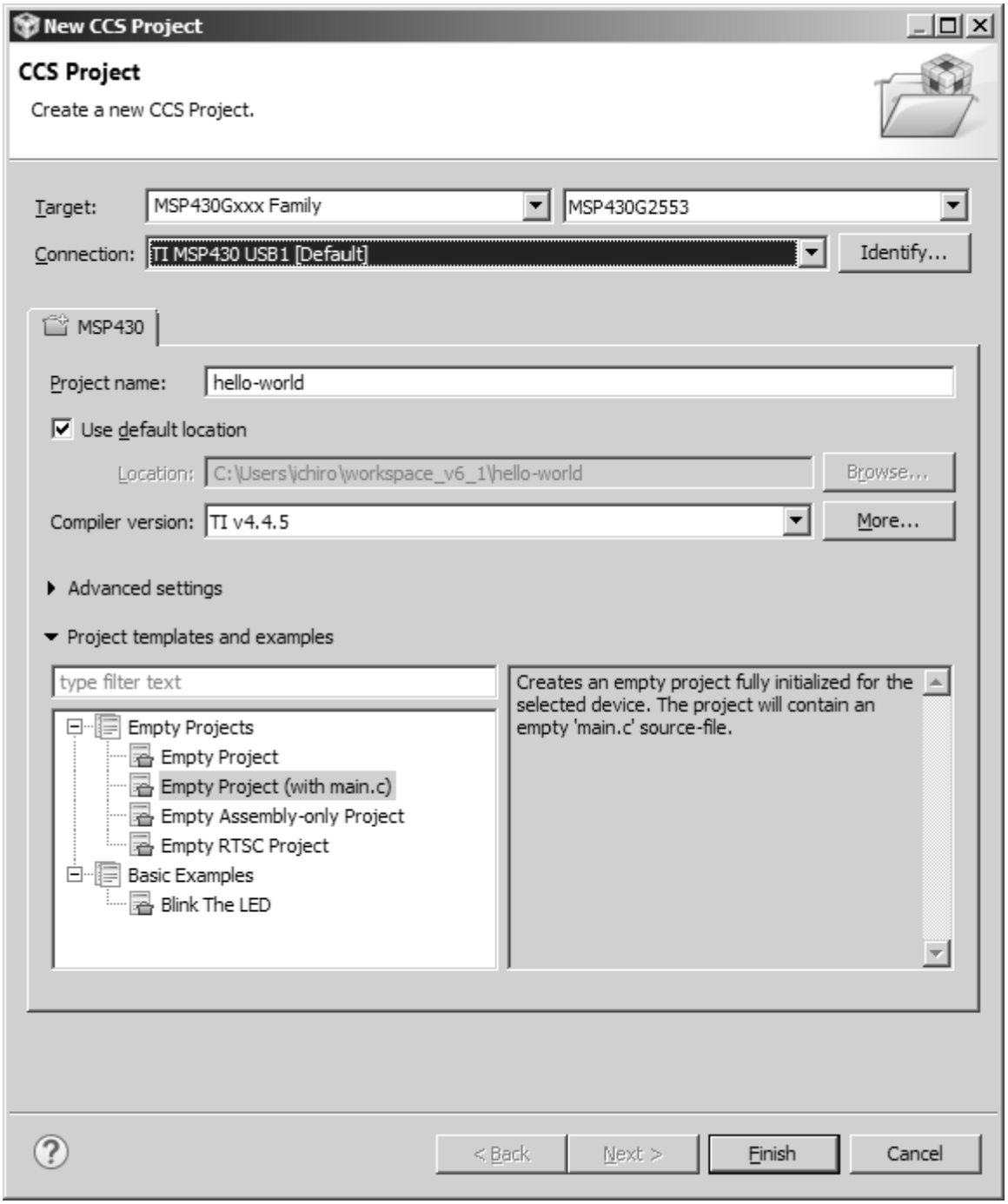


Рисунок 4. Окно настрой параметров проекта

Имя текущего рабочего проекта в системе Code Composer Studio выделяется полужирным шрифтом.

Шаг 4. После успешного создания нового проекта, в основном поле редактирования исходного кода автоматически создастся главный файл проекта. Традиционно для программного обеспечения, которое пишется на языке программирования C,

этот файл носит название «main.c». Рассмотрим более подробно содержимое этого файла (листинг 1).

Листинг 1

```
1 #include <msp430.h>
2
3 int main(void)
4 {
5       WDTCTL = (WDTPW | WDTHOLD);
6 }
```

Директива **#include <msp430.h>** подключает заголовочный файл с описанием периферийных устройств конкретного чипа семейства MSP430, выбранного на этапе конфигурирования проекта. В нашем случае будут автоматически использоваться настройки для MSP430G2553. В строке 3 определяется основная функция программы, задающая точку входа в программу. Она всегда должна называться «**main**».

В отличие от десктопных программ, программы для микроконтроллеров работают на «голом железе». Поэтому неважно значение какого типа будет возвращать эта функция и какие параметры будет принимать на вход. Все это имеет смысл только, если программа взаимодействует с операционной системой. Тем не менее, для удобства отладки рекомендуется в сигнатуре функции указывать целый тип возвращаемого значения (**int**). Директиву **return** можно опускать, предупреждений в логе компилятора при этом не возникает.

В строке 5 осуществляет настройка периферийного регистра **WDTCTL** [7, с. 347], отвечающего за работу сторожевого таймера. Сторожевой таймер (англ. *Watchdog timer*) — это аппаратно реализованная схема контроля над зависанием системы. Представляет собой таймер, который периодически сбрасывается контролируемой системой. Если сброса не произошло в течение некоторого интервала времени, происходит принудительная перезагрузка системы.

Битовая маска **(WDTPW | WDTHOLD)**, где **WDTPW** [7, с. 347] — восьмибитный пароль таймера (**0x5A00**), а **WDTHOLD** [7, с. 347] — флаг, останавливающий таймер. Отключение сторожевого таймера обязательно для любой программы, за исключением случаев, когда этот таймер используются не по прямому назначению. Значения, записанные в регистр **WDTCTL** по умолчанию таковы, что микроконтроллер будет перезагружен раньше, чем начнется выполнение основной программы.

Шаг 5. Запустите программу, нажав клавишу F11. Система Code Composer Studio автоматически скомпилирует вашу программу, соберет её и передаст на исполнение отладчику, который запишет её в память микроконтроллера и остановит поток выполнения на первой строке.

При этом изменится вид главного окна. Если говорить в терминах системы Eclipse, будет изменена перспектива. Для Code Composer Studio определено всего две перспективы: перспектива кода (обозначена «CCS Edit») и перспектива отладки («CCS Debug»). Перспектива отладки включает в себя окно исходного кода, окно с информацией о регистрах, выражениях и текущих переменных (рис. 5), окно процедур отладки (рис. 6), окно дисассемблера (рис. 7), окно терминала и вывода консоли (рис. 8).

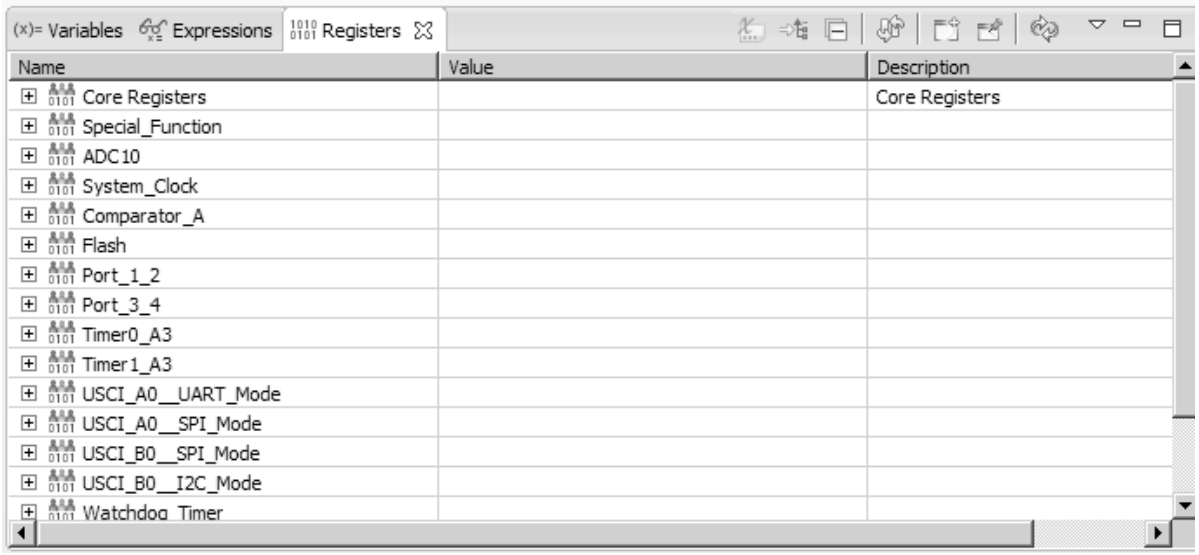


Рисунок 5. Окно информации о регистрах, переменных и выражениях

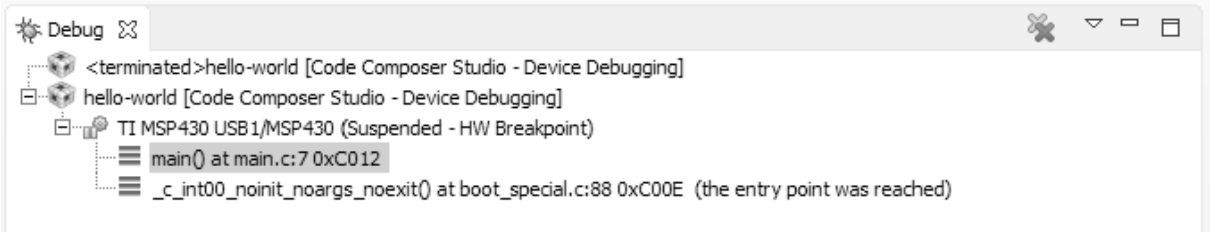


Рисунок 6. Окно процедур отладки

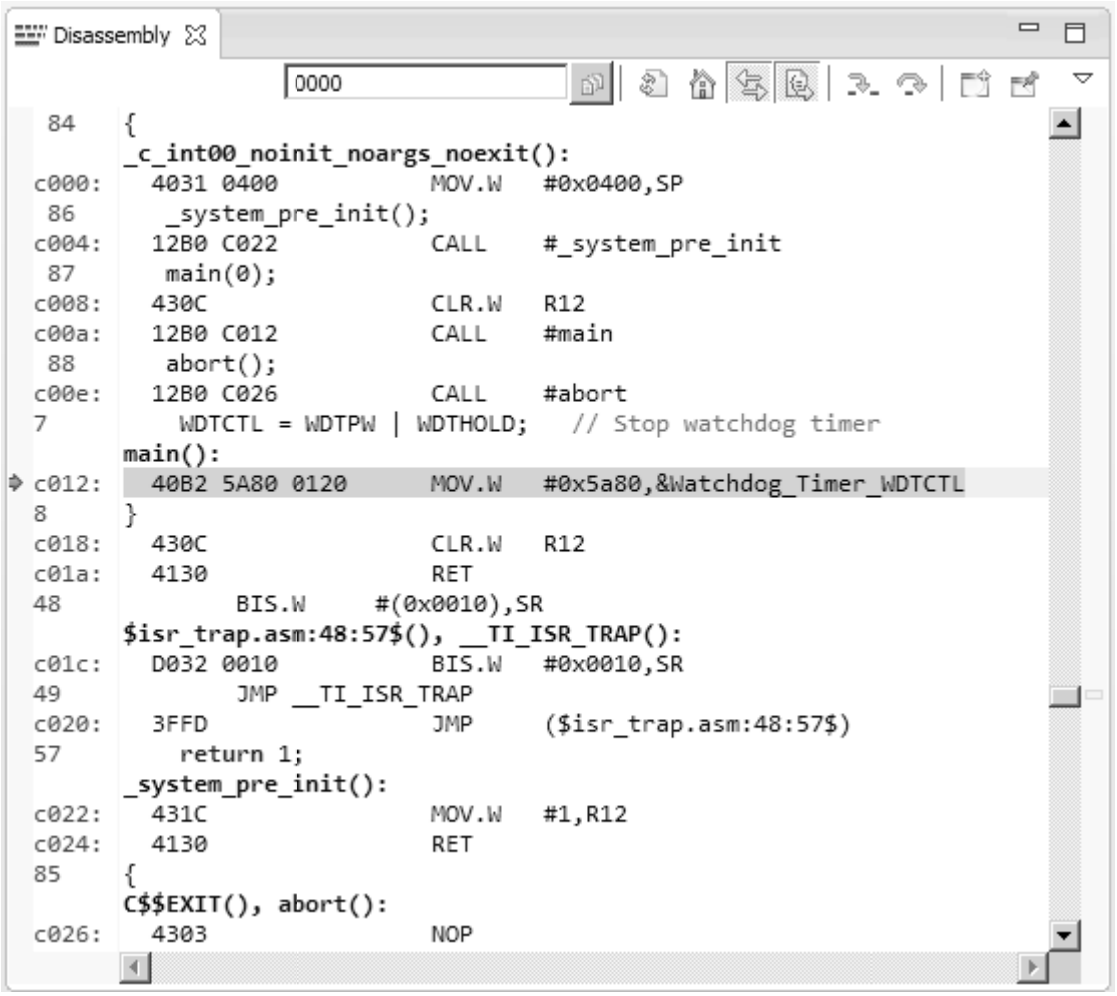


Рисунок 7. Окно дисассемблера

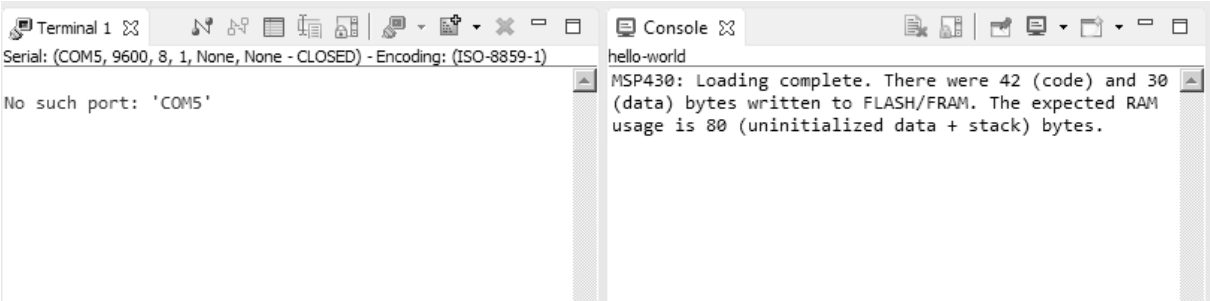


Рисунок 8. Окно терминала и вывода консоли

Например, в окне регистров вы можете увидеть значения, которые были записаны в регистр **WDTCTL**. Сейчас там нули, но, если вы нажмёте клавишу F5 (перейдете на один шаг вперёд по коду), значение флага **WDTHOLD** будет изменено на единицу. Изменившиеся биты регистра и весь регистр при этом будут окрашены в жёлтый цвет (рис. 9).

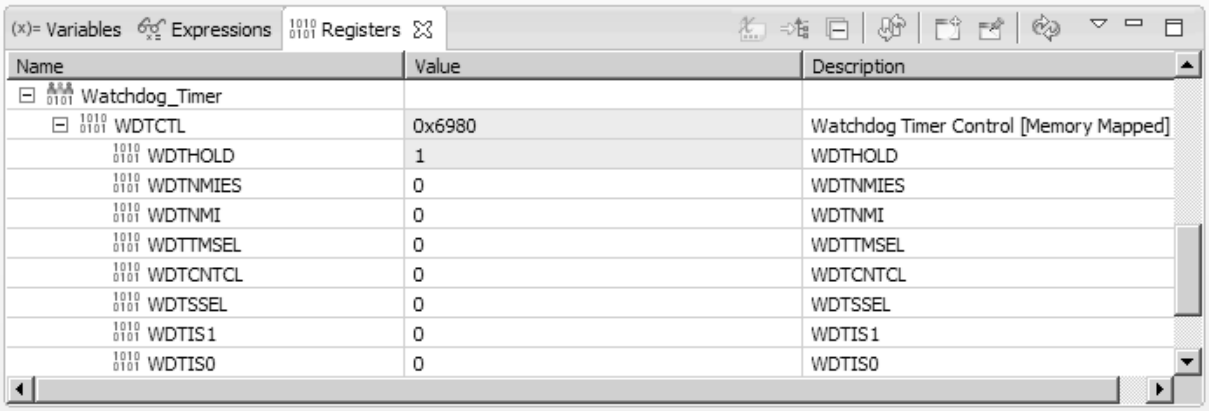


Рисунок 9. Содержимое регистра **WDTCTL** после изменения

Шаг 6. Переведите программу в режим постоянного выполнения, нажав клавишу F8. Теперь программа работает в бесконечном цикле, но ничего не делает. Измените код программы в соответствии с листингом 2.

Листинг 2

```
01 #include <msp430.h>
02
03 int main(void)
04 {
05      WDTCTL = WDTPW + WDTHOLD;
06      P1DIR |= BIT0;
07      for (;;) {
08          volatile unsigned int i;
09          i = 50000;
10          P1OUT ^= BIT0;
11          while (i--);
12      }
13 }
```

В строке 6 мы настраиваем направление работы нулевого вывода порта один. Как видно из кода, за это отвечает восьмибитный регистр **P1DIR** [7, с. 329]. Побитовая операция **|=** устанавливает в единицу те биты регистра, для которых в маске (переменная справа) соответствующие позиции также равны единице. Так, константа **BIT0** имеет значение **0x0001** (это значение задано в заголовочном файле **msp430.h**), поэтому получаем:

```
P1DIR = P1DIR | 0x0001
```

то есть вне зависимости, что было в нулевом бите регистра, после этой операции туда будет записана единица.

В строке 7 задаётся основной бесконечный цикл программы. В предыдущем примере кода его не было, но при внимательном рассмотрении дизассемблированного кода, можно заметить, что он добавляется компилятором автоматически. Хорошим тоном программирования микроконтроллеров является добавление пустого цикла в конец функции **main** в независимости от того пустой он или нет.

В строках 8 и 9 объявляется и определяется переменная-итератор, значение которой мы будет изменять ниже. Обратите внимание на использования класса памяти **volatile**. Без него компилятор уберёт («заоптимизирует») эту переменную, так как её значение используется только в условии пустого цикла **while**, что, по его мнению, бессмысленно.

В строке 10 мы по аналогии со строкой 6 меняем нулевой бит регистра **P1OUT** [7, с. 329], отвечающего за запись значений в порт один. Аналогично для получения значений из порта используется регистр **P1IN** [7, с. 329]. Направление порта (регистр **P1DIR**) можно менять динамически, по умолчанию все порты работают на вход (значение регистра **0x0000**). Побитовая операция **^=** инвертирует значение бита, заданного в маске **BIT0**, то есть если в нулевом бите регистра **P1OUT** была единица, туда запишется ноль и наоборот.

Строка 11 — бесконечный цикл, уменьшающий значение переменной-итератора **i**. Нужен для создания временно́й задержки.

Шаг 7. Перезапустите программу (F11 → Yes → F8). Если все сделано правильно, красный светодиод должен начать мигать с частотой примерно 0,5 Гц.

Шаг 8. Следующим шагом мы попытаемся поработать с выводом порта, получающим сигнал на вход. В качестве источника сигнала будет использоваться тактовая кнопка, подключенная к выводу P1.3 отладочной платы. Измените код проекта в соответствии с листингом 3.

Листинг 3

```
01 #include <msp430.h>
02 int main(void)
03 {
04      WDTCTL = (WDTPW | WDTHOLD);
05      P1DIR |= BIT0;
06      P1REN |= BIT3;
07      for (;;) {
08          while ((P1IN & BIT3) == BIT3);
09          P1OUT ^= BIT0;
10      }
11 }
```

В строке 6 мы включаем внутренний подтягивающий резистор для вывода 3 порта 1, то есть для кнопки. В соответствии со схемой отладочной платы, кнопка включена между соответствующим выводом микроконтроллера и землей. Включая внутренние резисторы, мы одновременно подключаем её и к питанию (V_{cc}).

Для использования кнопок в цифровой логике, они должны обеспечивать два состояния: логической единицы и логического ноля. Когда кнопка поднята, резистор «подтягивает» вывод к питанию, поднимая на нем напряжение до того уровня, которым запитана плата (в случае Launchpad это 3,6 В). Когда вы

нажимаете кнопку, вывод напрямую замыкается на землю. Таким образом, отпущенная кнопка — это логическая единица, нажатая — логический ноль.

В строке 8 мы ничего не делаем пока на выводе P1.3 высокий логический уровень, а как только он изменился, сразу меняем состояние светодиода и опять начинаем ждать нажатия кнопки. Такой режим работы называется опрос (англ. *polling*). Обратите внимание на то, что программа работает немного не так, как было запланировано. Кнопка иногда не срабатывает, иногда срабатывает дважды.

Это связанно с так называемым явлением «дребезга» контактов. Кнопка является механическим устройством, а ни одно механическое устройство не может быть совершенным настолько, чтобы срабатывать мгновенно. В момент нажатия возникаем переходный процесс, когда металлическая пластинка контакта замыкается со своей ответной частью и начинает слегка вибрировать (в силу инерции), что приводит к очень быстрому замыканию/размыканию контакта (рис. 10).

Эти множественные срабатывания приводят к неправильной интерпретации состояния кнопки. Существует два вида борьбы с дребезгом контактов: электрический, когда в схему добавляются новые элементы, способные подавить возникающий высокочастотный шум (например, RC-цепь), и программный (небольшая задержка для установления стабильного состояния).

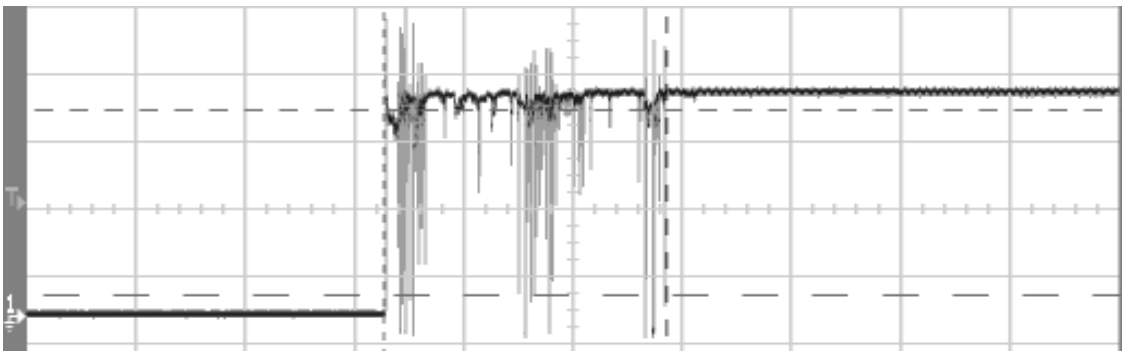


Рисунок 10. Осциллограмма дребезга контактов

ЛАБОРАТОРНАЯ РАБОТА 2
Прерывания и таймеры

Прерывания представляют собой сигналы о наступлении какого-либо события, которые периферийные устройства передают центральному процессору. При поступлении такого сигнала центральный процессор останавливает свою работу, сохраняя текущее состояние в оперативной памяти (как правило, сохраняется содержимое регистров), после чего управление передаётся процедуре обработчика прерывания. После того, как процедура выполнена, процессор восстанавливает свое состояние из памяти и продолжает работу.

Нельзя недооценивать роль прерываний во встроенных системах. Микроконтроллеры, в отличие от настольных систем, зачастую работают в автономном режиме, питаясь не от сети, что приводит к необходимости экономии электроэнергии.

Типовой сценарий работы микроконтроллера следующий. В основном цикле работы микроконтроллер ничего не делает, находясь в одном из режимов пониженного энергопотребления. Для микроконтроллеров семейства MSP430 существует пять таких режима, в самом экономном из которых микроконтроллер потребляет не более 100 нА энергии. Срабатывание одного из прерываний от периферийных устройств выводит его из состояния «сна» для выполнения кода обработчика, после чего он вновь переходит в режим пониженного энергопотребления.

Второй важной особенностью прерываний является возможность организации аппаратной многопоточности. Именно этот режим будет рассмотрен в данной лабораторной работе. Мы объединим код обоих примеров из предыдущей работы таким образом, что обработка нажатия кнопки будет в одном прерывании, а переключение состояния светодиода в другом. При этом частота переключения будет задаваться не

с помощью задержки в основном цикле, а посредством специального периферийного устройства, именуемого таймером.

Последовательность действий при выполнении лабораторной работы:

Шаг 1. Откройте проект из предыдущей работы и измените код в соответствии с листингом 4.

Листинг 4

```
01 #include <msp430.h>
02
03 int main(void)
04 {
05      WDTCTL = (WDTPW | WDTHOLD);
06      P1DIR |= BIT0;
07      TACCTL0 |= CCIE;
08      TACCR0 = 62500 - 1;
09      TACTL = (TASSEL_2 | MC_1 | ID_3);
10
11      __bis_SR_register(LPM0_bits | GIE);
12      while (1);
13 }
14 #pragma vector=TIMER0_A0_VECTOR
15 __interrupt void isr_timer_a(void)
16 {
17      P1OUT ^= BIT0;
18 }
```

В строке 7 настраивается блок **TACCTL0** [7, с.372], отвечающий за первый регистр захвата/сравнения таймера А (всего таких блоков три). Флаг **TACCTL0** разрешает прерывания от этого блока. Таймер — это просто счётный механизм, привязанный к импульсам тактового сигнала от системы синхронизации. Таймер А является шестнадцатибитным таймером. Это значит, что он считает от **0** до двоичного значения **0b1111111111111111**, или шестнадцатеричного **0xFFFF**, или десятичного **65535**. В строке 8 заполняется регистр **TACCR0** [7, с. 371]. В режиме сравнения в него записывается значение,

по достижении которого таймер должен подать сигнал центральному процессору, то есть **TACCR0** используется для указания верхней границы счёта.

В строке 9 записывается значение основного регистра управления таймером A **TACTL** [7, с. 370]. Значение **TASSEL_2** [7, с. 370] задаёт тактовый генератор. Для микроконтроллеров архитектуры MSP430 существует три вида тактовых генераторов:

1) ***MCLK*** [7, с. 24] — *Master Clock*, основной тактовый сигнал, он управляет вычислительным ядром и очередью команд программы.

2) ***SMCLK*** [7, с. 24] — *Sub-Main Clock*, дополнительный тактовый сигнал, используется для периферии. Его частота может совпадать с MCLK, а может отличаться, зависит от приложения.

3) ***ACLK*** [7, с. 24] — *Auxilliary Clock*, вспомогательный тактовый сигнал, обычно он генерируется извне микроконтроллера, и используется для периферии.

В данном случае тактирование будет происходить от **SMCLK**. Параметр **MC_1** задаёт режим прямого счёта от нуля до значения, записанного в регистре **TACCR0** с мгновенным возвратом в ноль (рис. 11). Также возможно считать в непрерывном режиме, то есть от нуля до **65535** (рис. 12), а также в режиме реверсивного счёта, то есть от нуля до значения регистра **TACCR0** и опять до нуля (рис. 13). Работа таймера начинается сразу после включения.

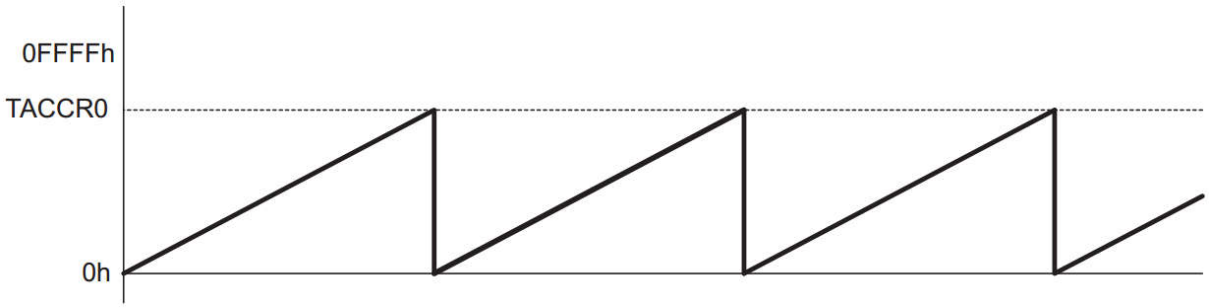


Рисунок 11. Режим прямого счёта таймера

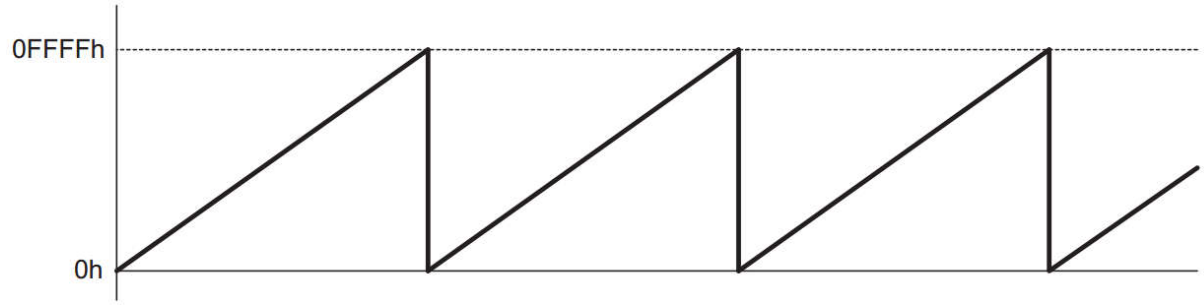


Рисунок 12. Режим непрерывного счёта таймера

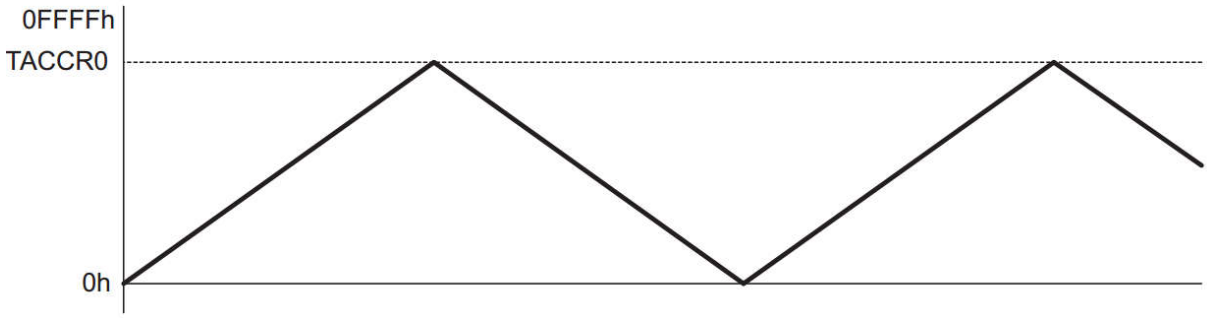


Рисунок 13. Режим реверсивного счёта таймера

Параметр **`ID_3`** задаёт делитель тактового сигнала, в данном случае частота тактового сигнала **`SMCLK`**, калиброванного на частоту примерно 1 МГц (встроенные источник тактового сигнала с цифровым управлением[8] является не слишком точным по сравнению с внешними резонаторами) будет поделена на 2^3, то есть составит ~125 КГц.

В строке 11 вызывается функция **`__bis_SR_register`**, задающая значения статусного регистра SR. Первым в неё передаётся флаг **`LPM0_bits`**, задающий нулевой уровень пониженного энергопотребления, при котором отключается центральный процессор и основной тактовый генератор **`MCLK`**, а **`SMCLK`** и **`ACLK`** остаются активными. Существуют еще четыре режима пониженного энергопотребления [7, с. 39]:

1) ***LPM1*** — Центральный процессор и тактовый сигнал MCLK выключены. Генератор постоянного тока выключён,

[8] англ. *Digital Clock Oscillator*, DCO.

если встроенные источник тактового сигнала с цифровым управлением не используется для формирования сигнала **SMCLK**, сигнал **ACLK** активен.

2) ***LPM2*** — Центральный процессор, тактовые генераторы MCLK, SMCLK, а также источник тактового сигнала с цифровым управлением выключены. **ACLK** и генератор постоянного тока активны.

3) ***LPM3*** — Центральный процессор, тактовые генераторы MCLK, SMCLK, а также источник тактового сигнала с цифровым управлением выключены. Генератор постоянного тока выключен, ACLK активен.

4) ***LPM4*** — Центральный процессор и все тактовые сигналы выключены, микроконтроллер находится в состоянии поддержания оперативной памяти.

Флаг **GIE** является флагом общего разрешения маскированных прерываний, то есть прерываний, которые можно запрещать установкой соответствующих битов в регистре управления.

В строках 14 и 15 определяется процедура обработчика прерываний. **TIMER0_A0_VECTOR** — адрес вектора прерывания. Имя обработчика выбирается произвольно, но хорошим тоном считается включать в это имя префикс ISR (сокр. от англ. *Interrupt Service Routine* — процедура обработки прерывания). В теле обработчика прерывания уже традиционно изменяется состояние светодиода.

Шаг 2. Запустите программу на исполнение, красный светодиод должен начать мигать с частотой равной:

$$\frac{SMCLK/ID\_3}{2 \times 62500} = \frac{1000000\ \text{МГц}/8}{125000} \cong 1\text{Гц}$$

Шаг 3. Прерывания могут быть вызваны любым периферийным устройством, являющимся частью рассматриваемого микроконтроллера. Измените код программы в соответствии

с листингом 5, чтобы реализовать обработчик прерывания от порта ввода вывода общего назначения.

Листинг 5

```
01 #include <msp430.h>
02
03 int main(void)
04 {
05      WDTCTL = (WDTPW | WDTHOLD);
06      P1DIR = BIT0;
07      P1OUT = BIT3;
08      P1REN |= BIT3;
09      P1IE |= BIT3;
10      P1IES |= BIT3;
11      P1IFG &= ~BIT3;
12      __bis_SR_register(LPM4_bits | GIE);
13 }
14 #pragma vector=PORT1_VECTOR
15 __interrupt void isr_port_1(void)
16 {
17   P1OUT ^= BIT0;
18   P1IFG &= ~BIT3;
19 }
```

Строки 1–8 не требуют пояснений, в строке 9 устанавливается флаг, разрешающий прерывания от вывода 3 порта 1, то есть для тактовой кнопки. В строке 10 устанавливается флаг задающий фронт импульса, в данном случае прерывание будет происходить при переходе из высокого логического состояния в низкое, то есть по заднему фронту сигнала.

В строке 11 очищается флаг прерывания, если этого не сделать, обработчик сработает сразу. В строке 12 устанавливается флаг общего разрешения прерываний, а центральный процессор переходит в режим пониженного энергопотребления. В строках 14 и 15 задаётся определение обработчика прерывания, а в строках 16–19 тело обработчика. Обратите внимание на необходимость каждый раз очищать флаг прерывания

по выходу из обработчика, если этого не делать, микроконтроллер зациклится в этом обработчике.

Задание для самостоятельной работы: напишите программу, которая реализует логику работы трехцветного светофора (красный → красный с жёлтым → зелёный → зелёный мигающий → жёлтый → красный). При нажатии на тактовую кнопку светофор должен переходить в ночной режим (жёлтый мигающий), при повторном нажатии — возвращаться в обычный режим.

Для выполнения работы соберите следующую электрическую схему (рис. 14 и 15).

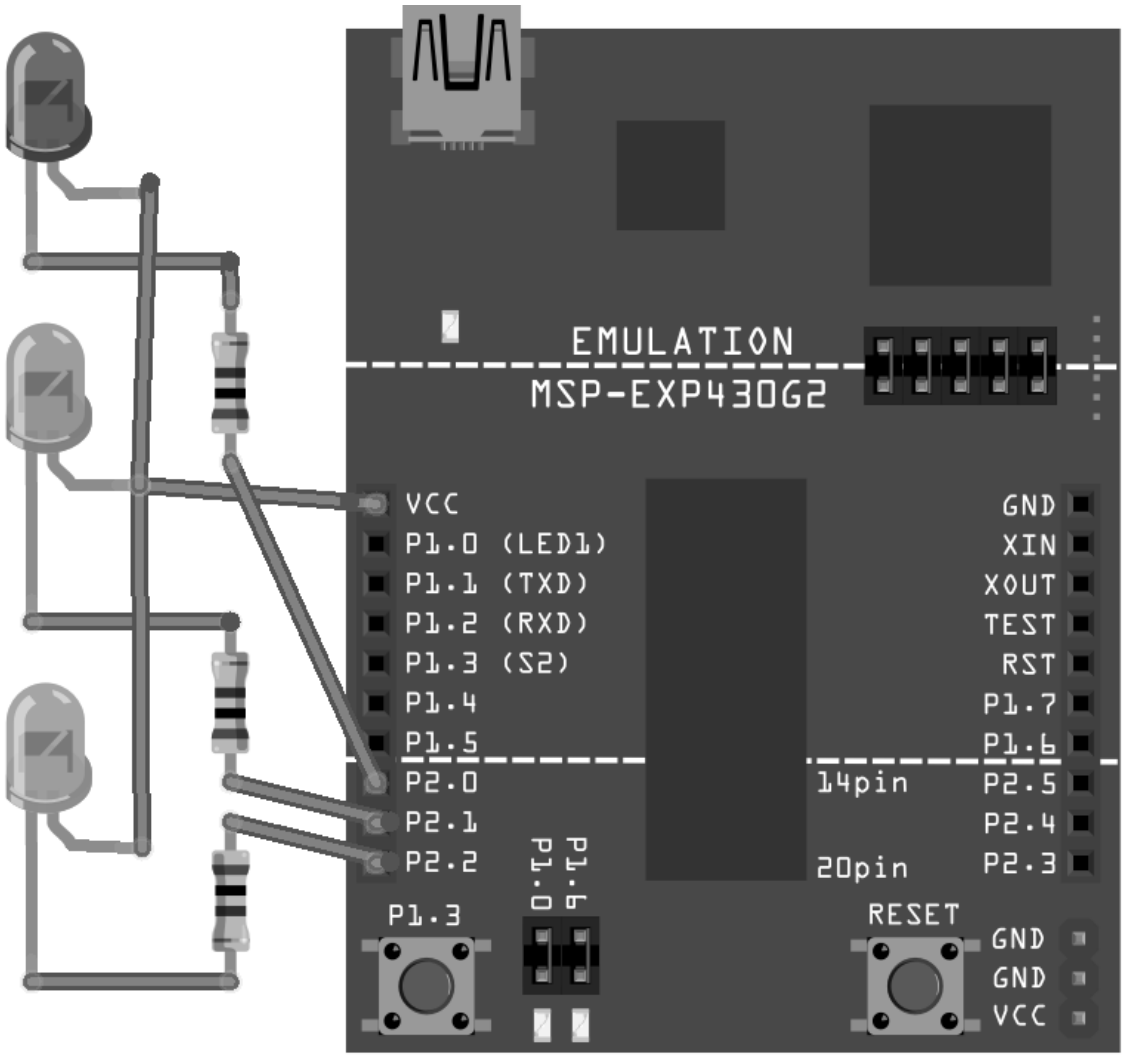


Рисунок 14. Схема подключения светодиодов

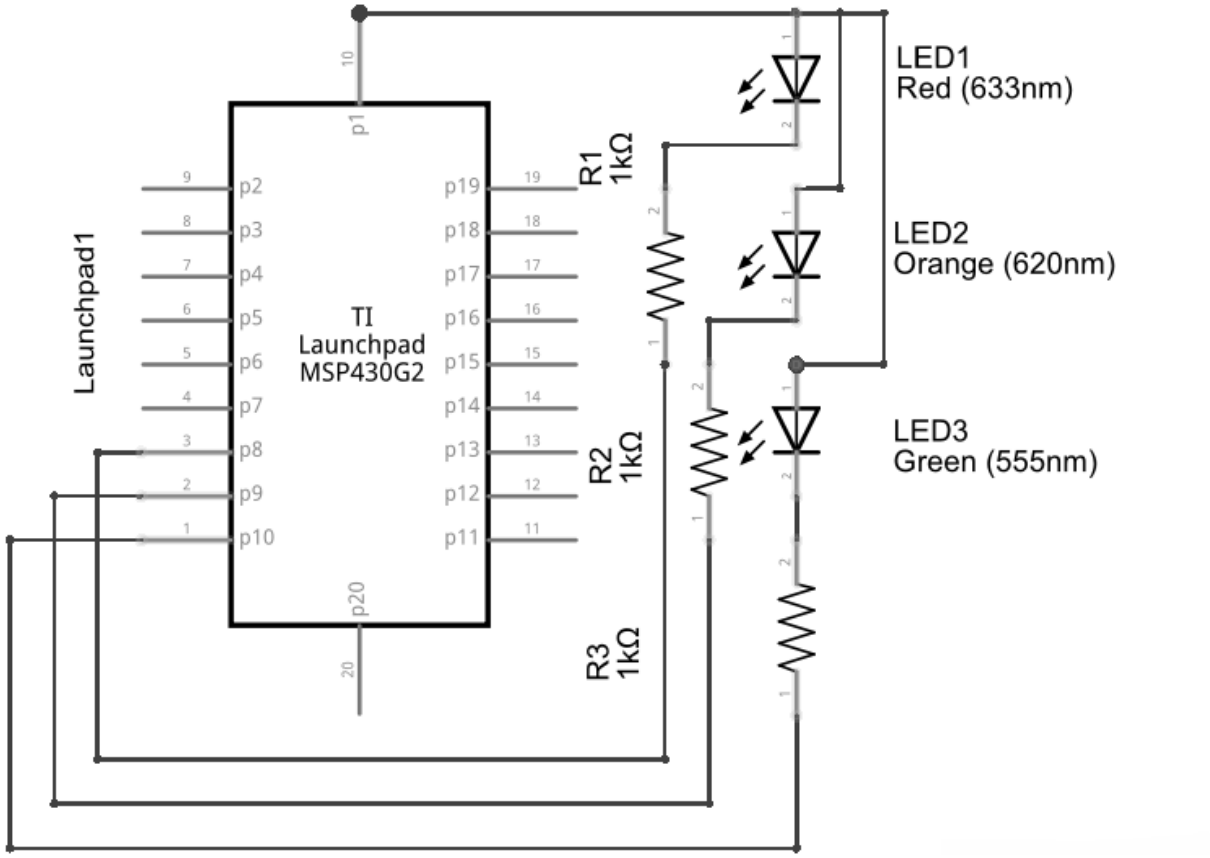


Рисунок 15. Принципиальная электрическая схема светофора

Программный код должен быть написан с применением прерываний, рекомендуется построить его по принципу конечного автомата с применением оператора **`switch`**. Более подробную информацию о проектировании конечных автоматов можно в [8].

ЛАБОРАТОРНАЯ РАБОТА 3
Широтно-импульсная модуляция

Широтно-импульсная модуляция (ШИМ) представляет собой метод изменения мощности, подводимой к нагрузке, посредством изменения скважности импульсов при неизменной частоте. В наиболее общем смысле, ШИМ — это процесс быстрого переключения состояния вывода порта ввода-вывода, при котором усредненное напряжение на выводе пропорционально отношению времени, когда на вывод подается высокий логический уровень, к времени, когда подаётся низкий (рис. 16).

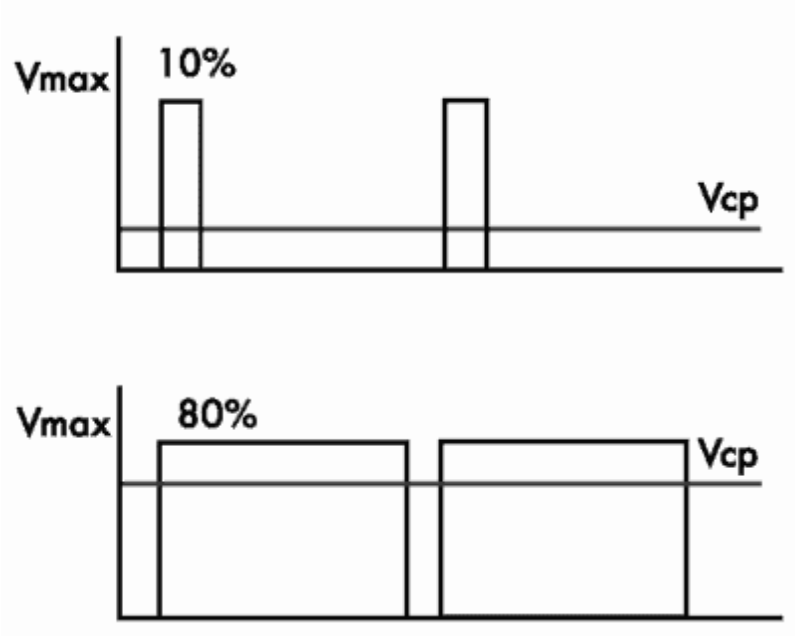


Рисунок 16. Широтно-импульсная модуляция

Например, при частоте модуляции 50 Гц (период 20 мс) и скважности 10 %, после фильтрации ($V_{ср}$) на выводе микроконтроллера мы получим разность потенциалов равную 3,6 В × 0,1 = 36 мВ, а при скважности 80 % — 2,88 В.

Широтно-импульсная модуляция находит самое широкое применение при проектировании встроенных систем, что

обуславливается простой реализации «в железе», а также возможностью гибкого управления вторичными источниками питания с высоким КПД. Также ШИМ используется в задачах управления, например, недорогие сервоприводы и контроллеры бесколлекторных двигателей принимают на вход ШИМ сигнал, скважность которого определяет угол поворота сервопривода или скорость вращения мотора соответственно.

В данной лабораторной работе будут рассмотрены различные режимы работы ШИМ в микроконтроллерах архитектуры MSP430.

Последовательность действий при выполнении лабораторной работы:

Шаг 1. Модифицируйте код программы в соответствии с листингом 6.

Листинг 6

```
01 #include <msp430.h>
02
03 int main(void)
04 {
05      WDTCTL = (WDTPW | WDTHOLD);
06      P1DIR |= BIT0;
07      TA0CCTL0 = CCIE;
08      TA0CCTL1 = CCIE;
09      TA0CCR0 = 10000 - 1;
10      TA0CCR1 = 100;
11      TA0CTL = (TASSEL_2 | MC_1);
12      __bis_SR_register(LPM0_bits | GIE);
13 }
14 #pragma vector=TIMER0_A0_VECTOR
15 __interrupt void isr_timer_a0(void)
16 {
17     P1OUT |= BIT0;
18 }
19 #pragma vector=TIMER0_A1_VECTOR
20 __interrupt void isr_timer_a1(void)
21 {
```

```
22      switch (TA0IV) {
23      case 2:
24          P1OUT &= ~BIT0;
25          break;
26      case 4: break;
27      case 10: break;
28      }
29 }
```

В листинге представлена реализация так называемой программной широтно-импульсной модуляции (англ. *Software PWM*), в данном случае используются только стандартные обработчики прерываний от таймера.

В строках 7 и 8 разрешаются прерывания от первого и второго регистров захвата сравнения. Всего таймер A имеет три регистра захвата сравнения, причем первый (**TA0CCR0**) задаёт период в режимах прямого и обратного счёта, а два остальных могут быть использованы для формирования более одного события на период. Отдельно обрабатывается событие переполнения счётчика таймера в режиме непрерывного счёта, о чём будет сказано ниже.

В строке 9 задаётся период ШИМ, в строке 10 — скважность. Для данного случая скважность равна 10 %. Вы можете менять значение регистра **TA0CCR1** по своему усмотрению, единственное ограничение — оно не должно превышать период. В 11 строке конфигурируется режим работы таймера A: тактирование от **SMCLK** и режим прямого счёта. В строке 12 поднимается флаг общего разрешения прерываний, а центральный процессор переводится в первый режим пониженного энергопотребления.

Далее по коду идут два обработчика прерываний от таймера A. Здесь необходимо пояснить, как организована обработка прерываний для вспомогательных регистров захвата сравнения. Первый обработчик соответствует регистру перио-

да **TA0CCR0**, а второй — **TA0CCR1**, **TA0CCR2** и флага переполнения. Все они объединены в один вектор **TA0IV**.

Обработчик первого прерывания аналогичен подобному из предыдущей работы. В нём мы просто устанавливаем высокий уровень сигнала на выводе, соединенном со светодиодом, то есть инициируем начало импульса. Второй обработчик содержит код проверки вектора **TA0IV**, реализованный с помощью оператора switch. Если значение вектора равно **2** (двоичное представление **0b0000000000000010**), значит счётчик достиг значения, записанного в регистре **TA0CCR1**. Возникло событие, по наступлению которого мы можем выполнить какой-либо произвольный код. В данном случае мы переводим вывод на низкий логический уровень, завершая импульс ШИМ. После чего принудительно выходим из обработчика (оператор **break**).

Аналогичным образом мы можем обработать события, возникающие при достижении счётчиком значения регистра **TA0CCR2** (**case 4**, строка 26) и по переполнению счётчика (**case 10**, строка 27). Может показаться, что строки 26 и 27 не нужны, так как мы не обрабатываем соответствующие случаи, тем не менее, хорошим тоном программирования микроконтроллеров являет создание таких пустых обработчиков. Это может помочь при отладке сложных программ.

В литературе также рекомендуется добавлять в эти обработчики пустые бесконечные циклы, а также создавать обработчики с такими циклами для всех реализованных в микроконтроллере векторов прерываний. Это обуславливается тем, что в случае аппаратного сбоя (сработало неинициализированное прерывание, причина неизвестна), центральный процессор перейдет в обработчик и «зависнет» там в бесконечном цикле, что будет легко отследить в отладчике.

Данный метод часто используется в программировании и называется «максимизацией ошибки». Программа не должна

«замалчивать» свои сбои, любой сбой должен приводить к полному краху программы. Так его гораздо легче будет воспроизвести, а значит проще и быстрее исправить. Общее правило здесь следующее: лучше пусть программа не работает вообще, чем работает иногда правильно, иногда неправильно и сложно определить из-за чего это происходит.

Шаг 2. Модифицируйте код программы в соответствии с листингом 7.

Листинг 7

```
01 #include <msp430.h>
02
03 int main(void)
04 {
05      WDTCTL = (WDTPW | WDTHOLD);
06      P1DIR |= BIT2;
07      P1SEL |= BIT2;
08      CCR0 = 512 - 1;
09      CCTL1 = OUTMOD_7;
10      CCR1 = 384;
11      TACTL = (TASSEL_2 | MC_1);
12
13      __bis_SR_register(CPUOFF);
14 }
```

В листинге рассматривается пример аппаратной реализации широтно-импульсной модуляции (англ. *Hardware PWM*). В данном случае за формирование импульсов отвечает блок таймера A, работающий автономно и асинхронно по отношению к центральному процессору. Вывод сигнала осуществляется за счёт использования альтернативной функции порта ввода вывода общего назначения.

В строке 7 задается альтернативная функция порта P1.2. В соответствии с листом данных на процессор MSP430G2553 для чипа в корпусе DIP20 этот порт связан с Timer0_A Out1 [9, с. 6]. В строке 8 в регистр **CCR0** записывается значение

периода. В строке 9 настраиваются параметры второго регистра захвата сравнения. Всего существует 8 режимов ШИМ:

1) ***OUTMOD_0*** — вывод.
2) ***OUTMOD_1*** — установка.
3) ***OUTMOD_2*** — переключение/сброс.
4) ***OUTMOD_3*** — установка/сброс.
5) ***OUTMOD_4*** — переключение.
6) ***OUTMOD_5*** — сброс.
7) ***OUTMOD_6*** — переключение/установка.
8) ***OUTMOD_7*** — сброс/установка.

Графическое представление всех рассмотренных видов ШИМ сигнала для случая, когда таймер работает в режиме прямого счёта дано на рис. 17.

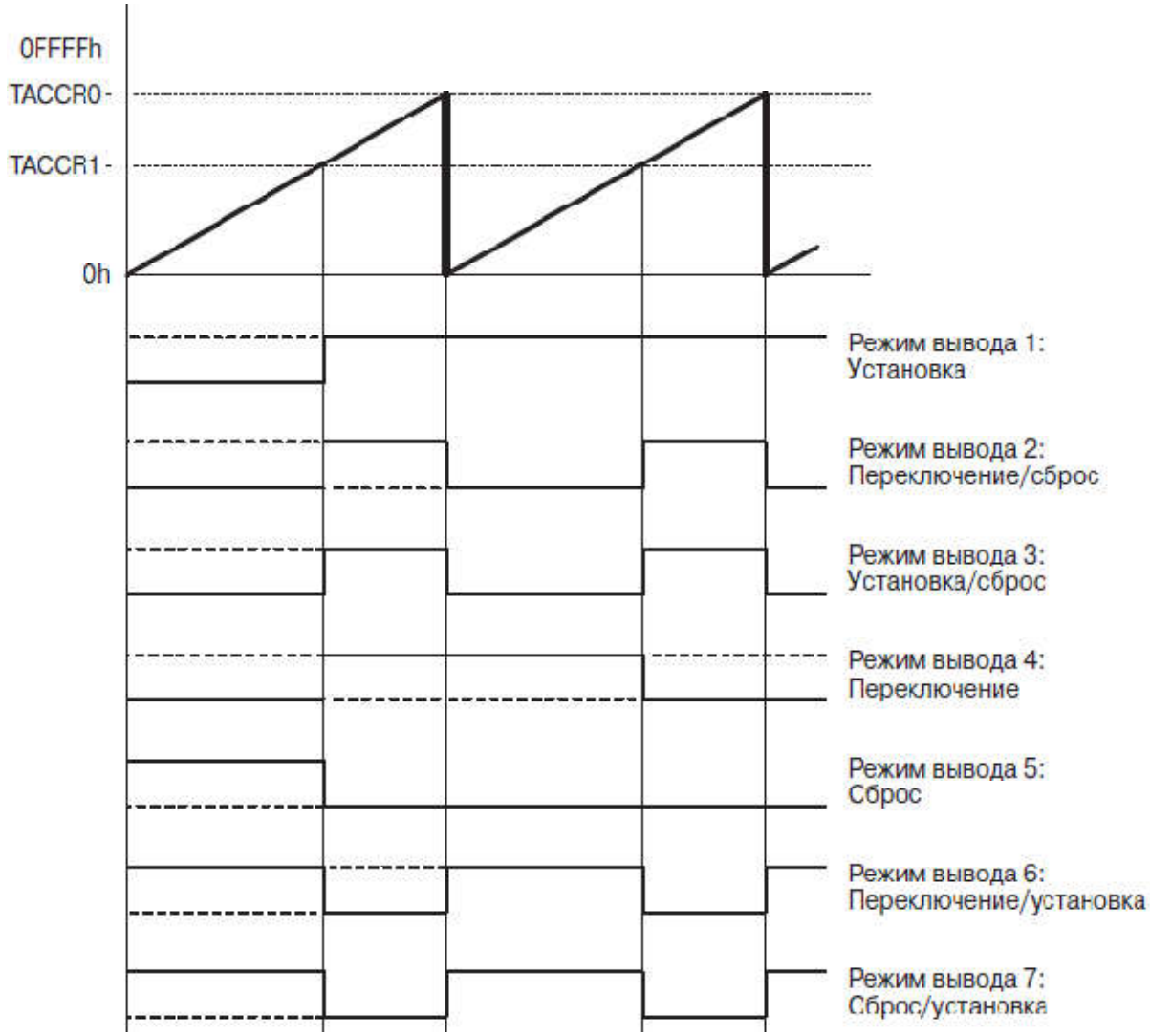


Рисунок 17. Режимы формирования ШИМ

В строке 10 задаётся значение регистра **CCR1**, определяющего скважность сигнала, в данном примере скважность равна 75 %. Увидеть данное значение можно подключив к выводу P1.2 индикаторный светодиод по схеме из предыдущей работы, либо воспользовавшись измерительными приборами: в простейшем случае достаточно вольтметра, при желании увидеть форму и длительность сигнала рекомендуется использовать логический анализатор или осциллограф (последний предпочтительнее).

В строке 11 задаётся тактирование от генератора **SMCLK** и режим прямого счёта. В строке 12 центральный процессор отключается, то есть переходит в первый режим пониженного энергопотребления. **CPUOFF** является синонимом **LPM0_bits**. Прерывания выключены.

Задание для самостоятельной работы: напишите программу, которая плавно меняет цвет трёхцветного (RGB) светодиода в соответствии с распределением цветов в спектре. Результат должен быть примерно как в данном видео: https://youtu.be/jBhR3cS1jrk.

Для выполнения работы соберите следующую электрическую схему (рис. 18 и 19).

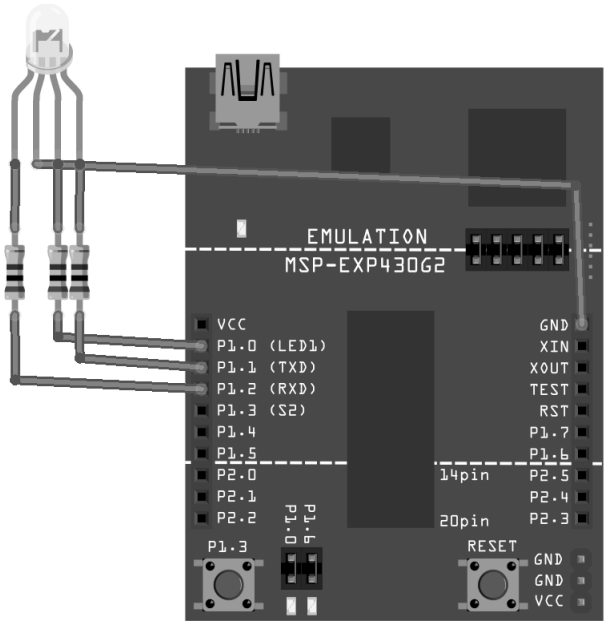


Рисунок 18. Схема подключения трёхцветного светодиода

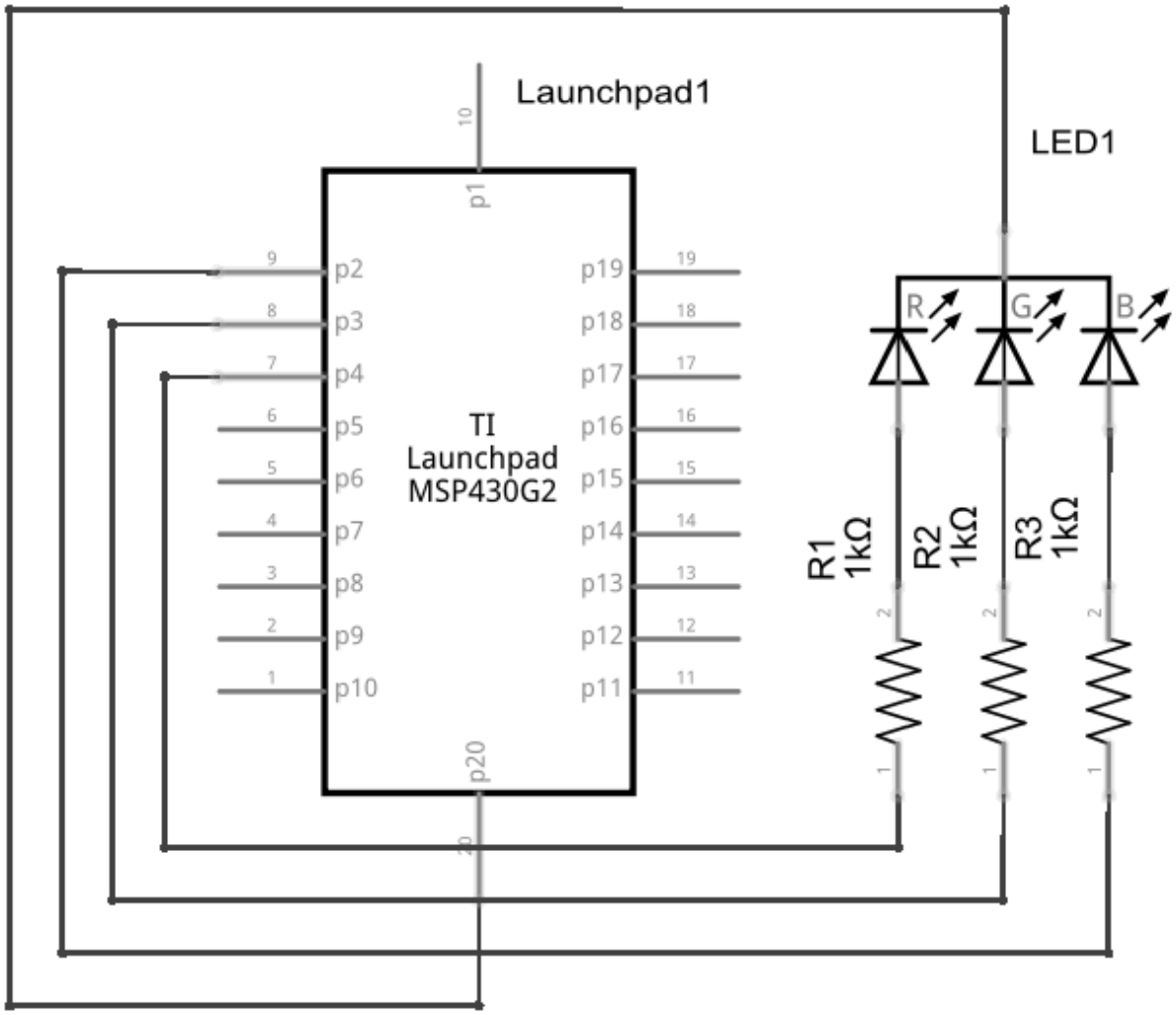


Рисунок 19. Принципиальная электрическая схема подключения трёхцветного светодиода

Обратите внимание на то, что вольтамперная характеристика светодиода нелинейна, поэтому если вы просто будете линейно менять скважность импульсов, вы никогда не получите плавного перехода между цветами, вместо этого будут резкие скачки яркости. Подумайте, как реализовать нелинейное изменение скважности.

Первая версия программы должна использовать программную реализацию ШИМ с применением прерываний таймера A, вторая — комбинацию прерываний и аппаратной ШИМ.

ЛАБОРАТОРНАЯ РАБОТА 4
Аналого-цифровой преобразователь

Аналогово-цифровой преобразователь — это периферийное устройство, позволяющее переводить непрерывный сигнал в последовательность дискретных значений с определённой частотой дискретизации и разрешением. Разрешение определяет количество уровней квантования на одну выборку сигнала. Как правило, с помощью АЦП измеряется напряжение на входе микроконтроллера. Например, для размаха в 3,6 В и разрешения в 10 бит на одну выборку существует 1024 уровня квантования, что равно разрешению в 3,5 мВ. В реальных приложениях редко удаётся достичь заявленного разрешения по причине шума.

Принято считать, что для большинства неспециализированных АЦП без прецизионных источников опорного напряжения как минимум младший бит содержит шум. С ним можно бороться программным (фильтрация, например усреднение) или аппаратными (использование внешнего стабильного и точного источника опорного напряжения) средствами, либо их комбинацией. Также можно просто отбрасывать (обнулять) биты, содержащие шум, снижая тем самым разрешение. Выбор конкретного метода зависит от назначения встраиваемой системы, требований, предъявляемых к ней, а также возможностей микроконтроллера.

Существует несколько разновидностей АЦП: последовательные прямого преобразования, последовательного приближения, последовательные с сигма-дельта модуляцией, параллельные одноступенчатые, параллельные двух- и более ступенчатые (конвейерные).

В микроконтроллерах с архитектурой MSP430 реализован АЦП последовательного приближения. Принцип его работы достаточно прост. Вначале измеряемое напряжение сравнивается

с половинным опорным напряжением посредством встроенного одноразрядного компаратора. Как известно, компаратор представляет собой устройство, принимающее на вход два сигнала и выдающее высокий логический уровень, если один из них больше другого по величине, а низкий, если наоборот. Таким образом, когда исследуемое напряжение больше половины опорного, старший бит результата дискретизации устанавливается в единицу, если меньше — в ноль.

Затем опорное напряжение устанавливается на 1/4 или 3/4 от базового в зависимости от предыдущего результата и вновь сравнивается с исследуемым сигналом. При этом вычисляется следующий бит результата оцифровки. Используя такой метод, можно получить цифровое значение любой разрядности в зависимости от того, сколько измерений было сделано. В случае десятибитного аналогово-цифрового преобразователя процессоров семейства MSP430G2 проводится десять таких измерений. Максимальная скорость преобразования достигает 200 тысяч выборок в секунду.

В данной лабораторной работе будет рассмотрена реализация оцифровки сигнала от встроенного в микроконтроллер температурного сенсора.

Последовательность действий при выполнении лабораторной работы:

Шаг 1. Модифицируйте код программы в соответствии с листингом 8.

Листинг 8

```
01 #include <msp430.h>
02 #define ADC_DELTA       3
03 static unsigned int first_adc_val;
04 int main(void)
05 {
06     WDTCTL = (WDTPW | WDTHOLD);
07     ADC10CTL1 = (ADC10DIV_3 | INCH_10 |
                    SHS_1 | CONSEQ_2);
```

```
08      ADC10CTL0 = (SREF_1 | ADC10SHT_3 | REF2_5V |
                     ADC10IE | REFON | ADC10ON);
09      __enable_interrupt();
10      TACCR0 = 30;
11      TACCTL0 |= CCIE;
12      TACTL = (TASSEL_2 | MC_1);
13      LPM0;
14      TACCTL0 &= ~CCIE;
15      __disable_interrupt();
16      ADC10CTL0 |= ENC;
17      TACCTL1 = OUTMOD_4;
18      TACTL = (TASSEL_2 | MC_2);
19      while (!(ADC10IFG & ADC10CTL0));
20      first_adc_val = ADC10MEM;
21      P1OUT = BIT0;
22      P1DIR |= BIT0;
23      __bis_SR_register(LPM0_bits | GIE);
24  }
25  #pragma vector=ADC10_VECTOR
26  __interrupt void isr_adc10(void)
27  {
28      if (ADC10MEM >= first_adc_val + ADC_DELTA)
29          P1OUT |= BIT0;
30      else
31          P1OUT &= ~BIT0;
32  }
33  #pragma vector=TIMER0_A0_VECTOR
34  __interrupt void isr_ta0(void)
35  {
36      TACTL = 0;
37      LPM0_EXIT;
38  }
```

В листинге представлена реализация анализатора градиента температуры. Если за определённый период времени, температура, регистрируемая датчиком, увеличивается больше чем на 2 °C по сравнению с первым измерением, загорается

красный индикаторный светодиод. После возвращения температуры в изначальный диапазон, светодиод гаснет.

Для управления периферийным блоком АЦП используется два регистра управления: **ADC10CTL0** и **ADC10CTL1** [7, с. 477]. В строке 7 задаются параметры регистра **ADC10CTL1**. Выбирается десятый канал преобразования — **INCH_10**, всего есть 8 внешних каналов и 4 внутренних, десятый соединен с внутренним датчиком температуры.

Флаг **CONSEQ_2** задаёт циклический одноканальный режим преобразования. Также существует однократный одноканальный (**CONSEQ_0**), однократный последовательный (CONSEQ_1) и циклический последовательный (**CONSEQ_3**) режимы преобразования.

ADC10DIV_3 задаёт делитель тактового сигнала АЦП, в данном случае — 8. **SHS_1** — источник сигнала запуска выборки/преобразования, в нашем случае используется модуль вывода 1 таймера А.

Для регистра **ADC10CTL0** задаются следующие флаги. **SREF_1** —положительное опорное напряжение задаётся внутренним источником, отрицательное — потенциалом земли микроконтроллера. **ADC10SHT_3** — время выборки, равное 64 тактам сигнала **ADC10CLK**, по умолчанию внутренний генератор модуля АЦП.

REF2_5V — величина опорного напряжения 2,5 В. **ADC10IE** — разрешить прерывания от модуля АЦП. **REFON** — включить генератор опорного напряжения. **ADC10ON** — включить модуль АЦП.

В строке 9 вызывается функция **__enable_interrupt**, разрешающая прерывания без перехода в режим пониженного энергопотребления. В строке 10 задаётся период для таймера А. В строках 11 и 12 разрешаются прерывания от этого таймера, после чего он запускается. В строке 13 контроллер

переводится в первый режим пониженного энергопотребления (с прерываниями, которые уже были разрешены ранее).

Логика работы здесь следующая: отсчёт таймера задаёт небольшую задержку при старте микроконтроллера, которая позволяет установиться значению опорного источники напряжения. По достижению счётчиком таймера значения **CCR0** происходит прерывание, управление передаётся соответствующему обработчику (строки 33–38). В обработчике таймер останавливается (в регистр **TACTL** записывается 0), а микроконтроллер выходит из режима пониженного энергопотребления. После чего мы запрещаем прерывание от регистра **CCR0** (строка 14) и опускаем флаг общего разрешения прерываний (строка 15).

В строке 16 мы разрешаем начать преобразования, устанавливаем режим переключения для **CCR1**, так как выше мы задали вывод 1 таймера A в качестве источника сигнала запуска выборки/преобразования (строка 17), а затем запускаем таймер A в режим непрерывного счёта (строка 18).

В строке 19 ждём первого сигнала от блока АЦП, то есть значения первого преобразования и записываем его в переменную **first_adc_val** (строка 20). Обратите внимание на то, что в этот момент опущен флаг общего разрешения прерываний, значит, хотя периферийный модуль АЦП и подал сигнал о произошедшем прерывании, оно не будет обработано, а мы не перейдем в функцию обработчика.

После записи первого значения преобразования мы настраиваем вывод светодиода на выход, обнуляем его (светодиод не должен гореть на старте) и переходим в режим пониженного энергопотребления с прерываниями.

В обработчике прерывания от модуля АЦП (строки 25–32) мы сравниваем текущее значение преобразования **ADC10MEM** с первоначальным значением (переменная **first_adc_val**) плюс **ADC_DELTA**, задающее приращение температуры

по сравнению с изначальным. Значение 3 примерно равно двум градусам Цельсия.

Шаг 2. Запустите программу на исполнение. Прикоснитесь пальцем к корпусу микроконтроллера так, чтобы площадь соприкосновения была максимальной или поднесите к чипу любой источник тепла (ни в коем случае не используйте открытый огонь!). Дождитесь, пока температура установится, то есть светодиод перестанет мигать.

Мигание демонстрирует упомянутые выше шумы, когда в процессе одного измерения из-за погрешностей оцифровки температура уже выше порогового значения, а на следующем — нет. Уберите источник тепла, дождитесь выключения светодиода.

Задание для самостоятельной работы. Напишите программу, которая линейно изменяет яркость встроенного светодиода при изменении положения подключенного к микроконтроллеру потенциометра. Для достижения линейного изменения яркости используйте код из предыдущей работы. Для выполнения работы соберите следующую электрическую схему (рис. 20).

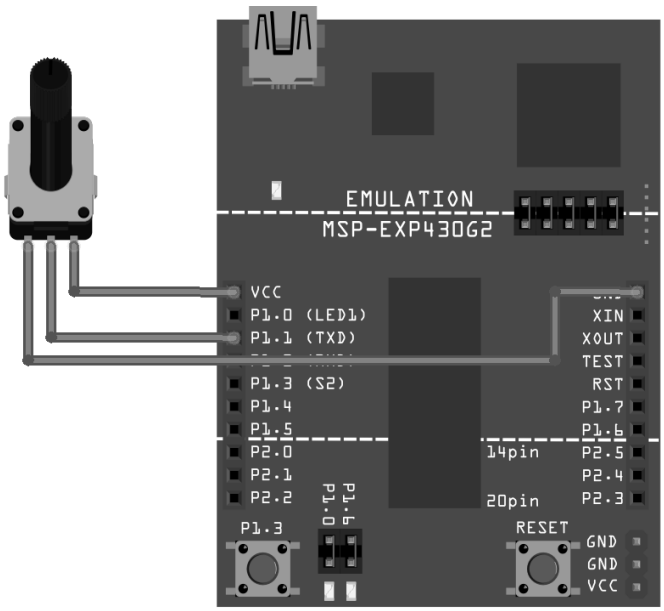


Рисунок 20. Схема подключения потенциометра

ЗАКЛЮЧЕНИЕ

Это учебное пособие — лишь начало разговора о многогранном мире интеллектуального производства.

Все более и более человек проникает в глубины тайн природы и ставит её могучие силы себе на службу. Развитие ядерной и термоядерной энергетики, реактивной и космической техники, высокоточного приборостроения, химии металлов и многих других отраслей промышленности невозможно без широкого применения интеллектуальных производственных систем, оснащенных высокопроизводительными автоматами.

Электроника — особенно цифровая — одна из самых молодых областей научного знания. В начале 1920-х годов Олег Лосев в ходе экспериментов с кристадином обнаружил, что на кристалле оксида цинка возможно получить усиление и генерацию сигнала. В 1947 году Уильямом Шокли, Джоном Бардином и Уолтером Браттейном был продемонстрирован первый действующий транзистор. В 1970 году группа учёных и инженеров под руководством Рэя Хольта разработала и испытала первый в мире процессор. Примеров подобных темпов развития в пятидесятилетний срок не знает ни одна другая отрасль.

Прогресс совершенствования электронных устройств впечатляет. В 1974 году десятиразрядный аналого-цифровой преобразователь для оцифровки видеосигнала представлял собой серийное изделие в стальном корпусе, которое потребляло десятки ватт электроэнергии, весило приблизительно 1,5 килограмма и стоило около четырёх месячных зарплат инженера. В настоящее время то же устройство — это изготавливаемая в массовых количествах микросхема весом в два-три грамма, с энергопотреблением порядка долей ватта и стоимостью, сравнимой со стоимостью двух средних обедов в столовой.

Отечественная электронная промышленность, преодолев разрушительные последствия распада СССР и используя

участие государственного капитала, смогла решить ряд серьёзных организационных и технических проблем, окрепнуть и заявить требования на долю мирового рынка электроники. Причём сказанное касается не только средств коммутации и области высоких частот, где позиции России, как одного из мировых лидеров, неоспоримы. В конце 2015 года компанией «МЦСТ» была выпущена опытная партия процессоров «Эльбрус-8С», изготовленных по технологии 28 нм.

Разумеется, научно-технический прогресс прямо и непосредственно влияет и на производственную среду. Развитие встроенных вычислительных систем шагнуло далеко вперёд. Повышаются требования к ответственности и умениям персонала, управляющего подчас в одиночку целыми производственными цехами, Теперь рабочим, инженерами технологам приходится иметь дело с приборами и механизмами, в которых воплощены самые последние достижения науки и техники.

В завершение хотелось бы вспомнить слова Героя Социалистического Труда, академика Академии Наук СССР Владимира Афанасьевича Обручева:

— Не отрекайтесь от мечты! Я разумею юношеские мечтания об открытиях, о творчестве. Есть люди, которые легко уступают обстоятельствам, сдаются после неудачного экзамена, при семейных или служебных затруднениях. Но затруднения проходят, а время упущено, и остаётся горькое сожаление о жизни, прожитой без огня, растраченной на мелочи, на труд, лишённый радости

Желаем вам широкой и светлой дороги, имя которой — Знание.

СПИСОК ЛИТЕРАТУРЫ

1) ***Spy-Bi-Wire*** [Электронный ресурс] : Материал из Википедии — свободной энциклопедии : Версия 614261979, сохранённая в 18:53 UTC 24 июня 2014 / Авторы Википедии // Википедия, свободная энциклопедия. — Электрон. дан. — Сан-Франциско: Фонд Викимедиа, 2015. — Режим доступа: https://en.wikipedia.org/w/index.php?title=Spy-Bi-Wire&oldid=614261979, свободный. — Загл. с экрана. — Яз. англ.

2) ***JTAG*** [Электронный ресурс] : Материал из Википедии — свободной энциклопедии : Версия 72526168, сохранённая в 12:31 UTC 3 августа 2015 / Авторы Википедии // Википедия, свободная энциклопедия. — Электрон. дан. — Сан-Франциско: Фонд Викимедиа, 2015. — Режим доступа: http://ru.wikipedia.org/?oldid=72526168, свободный. — Загл. с экрана. — Яз. рус., англ.

3) ***I²C*** [Электронный ресурс] : Материал из Википедии — свободной энциклопедии : Версия 75185392, сохранённая в 05:39 UTC 19 декабря 2015 / Авторы Википедии // Википедия, свободная энциклопедия. — Электрон. дан. — Сан-Франциско: Фонд Викимедиа, 2015. — Режим доступа: http://ru.wikipedia.org/?oldid=75185392, свободный. — Загл. с экрана. — Яз. рус., англ.

4) ***Serial Peripheral Interface*** [Электронный ресурс] : Материал из Википедии — свободной энциклопедии : Версия 75232126, сохранённая в 13:26 UTC 21 декабря 2015 / Авторы Википедии // Википедия, свободная энциклопедия. — Электрон. дан. — Сан-Франциско: Фонд Викимедиа, 2015. — Режим доступа: http://ru.wikipedia.org/?oldid=75232126, свободный. — Загл. с экрана. — Яз. рус., англ.

5) ***Универсальный асинхронный приёмопередатчик*** [Электронный ресурс] : Материал из Википедии — свободной энциклопедии : Версия 75133136, сохранённая в 08:29 UTC 16 декабря

2015 / Авторы Википедии // Википедия, свободная энциклопедия. — Электрон. дан. — Сан-Франциско: Фонд Викимедиа, 2015. — Режим доступа: http://ru.wikipedia.org/?oldid=75133136, свободный. — Загл. с экрана. — Яз. рус., англ.

6) ***Bootstrap loader*** [Электронный ресурс] — Режим доступа: http://www.computerhope.com/jargon/b/bootload.htm, свободный. — Загл. с экрана. — Яз. англ.

7) ***MSP430x2xx Family. User's Guide*** [Электронный ресурс] — Режим доступа: http://www.ti.com/lit/ug/slau144j/slau144j.pdf, свободный. — Загл. с экрана. — Яз. англ.

8) ***AVR. Учебный курс. Конечный автомат. Easy Electronics — электроника для всех*** [Электронный ресурс] — Режим доступа: http://easyelectronics.ru/avr-uchebnyj-kurs-konechnyj-avtomat.html, свободный. — Загл. с экрана. — Яз. рус.

9) ***MSP430G2x53, MSP430G2x13 Mixed Signal Microcontroller (Rev. J)*** [Электронный ресурс] — Режим доступа: http://www.ti.com/lit/ds/symlink/msp430g2553.pdf, свободный. — Загл. с экрана. — Яз. англ.

Миссия университета – генерация передовых знаний, внедрение инновационных разработок и подготовка элитных кадров, способных действовать в условиях быстро меняющегося мира и обеспечивать опережающее развитие науки, технологий и других областей для содействия решению актуальных задач.

КАФЕДРА ТЕХНОЛОГИИ ПРИБОРОСТРОЕНИЯ

Кафедра технологии приборостроения относится к числу ведущих кафедр института со дня его основания в 1931 году. Тогда она называлась кафедрой механической технологии и возглавлялась известным ученым в области разработки инструмента профессором А. П. Знаменским. Позже она была переименована в кафедру технологии приборостроения.

За время своего существования кафедра выпустила из стен института более двух тысяч квалифицированных инженеров, более сотни кандидатов и докторов наук. В разные годы ее возглавляли известные ученые и педагоги профессора Николай Павлович Соболев и Сергей Петрович Митрофанов.

Кафедра имеет выдающиеся научные достижения. Заслуженным деятелем науки и техники РСФСР, профессором С. П. Митрофановым были разработаны научные основы группового производства, за что он был удостоен Ленинской премии СССР. Методы группового производства с успехом применяются в промышленности и постоянно развиваются его учениками. Заслуженным изобретателем Российской Федерации Юрием Григорьевичем Шнейдером разработаны

метод и инструментарий нанесения регулярного микрорельефа на функциональной поверхности.

В настоящее время кафедра осуществляет выпуск бакалавров по направлениям подготовки 09.03.01 «Информатика и вычислительная техника» и 12.03.01 «Приборостроение»; магистров по направлениям подготовки 09.04.01 «Информатика и вычислительная техника» 12.04.01 «Приборостроение».

Афанасьев Максим Яковлевич
Федосов Юрий Валерьевич

Встроенные компьютерные системы

Методические рекомендации по выполнению лабораторных работ

В авторской редакции
Редакционно-издательский отдел Университета ИТМО
Зав. РИО Н. Ф. Гусарова
Подписано к печати
Заказ №
Тираж
Отпечатано на ризографе

Редакционно-издательский отдел
Университета ИТМО
197101, Санкт-Петербург, Кронверкский пр., 49