МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Нижегородский государственный университет им. Н.И. Лобачевского
Национальный исследовательский университет

ЗНАКОМСТВО С МИКРОКОНТРОЛЛЕРОМ СЕРИИ MSP430

Практикум

Рекомендовано методической комиссией радиофизического факультета для студентов ННГУ, обучающихся по направлениям подготовки 03.03.03 «Радиофизика», 02.03.02 «Фундаментальная информатика и информационные технологии» и специальности 10.05.02 «Информационная безопасность телекоммуникационных систем»

Нижний Новгород
2015

УДК 681.3

ББК 32.973.2

ЗНАКОМСТВО С МИКРОКОНТРОЛЛЕРОМ СЕРИИ MSP430 / Составители: Шкелёв Е.И., Калинин В.А., Пархачёв В.В. – Практикум. – Нижний Новгород: Нижегородский госуниверситет, 2015. – 27 с.

Рецензент: доцент кафедры общей физики, к.ф.-м.н. С.Н. Жуков

Методические указания содержат описание структуры, архитектуры, системы команд и средств программирования микроконтроллеров MSP430. Представлены принципиальная схема целевой микропроцессорной системы и дано описание лабораторной установки. Приведён краткий экскурс в среду разработки программного обеспечения IAR Embedded Workbench. Имеется перечень контрольных вопросов.

Методические указания к лабораторной работе разработаны для преподавания на радиофизическом факультете дисциплины «Аппаратные средства вычислительной техники» по направлению подготовки 02.03.02 «Фундаментальная информатика и информационные технологии», дисциплины «Микропроцессорные системы» по направлению подготовки 03.03.03 «Радиофизика» и дисциплины «Аппаратные средства телекоммуникационных систем» по специальности 10.05.02 «Информационная безопасность телекоммуникационных систем».

Ответственные за выпуск:
председатель методической комиссии радиофизического факультета ННГУ,
к.ф.-м.н., доцент **Н.Д. Миловский**,
зам. председателя методической комиссии радиофизического факультета
ННГУ, д.ф.-м.н., профессор **Е.З. Грибова**

УДК 681.3
ББК 32.973.2

1. Введение

Цель лабораторной работы состоит в изучении структуры, архитектуры и программирования микроконтроллеров (МК) фирмы Texas Instruments серии MSP430 на примере MSP430F2618, а также в освоении способов тестирования и средств отладки разрабатываемого программного обеспечения.

Возрастающая степень интеграции цифровых микросхем определила появление промышленных МК [1], реализованных на одном кристалле, в который кроме центрального процессорного элемента (ЦПЭ) интегрирован набор компонентов, состав и предназначение которых определяется конкретным приложением [2, 3]. К числу таких компонентов могут относиться устройства постоянной и перепрограммируемой памяти, последовательные и параллельные порты разного назначения, средства поддержки цифровых каналов связи, аналого-цифровые (АЦП) и цифро-аналоговые (ЦАП) преобразователи и др. Важной характеристикой МК является не только вычислительная производительность, но и возможность работы с низким энергопотреблением.

Лабораторная работа включает две основные части:

- в первой рассматривается структурное построение, принцип работы функциональных компонент, регистровая модель и набор команд МК MSP430F2618[1, 4-6];
- вторая знакомит со средствами программирования и размещения программного кода в памяти МК и со способом его отладки.

Программный код для МК разрабатывается на компьютере (не инструментальной ЭВМ) с операционной системой Windows. Для этого используется интегрированная среда разработки (IDE) Embedded Workbench компании IAR Systems, которая исполняется как Windows-приложение и имеет в своём составе кросс-ассемблер, позволяющий генерировать двоичный код программы для выбранного микроконтроллера. Двоичный код программы заносится в программируемую память (во *flash*-память) МК. Для этого используется стандартный последовательный интерфейс IEEE P1149 JTAG, разработанный «Объединённой рабочей группой по автоматизации тестирования (*Joint Test Automation Group*)». JTAG стандарт определяет метод поочередного сканирования и управления состояниями входов/выходов компонентов целевой микропроцессорной (МП) системы, в том числе входов/выходов программируемой памяти. Благодаря этому возможно программирование *flash*-памяти путём посылки через контакты JTAG порта битовой последовательности, составленной из адресных кодов и кодов данных.

Целевая МП система, основой которой является микроконтроллер серии MSP430, представлена в виде платы. Предназначением этой платы является освоение средств программирования МК, включая изучение среды разработки IDE и её взаимодействия с целевой МП системой. Программирование МК осуществляется через USB порт инструментальной ЭВМ (рис. 1), который соединён с целевой системой через USB-JTAG *интерфейс*. Такой интерфейс необхо-

дим для того, чтобы выводимые через USB порт сигналы воспринимались JTAG портом целевой системы, а сигналы идущие от целевой системы воспринимались USB портом компьютера.

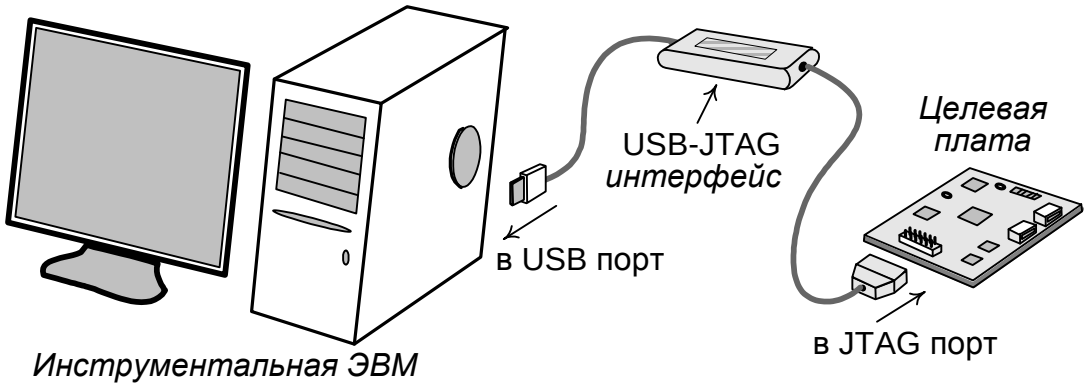

Рис. 1. Схема программирования целевой МП системы

После программирования МК способен работать в автономном режиме, исполняя загруженную в его память программу. При этом питание платы осуществляется от отдельного источника напряжения 3.3 В, который подключается через соответствующий разъём на плате. Микроконтроллер сконструирован с расчётом на низкое энергопотребление, и это позволяет питать его от батарейного источника – от аккумулятора. Предусмотрены средства управления энергопотреблением, основой которых является управление тактовой частотой ядра, а также управление синхронизирующими импульсными последовательностями для устройств ввода/вывода (УВВ).

2. Описание микроконтроллера

Микроконтроллеры семейства MSP430 фирмы Texas Instrumenrs построены как машина фон Неймана (см. рис. 2, на котором приведена структурная схема МК MSP430F2618) с общей внутри кристалла системной магистралью, образованной 16-разрядной шиной адреса ША (MAB – *Memory Address Bus*), 16-разрядной шиной данных ШД (MDB – *Memory Data Bus*) и шиной управления ШУ (MCB – *Memory Control Bus*). Адреса портов ввода/вывода отображены на память и занимают выделенное для них 4-битовое адресное пространство. Часть шины данных имеет пониженную 8-битовую разрядность, соединенную с основной её частью с помощью моста 16/8.

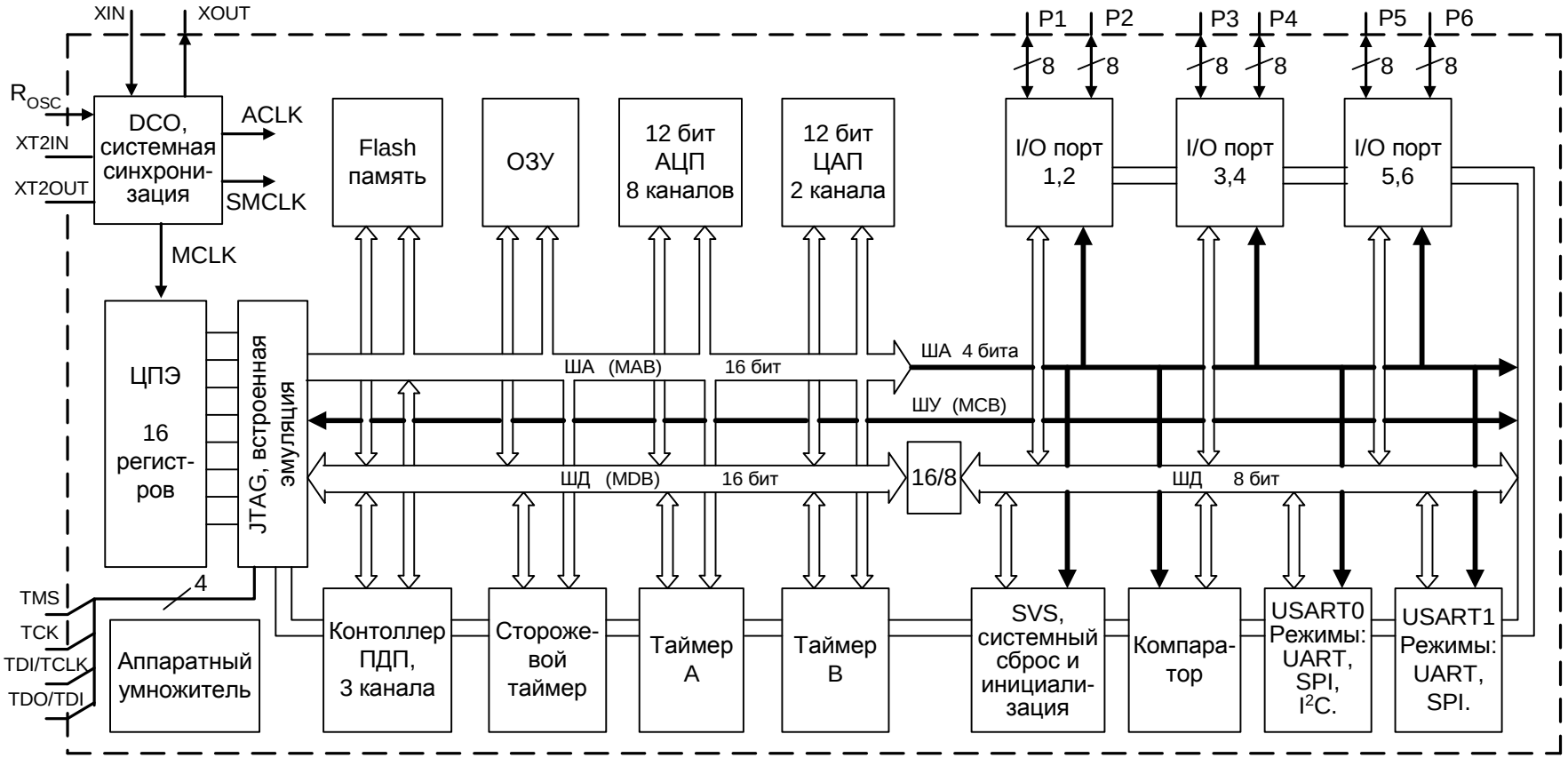

Рис. 2. Структура микроконтроллера MSP430F2618

В состав микроконтроллера входят:

1) *центральный процессорный элемент* ЦПЭ, имеющей 16 16-разрядных регистров, включая программный счётчик (PC), указатель стека (SP), регистр состояния (SR) и генератор констант (CG). Генератор констант служит для получения наиболее часто используемых констант (-1, 0, 1, 2, 4, 8) и позволяет, в дополнение к 27 основным командам, эмулировать еще 24 дополнительные инструкции. Последние не относятся к исполняемым ядром МК командам и введены лишь для облегчения программирования. Компилятор автоматически в процессе трансляции исходного ассемблерного кода заменяет эмулирующие инструкции на эквивалентные им реально исполняемые инструкции;

2) *встроенная flash-память* на 48 Кбайта; используется для хранения программного кода и кода данных. Содержимое памяти *flash*-памяти определяется переданной через JTAG порт информацией. При исполнении программы обращение к *flash*-памяти возможно только в режиме чтения. Начальный адрес *flash*-памяти зависит от её объема. Конечный адрес всегда равен 0FFFFh;

3) *оперативное запоминающее устройство* (ОЗУ) объёмом 10 Кбайт.

На рис. 3 представлена карта памяти МК семейства MSP430. В верхней области адресного пространства (в области старших адресов 0FFE0h – 0FFFFh) расположена таблица векторов прерывания. Область 0h – 01FFh отведена под отображённые на память регистры процессора. Адреса с 0100 до 01FFh предназначены для 16-разрядных, а адреса с 010h по 0FFh – для 8-разрядных УВВ. Далее идёт адресное пространство ОЗУ, а ниже таблицы векторов прерываний расположена область адресов *flash*-памяти. Нижняя граница адресного пространства *flash*-памяти и верхняя граница адресов ОЗУ зависят от объёма интегрированной в кристалл МК памяти, и эти границы определяются конкретной модификацией микроконтроллера.

Обмен данными с 16-разрядными УВВ происходит по 16-разрядной шине данных. Для доступа к 8-разрядным портам ввода/вывода используется 8-разрядная шина данных.

К интегрированным в кристалл МК устройствам ввода/вывода относятся:

1) 12-*разрядный аналогово-цифровой преобразователь* с частотой дискретизации 200 кГц. Имеет 8 внешних входов для аналоговых сигналов, буфер на 16 слов для хранения результатов преобразования, организованный как очередь с автоматическим заполнением и сохранением её содержимого в основной памяти. Источник опорного напряжения встроен в схему АЦП. Кроме того, в него интегрированы датчик температуры и цепь обнаружения разряда источника питания, позволяющая контролировать состояние источника (батареи) при автономной работе;

Рис. 3. Карта памяти микроконтроллеров семейства MSP430

2) *двухканальный 12-разрядный цифро-аналоговый преобразователь* (ЦАП);

3) *аналоговый компаратор* – без гистерезиса со встроенным программируемым делителем для установки напряжения компарирования (опорного напряжения). Компаратор имеет несколько входов, переключаемых с помощью мультиплексора, и программно отключаемый RC-фильтр на выходе;

4) *контроллер прямого доступа к памяти* **(**КПДП**)**, имеющий три канала, каждый из которых может использоваться для перемещения данных из устройства (или в устройство) ввода/вывода в оперативную (или из оперативной) память(и) независимо от ЦПЭ;

5) *супервизор напряжения источника питания с блоком системного сброса и инициализации* контролирует состояние источника питания и генерирует сигнал системного сброса при снижении напряжения питания ниже установленного порогового значения. Используется также при включении МК в работу, т.е. при подаче на него питания. После установки напряжения питания до

рабочего уровня генерирует сигнал запроса прерывания, запускающего процедуру инициализации;

6) *два универсальных* 16-*битовых таймера* (Timer A и Timer B) с режимами «захват/сравнение» и возможностью формирования сигнала широтно-импульсной модуляции (ШИМ). Оба таймера выполнены в виде 16-разрядных двоичных счётчиков, подсчитывающих импульсы, получаемые либо от устройства синхронизации, либо от выделенных для этого выводов микросхемы МК. Счётчики таймеры способны работать в нескольких режимах. При этом разрядность счётчика таймера B может программно установиться на 8, 10, 12 или 16. Для фиксации (захвата) данных о результате счёта и для генерации временных интервалов имеются блоки захвата/сравнения. Таймер A имеет три таких блока, а в таймере B их число может быть увеличено (в зависимости от модификации) до семи. Любой из таймеров может выполнять множественный одновременный счёт, множественные операции захвата/сравнения, формирование выходных сигналов специальной формы, в частности, сигналы с широтно-импульсной (ШИМ) модуляцией, а также комбинации названных операций;

7) *сторожевой таймер*. Главная функция сторожевого таймера – контроль системного рестарта при сбое в работе программы: если программа не производит обращение к сторожевому таймеру в течение заданного интервала времени, то производится системный сброс. В тех случаях, когда данная функция таймера не требуется, он может быть запрограммирован на работу в качестве интервального таймера;

8) *два универсальных синхронно/асинхронных приёмо-передатчика* (USART0,1). USART0 работает в трёх режимах: в режиме универсального асинхронного приёмо-передатчика (UART режим), в режиме синхронного периферийного интерфейса (SPI) и в режиме I^2C (в режиме двухпроводного интерфейса). В USART1 такой режим работы не предусмотрен.

При обмене данными в асинхронном режиме UART каждый передаваемый символ содержит стартовый бит, семь или восемь битов данных, бит контроля чётности и один или два стоповых бит. Период следования битов определяется выбранным источником тактовых импульсов и настройкой работы сдвиговых регистров, преобразующих поступающий по шине данных MDB параллельный код в последовательный, а принимаемый последовательный код – в параллельный для передачи его на шину MDB. Скорость передачи определяется частотой тактирования принимающего и предающего сдвиговых регистров. В асинхронном режиме обмена данными между двумя устройствами используется протокол с форматом «свободная линия». При таком формате блоки данных на линиях передачи или приёма разделены временем простоя (временем, когда линия свободна). Простой линии приёма обнаруживается, когда приняты 10 или более не изменяющихся со временем логических единиц (меток) после первого стопового бита символа. При использовании двух стоповых битов, второй стоповый бит принимается за первый маркерный бит периода простоя. Когда связываются три или более устройств, к биту данных добавляется адресный

бит, что позволяет USART поддерживать многопроцессорные коммуникационные форматы со «свободной линией».

В синхронном режиме (в режиме SPI) последовательные данные передаются и принимаются множеством устройств с использованием общего тактирования, обеспечиваемого ведущим. Ведущий определяется по значению дополнительного сигнала (сигнала STE – разрешение передачи ведомого), управляемого ведущим. Сигнал STE необходим для разрешения приёма и передачи данных устройством, на выводе которого этот сигнал установлен.

Режим I^2C обычно используется для внутрисистемного обмена между интегральными схемами встроенных МП систем и поддерживает любые ведущие или ведомые устройства, совместимые с интерфейсом I^2C через последовательную двухпроводную шину I^2C. Внешние компоненты, присоединенные к шине I^2C последовательно передают и/или принимают последовательные данные в/из USART;

9) *шесть портов общего назначения* **(**P1–P6**)** являются многофункциональными портами ввода/вывода, которые могут быть использованы как 8-разрядные параллельные цифровые порты, так и как средство для подключения к контактам микроконтроллера таймеров, аналоговых входов АЦП, выходов ЦАП, входов и выходов компаратора и последовательных портов, а также для выдачи и приёма сигналов синхронизации. Для реализации этих возможностей предусмотрены средства конфигурации портов посредством занесения в принадлежащие им регистры управления соответствующей информации.

Система тактирования микроконтроллера разработана специально для использования в приложениях с питанием от батарей. Основное тактирование (MCLK – *Master Clock*), необходимое для работы ЦПЭ, имеет три источника:

- внутренний низкочастотный/высокочастотный генератор последовательности тактовых импульсов LFXT1CLK с внешним резонатором – с низкочастотным часовым кристаллом на 32768 Гц или с одним из стандартных резонаторов (кристаллов) на диапазон от 450 кГц до 8 МГц;
- один из источников дополнительных тактовых импульсов XT2CLK: либо высокочастотный внутренний генератор со стандартными внешними резонаторами, либо внешний генератор на диапазон от 450 кГц до 8 МГц;
- встроенный тактовых импульсов DCOCLK с цифровым управлением (DCO – *Digitally Controlled Oscillator*), состоящий из задающего управляемого RC-генератора и дополнительной схемы формирования тактирующей последовательности DCOCLK.

Наряду с основным имеет место вспомогательное (ACLK – *Auxiliary Clock*) и второстепенное (SMCLK – *Sub-System Master Clock*) тактирование. Модуль ACLK предназначен для высокоскоростных устройств ввода/вывода и является буферизированным источником тактовых импульсов LFXT1CLK с делителем на 1, 2, 4 или 8. Период следования ACLK программно задаётся для конкретных периферийных модулей. Источником второстепенного SMCLK

тактирования в зависимости от программных установок является либо XT2CLK (при условии доступности), либо DCOCLK с коэффициентом деления 1, 2, 4 или 8. Параметры SMCLK также задаются программно для конкретных УВВ.

Архитектура MSP430 позволяют эффективно использовать 16-разрядный ЦПЭ только в нужные промежутки времени, что позволяет на низких тактовых частотах снизить энергопотребление, сводя его до ультранизкого, а на высоких при активизации основного высокоскоростного модуля тактирования – выполнять быструю обработку сигналов. Основной модуль тактирования может быть программно сконфигурирован

- на работу с номинальной частотой без использования внешних компонент кроме одного внешнего резистора и
- на работу с одним или двумя внешними кристаллами (резонаторами) XT1 и XT2. Внешние компоненты подключаются к соответствующим контактам Rosc, XIN, XOUT, XT2IN и XT2OUT.

3. Система прерываний и рабочие режимы

Механизм прерываний обеспечивает реакцию ЦПЭ на происходящие в МП системе события, связанные с работой её функциональных узлов. Получив запрос прерывания процессор приостанавливает выполнение текущей программы и переходит на исполнение процедуры обслуживания поступившего запроса. Прерывания делятся на три типа:

- системное (системный сброс);
- немаскируемое (NMI);
- маскируемые.

Запросы прерывания со стороны устройств ввода/вывода относятся к классу маскируемых и возникают в основном при необходимости принять или передать данные в УВВ. **Маскируемые прерывания** можно запретить или разрешить путём установки или сброса бита общего разрешения прерываний GIE в регистре состояния SR, а также индивидуальных бит разрешения в *регистре* (*или регистрах*) *управления* работой конкретного УВВ. Поступившие от УВВ запросы прерывания как *флаги прерываний* фиксируются в соответствующих *регистрах флагов прерывания*.

Сигнал запроса прерывания (сигнал IRQ – *Interrupt Request*) генерирует каждое из незамаскированных и готовых к обмену данными устройство. Сигналы IRQ направлены на то, чтобы ЦПЭ прерывал исполнения текущего потока команд и перешёл на процедуру ISR (*Interrupt Service Routine*) обработки запроса прерывания.

Запросы от отдельных УВВ объединяются по «ИЛИ» (рис. 4). В результате формируется обращённый к ЦПЭ запрос INTR (*Interrupt Request*), который воспринимается, если установлен бит общего разрешения прерываний GIE (элемент «И» на рис. 4). В ответ на запрос INTR ЦПЭ посылает сигнал под-

тверждения INTA (*Interrupt Acknowledge*). Подтверждение INTA распространяется последовательно от одного УВВ к другому. При этом первое получившее подтверждение INTA устройство прекращает дальнейшее распространение этого игнала. Тем самым реализуется схема расстановки приоритетов, согласно которой наивысшим приоритетом обладает наиболее близкое к ЦПЭ УВВ.

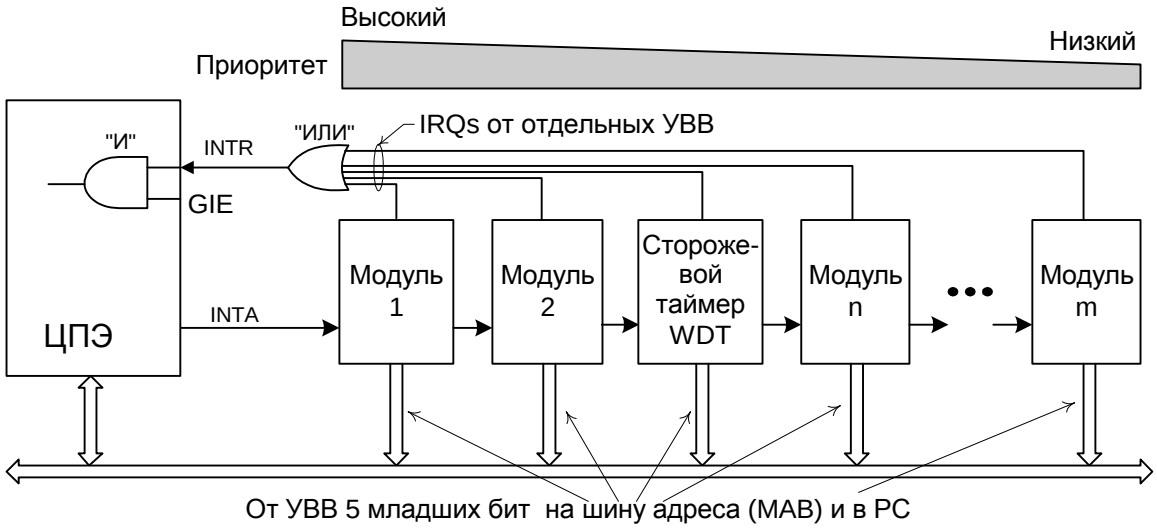

Рис. 4. Логика приоритетов УВВ

Получив подтверждение INTA, УВВ выставляет на шину адреса вектор прерывания, значение которого берётся из *регистра вектора прерывания*. По вектору прерывания определяется стартовый адрес процедуры обработки прерывания ISR.

В целом последовательность действий при обработка запроса прерывания выглядит следующим образом:

1) любая текущая команда выполняется до конца;
2) содержимое программного счетчика PC помещается в стек;
3) в стек помещается содержимое регистра состояния SR;
4) если имело место несколько запросов прерываний, то сначала обрабатывается запрос с наивысшим приоритетом, а остальные ожидают обслуживания. Наивысшим приоритетом обладает УВВ, которое наиболее близко расположено по отношению к ЦПЭ;
5) автоматически сбрасывается флаг прерывания а регистре флагов источника запроса. Флаги остальных запросов остаются установленными в ожидании обслуживания;
6) регистр состояния SR очищается. Неизменным остаётся только бит SCG0 (о назначении разрядов регистра SR см. ниже). В результате прекращается работа в любом режиме пониженного энергопотребления. Это необходимо для, чтобы ЦПЭ смог быстро исполнить соответствующую ISR;
7) из регистра вектора прерывания обрабатываемого УВВ берётся значение вектора прерывания. По нему определяется значение пяти младших бит программного счётчика PC и стартовый адрес ISR. При этом старшие раз-

ряды PC устанавливаются в «1». Последующее инкрементирование PC обеспечивает исполнение процедуры обработки прерывания.

Каждый вектор прерывания ассоциирован с соответствующим элементом *таблицы векторов прерываний* (таблица 1), которая расположена в памяти МК по базовому адресу 0FFE0h. В таблицу занесены стартовые адреса ISR тех УВВ, для которых предусмотрена возможность работы в режиме с прерываниями, а также точка входа в программу начальной установки МК (RESET_VECTOR).

Значение вектора прерывания равно смещению относительно базового адреса 0FFE0h и для его кодирования достаточно 5-ти разрядов. Поэтому векторы прерываний передаются в ЦПЭ по пяти младшим линиям шины адреса MAB.

Программа обработки прерывания заканчивается командой возврата из прерывания RETI (*Return from Interrupt*). По этой команде

1) восстанавливается из стека содержимое регистра SR. В результате становятся действующими все предыдущие установки разрядов в регистре SR и восстанавливается предшествующий режим работы МК,
2) восстанавливается из стека содержимое программного счетчика PC и продолжается выполнение прерванной программы.

Возможны вложенные прерывания, если бит GIE будет установлен во время выполнения текущей процедуры обработки прерывания. Когда вложенные прерывания разрешены, любой запрос прерывания с приоритетом выше текущего вызовет переход на выполнение соответствующей новому запросу ISR.

Немаскируемые (NMI) прерывания – это прерывания от источников, состояние которых имеет критическое значение для работы МП системы. На них не действует общий бит разрешения прерываний GIE и их можно запретить только установкой индивидуального бита разрешения. Немаскируемые прерывания обслуживаются по выделенному для них вектору NMI_VECTOR.

Таблица 1

Вектор прерывания	**Адрес**	**Смещение относительно 0FFE0h**	**Источник запроса прерывания**
DACDMA_VECTOR	0FFE0h	0	ЦАП/КПДП (при использовании режима ПДП)
PORT2_VECTOR	0FFE2h	2	Порт P2
USART1TX_VECTOR	0FFE4h	4	Порт USART1 (режим передачи)
USART0RX_VECTOR	0FFE6h	6	Порт USART1 (режим приёма)
PORT1_VECTOR	0FFE8h	8	Порт P1
TIMERA1_VECTOR	0FFEAh	10	Таймер A
TIMERA0_VECTOR	0FFECh	12	Таймер A

ADC12_VECTOR	0FFEEh	14	АЦП
USART0TX_VECTOR	0FFF0h	16	Порт USART0 (режим передачи)
USART0RX_VECTOR	0FFF2h	18	Порт USART0 (режим приёма)
WDT_VECTOR	0FFF4h	20	Сторожевой таймер
CMPRA_VECTOR	0FFF6h	22	Компаратор A
TIMERB1_VECTOR	0FFF8h	24	Таймер B
TIMERB0_VECTOR	0FFFAh	26	Таймер B
NMI_VECTOR	0FFFCh	28	Немаскируемое прерывание
RESET_VECTOR	0FFFEh	30	Сброс (наибольший приоритет)

Процессоры семейства MSP430 разработаны для приложений с ультранизким потреблением энергии и способны работать в разных режимах. Смена режимов работы обусловлена:

- стремлением снизить уровень потребляемой ядром микроконтроллера энергии,
- обеспечением необходимой скорости обработки данных и пропускной способности устройств ввода/вывода и
- минимизацией потребления тока устройствами ввода/вывода.

Энергосберегающие режимы (LPMs – *Low Power Modes*) перечислены в таблице 2. Уровень энергопотребления определяется битами CPUOFF, OSCOFF, SCG0 и SCG1 в регистре состояния SR (рис. 5).

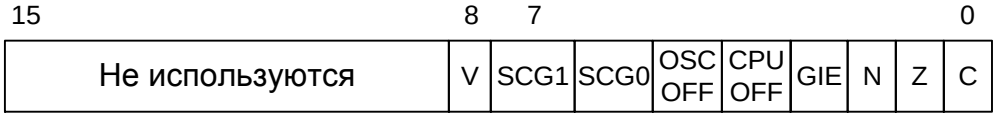

Рис. 5. Формат регистра состояния SR

Включение битов режима CPUOFF, OSCOFF, SCG0 и SCG1 в регистр SR обусловлено тем, что при переходе на процедуру обработки прерывания текущий режим работы может быть восстановлен путём сохранения содержимого SR в стеке. Такое восстановление происходит всегда при возврате из процедуры обработки прерывания.

Помимо бит режима в регистр SR включены флаги (признаки) операций арифметическо-логического устройства АЛУ (о флагах АЛУ см. раздел 3). По флагам операций (по переполнению V, по переносу C, по знаку S, по отрицательному результату N) определяется корректность работы АЛУ, а также реализуются команды условных переходов.

Таблица 2

SCG1	SCG0	OSCOFF	CPUOFF	Режим	Состояние ЦПУ и систем тактирования
0	0	0	0	Активный	ЦПЭ и все системы тактирования активны
0	0	0	1	LPM0	MCLK отключён и ЦПЭ не тактируется (отключён); SMCLK и ACLK активны
0	1	0	1	LPM1	ЦПЭ, MCLK и DCO отключены; DCO не используется для MCLK или SMCLK тактирования; SMCLK и ACLK активны
1	0	0	1	LPM2	ЦПЭ, MCLK, SMCLK и DCO отключены; задающий RC-генератор остается включённым; ACLK активно
1	1	0	1	LPM3	ЦПЭ, MCLK, SMCLK и DCO отключены; RC-генератор отключен; ACLK активно
1	1	1	1	LPM4	ЦПЭ и все системы тактирования отключены

4. Центральный процессорный элемент

Центральный процессорный элемент состоит из 16-битового арифметическо-логического устройства (АЛУ) и набора из 16-ти регистров – регистры R0–R15 (рис. 6). Среди них 12 регистров R4–R15 – это регистры общего назначения, а четыре регистра R0–R3 выполняют соответственно функции счётчика команд PC, указателя вершины стека SP, регистра статуса SR и генератора констант CG. Несмотря на специализацию некоторых регистров (это относится к регистрам R0–R3), по отношению к АЛУ все регистры равноправны.

16-разрядный программный счётчик PC/R0 указывает на следующую команду, которая будет выполняться после исполнения текущей команды. PC инкрементируется соответственно тому, из какого чётного числа байтов (два, четыре или шесть) состоит каждая команда. Команды располагаются в адресном пространстве 64 Кбайт и выравниваются по границам слов, поэтому значения PC всегда соответствуют чётным адресам.

Указатель стека SP/R1 используется ЦПЭ для хранения адресов возврата из подпрограмм и процедур обработки прерываний. Стек может быть построен по предекрементной или постинкрементной схеме. Указатель стека SP может использоваться со всеми командами и во всех режимах адресации. Указатель стека SP инициализируется пользователем и выравнивается по чётным адресам.

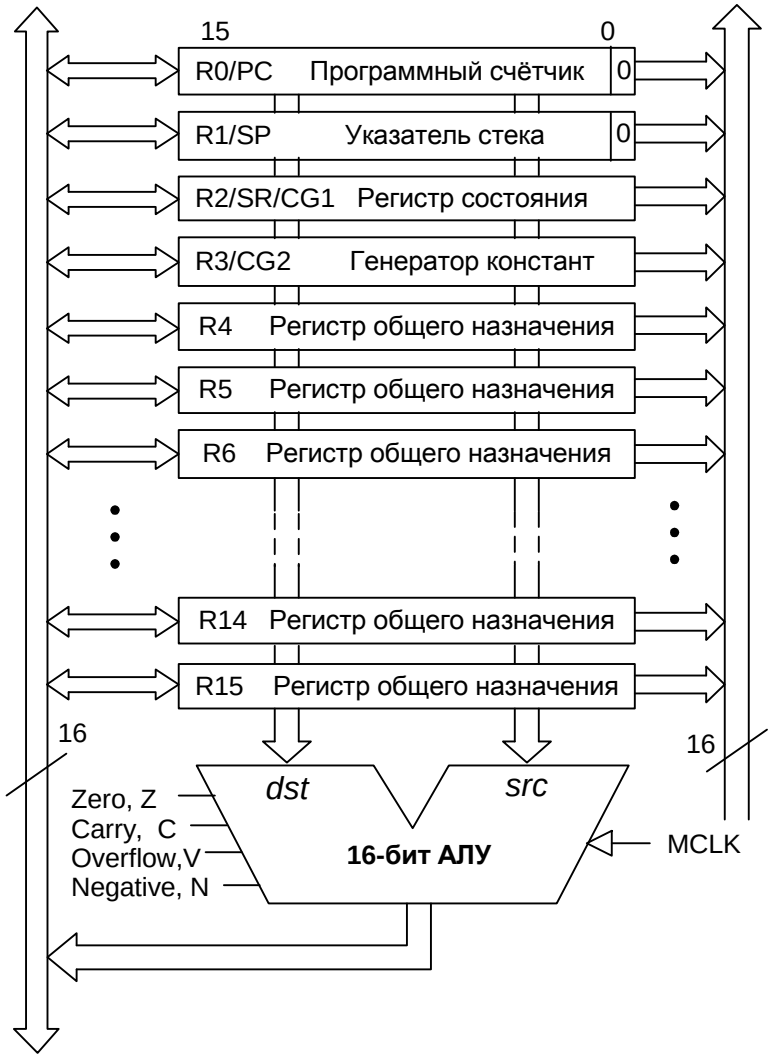

Рис. 5. Блок-схема ЦПЭ

В регистр состояния SR/R2 включены следующие флаги (признаки) операций АЛУ: флаги нуля Z (zero), переноса C (*carry*), переполнения V (*overflow*) и отрицательного результата N (*negative*). R2 используется как регистр источник или как регистр приёмник и при необходимости может быть использован для поддержки работы генератора констант.

Работу генератора констант CG поддерживают регистры R2 и R3. С их помощью генерируются шесть констант (-1, 0, 1, 2, 4, 8), используемых при обращении к памяти в режимах с косвенной регистровой и косвенной регистровой с атоинкрементацией или автодекрементацией адресации. Способы использования регистров R2 и R3 и значений генерируемых ими констант зависят от режима адресации, определяемого соответствующими битами в кодах операций. Применение генератора констант даёт следующие преимущества:

- не требуется обращение к памяти для получения констант -1, 0, 1, 2, 4, 8;
- сокращается формат команд обращения к памяти, поскольку не требуется включения в них дополнительного слова, если значение этого слова совпадает с одной из шести автоматически генерируемых констант;

– если одна из шести констант рассматривается как непосредственный операнд, то Ассемблер автоматически генерирует команды, рассчитанные на использование генератора констант. При использовании регистров R2 и R3 в режиме генерации констант, их адресация не может быть явной – они действуют по умолчанию и только как регистры-источники.

Двенадцать регистров общего назначения R4-R15 могут быть использованы в качестве регистров данных, указателей адресов или индексных значений и доступны при выполнении команд работы с байтами или словами.

Набор команд

При написании программ необходимо учитывать особенности ЦПЭ, его архитектуру (регистровую модель) и набор команд, с помощью которых реализуются арифметические и логические операции, условные и безусловные переходы, вызовы и возвраты из процедур, сохранение содержимого регистров в памяти и загрузка данных из памяти в регистры, обмен данными с портами (регистрами) ввода/вывода. Команды в памяти представляются в виде двоичного кода, содержащего код операции (КОП) и указание на то, где расположены операнды, или сами операнды. Программирование в двоичных кодах весьма трудоёмко и для его облегчения двоичным кодам команд сопоставляются мнемонические обозначения, которые являются основой низкоуровневого языка программирования – языка Ассемблера.

Полный набор команд МК семейства MSP430 включает 27 команд ядра и 24 эмулированные команды. Команды ядра – это команды, имеющие уникальный код операции декодируемый ЦПЭ. Эмулированные команды представляют собой инструкции, облегчающие чтение и написание кода, но не имеющие собственного кода операции, поэтому Ассемблер автоматически меняет их на эквивалентные команды ядра. Использование эмулирующих команд не приводит к увеличению объёма кода или снижению производительности. Существует три формата команд ядра:

– с двойным операндом – с операндом-источником *src* и операндом-приемником *dst* (таблица 3);
– с одиночным операндом – с одним операндом-источником *src* или с одним операндом-приемником *dst* (таблица 4);
– команды перехода, вызова, возврата (таблица 5).

В начале каждой команды указано её мнемоническое обозначение. Затем указываются операнды (один или два). Команды с одним и двумя операндами могут быть предназначены для работы с байтами или со словами, на что показывает соответствующее расширение мнемоник КОП «.B» или «.W». Если расширение в КОП не показано, то по умолчанию такие команды предназначаются для работы со словами. Операндами может быть содержимое регистров или содержимое ячеек памяти. Но в командах с двумя операндами одним из операндов является регистр.

Команды с двойным операндом

Таблица 3

Команда	**Действие команды**	**Биты статуса**			
		V	N	Z	C
MOV(.B) *src*, *dst*	*src* → *dst*	-	-	-	-
ADD(.B) *src*, *dst*	*src* + *dst* → *dst*	*	*	*	*
ADDC(.B) *src*, *dst*	*src* + *dst* + C→ *dst*	*	*	*	*
SUB(.B) *src*, *dst*	*src* + *.not. dst* + 1→ *dst*	*	*	*	*
SUBC(.B) *src*, *dst*	*src* + *.not. dst* + C→ *dst*	*	*	*	*
CMP(.B) *src*, *dst*	*dst* – *src*	*	*	*	*
DADD(.B) *src*, *dst*	*src* + *dst* + C→ *dst* (десятичное)	*	*	*	*
BIT(.B) *src*, *dst*	*src .and. dst*	0	*	*	*
BIC(.B) *src*, *dst*	*.not.src .and. dst* → *dst*	-	-	-	-
BIS(.B) *src*, *dst*	*src .or. dst* → *dst*	-	-	-	-
XOR(.B) *src*, *dst*	*src .xor. dst* → *dst*	*	*	*	*
AND(.B) *src*, *dst*	*src .and. dst* → *dst*	0	*	*	*

«*» – влияет на бит статуса; «-» – не влияет на бит статуса;
«0» – бит статуса очищается;
в круглых скобках необязательное расширение мнемоник Ассемблера.

Команды с одним операндом

Таблица 4

Команда	**Действие команды**	**Биты статуса**			
		V	N	Z	C
RLA(.B) *dst*	Арифметический сдвиг вправо на 1 бит	0	*	*	*
RLC(.B) *dst*	Циклический сдвиг вправо на 1 бит через триггер переноса C	*	*	*	*
RRA(.B) *dst*	Арифметический сдвиг вправо на 1 бит	0	*	*	*
RRC(.B) *dst*	Циклический сдвиг вправо на 1 бит через триггер переноса C	*	*	*	*
PUSH(.B) *dst*	SP-2 → SP, *src* → @SP. Проталкивание в стек	-	-	-	-
SWPB *dst*	Переставить байты местами	-	-	-	-
CALL *dst*	SP-2 → SP, PC+2 → @SP, *dst* → PC. Вызов процедуры	-	-	-	-
RET	@SP → PC SP+2 → SP. Возврат из процедуры	*	*	*	*
RETI	TOS → SR, SP+2 → SP; TOS → PC,SP+2 → SP. Возврат из прерывания (TOS – вершина стека)	*	*	*	*
SXT	Bit 7 → Bit 8.....Bit 15. Расширение знака байта на слово	0	*	*	*

«*» – влияет на бит статуса; «-» – не влияет на бит статуса;
«0» – бит статуса очищается;

Команды перехода

Таблица 5

Команда	Действие команды
JEQ/JZ *метка*	Переход к метке, если бит Z нуля установлен
JNE/JNZ *метка*	Переход к метке, если бит Z нуля сброшен
JC *метка*	Переход к метке, если бит C переноса установлен
JNC *метка*	Переход к метке, если бит переноса C сброшен
JN *метка*	Переход к метке, если бит N отрицательного результата установлен
JGE *метка*	Переход к метке, если (N.XOR.V)=0
JL *метка*	Переход к метке, если (N.XOR.V)=0
JMP *метка*	Безусловный переход

Переходы делятся на условные и безусловные. Условные переходы обеспечивают ветвление программы относительно программного счётчика PC и не оказывают влияния на биты статуса. Возможный диапазон переходов с помощью условных команд составляет от –511 до +512 слов относительно текущего значения PC. Указанное в команде 10-разрядное смещение обрабатывается как число со знаком. При этом его значение удваивается и только после этого складывается со значением программного счётчика. Это обусловлено тем, что адреса всех команд выравниваются по границам 16-разрядных слов.

Режимы адресации

При работе с данными и при реализации переходов используются несколько способов вычисления исполнительного адреса операндов *src* и *dst*, называемых режимами адресации. Режимы адресации микроконтроллеров семейства MSP430 представлены в таблице 6.

Таблица 6

Режим адресации	Синтаксис	Описание
Регистровый	Rn	Содержимое регистра Rn является операндом
Индексный	X(Rn)	Значение (Rn+X) указывает на операнд – ячейку памяти. Значение X находится в следующем за кодом команды слове
Символьный	ADDR	Адрес ADDR определяется по смещению X относительно PC. Другими словами, на операнд указывает значение (PC+X). Значение X сохранено в следующем слове. Реализован как индексный режим X(PC)
Абсолютный	&ADDR	Слово, следующее за командой с адресом (PC + X=0), содержит адрес операнда. Реализован как индексный режим X(R2) с использование генератора констант – константы 0, извлекаемой из регистра R2

Косвенный регистровый	@Rn	Содержимое регистра Rn используется как указатель на операнд
Косвенный с автоинкрементацией	@Rn+	Содержимое Rn используется как указатель на операнд. Содержимое Rn впоследствии увеличивается на 1 для байтовых команд и на 2 для команд-слов
Прямой (непосредственный)	#N	Слово, следующее за командой, содержит непосредственную константу N. Реализован как косвенный автоинкрементный режим @PC+

5. Умножитель-аккумулятор (MAC)

К вычислительным устройствам микроконтроллера относится 16-*разрядный аппаратный умножитель*. Аппаратный умножитель не входит в состав ЦПЭ и работает независимо от него. Поддерживаются четыре типа операций: знаковое и беззнаковое умножение (MPYS и MPY), знаковое и беззнаковое умножение с накоплением (MACS и MAC).

Тип операции выбирается адресом, в который записан первый операнд. Умножитель имеет два 16-разрядных регистра операндов OP1 и OP2 и три регистра результата RESLO, RESHI и SUMEXT (рис. 6) и приступает к работе сразу после загрузки регистров операндов. Поэтому для умножителя не требуется специальных инструкций и дополнительных циклов синхронизации. Результат автоматически сохраняется в регистрах результата. Операции выполняются при любых комбинациях 8- и 16-разрядных операндов.

В регистре RESLO содержится младшее слово результата, в RESHI – старшее слово результата, а в регистре SUMEXT находится информация о результате. Для появления результата необходимо 3 такта MCLK. Результат может быть прочитан следующей командой после записи в OP2. Исключение составляет случай, когда используется косвенный режим адресации к регистру результата. В этом случае необходимо вставить команду NOP перед чтением результата.

Регистр первого операнда OP1 имеет четыре адреса 130h (MPY), 132h (MPYS), 134h (MAC) и 136h (MACS), используемые при выборе режима умножения. Запись первого операнда по желаемому адресу позволяет выбрать тип операции умножения, но не приведет к началу выполнения какой-либо операции. Запись операнда в регистр второго операнда OP2 инициирует операцию умножения. Запись в OP2 стартует выбранную операцию над значениями, сохраненными в OP1 и OP2. Результат записывается в три регистра результата RESLO, RESHI и SUMEXT. Повторение операций умножения может выполняться без перезагрузки OP1, если значение в OP1 используется для последовательных операций и если нет необходимости перезаписывать значение в OP1 для выполнения операций.

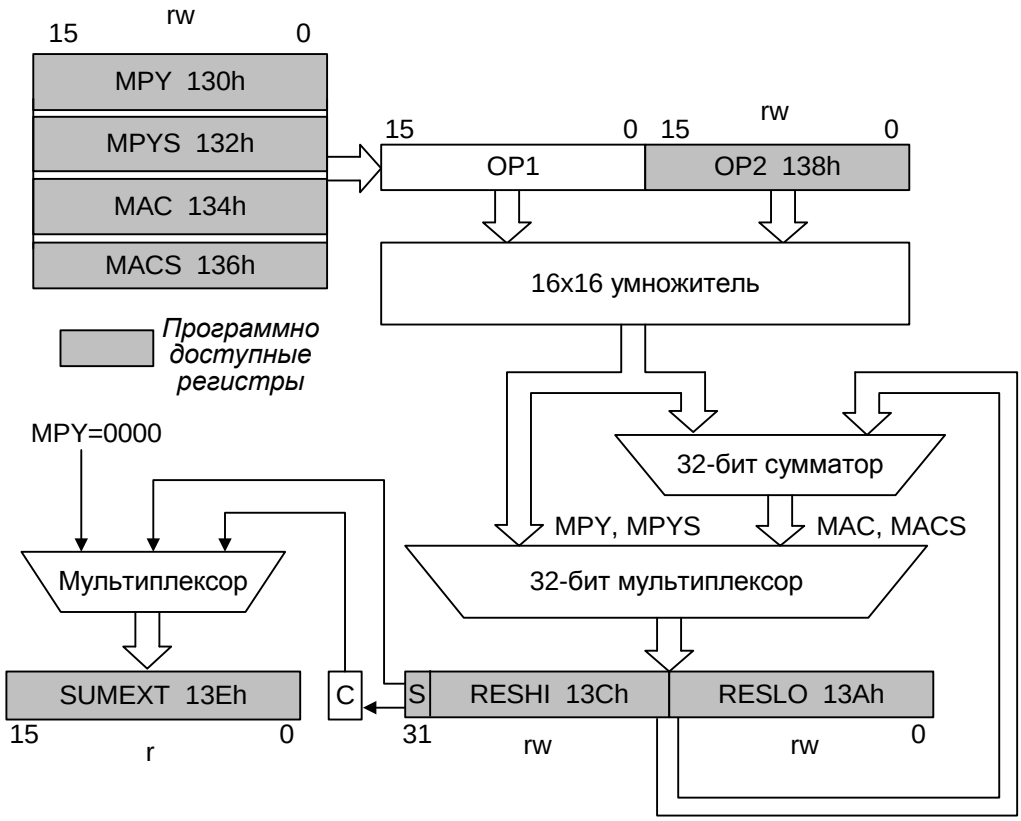

Рис. 6. Структурная схема умножителя-аккумулятора

Результат размещается в двух регистрах – в младшем RESLO и старшем RESHI 16-разрядных регистрах результата. Содержимое старшего регистра результата RESHI зависит от операции умножения.

6. Тестирование системы

Для тестирования системы используется последовательный интерфейс, выполненный в стандарте IEEE P1149 JTAG Этот стандарт, как уже говорилось, определяет метод поочередного сканирования состояний входов/выходов компонентов системы и используется внутрисхемным эмулятором для доступа к встроенной системе поддержки тестирования. Эмуляторы используют порт JTAG для текущего контроля и управления процессором, установленным на целевой печатной плате, при отладке системы и её программного обеспечения. Для отладки используется также программный симулятор инструментальной ЭВМ.

Сканирование состояний входов/выходов (сканирование границ) позволяет разработчику системы проверять схему соединений на печатной плате с привлечением минимального количества специального контрольного оборудования. Сканирование возможно благодаря наличию возможности управления и отслеживания каждого входа и выхода на каждом кристалле с помощью набора последовательно просматриваемых защёлок. Каждый вход и выход подсоеди-

няется к своей защёлке, а защёлки соединяются в длинный регистр сдвига, так что данные могут считываться или записываться в них через последовательный тестовый порт. Для двунаправленных выводов возможна комбинация функций ввода и вывода. Взаимодействие с JTAG портом осуществляется с помощью сигналов:

- TCK (входной) – сигнал тактовой синхронизации операций тестовой логики. Используется для синхронизации данных в защёлках просмотра и управляющей последовательности тестового конечного автомата;
- TMS (входной) – выбор тестового режима. Первичный управляющий сигнал. Синхронный по отношению к TCK. Последовательность значений на TMS задает текущее состояние порта тестирования;
- TDI/TCLK (входной/выходной) – вход тестовых данных или вход синхронизации. Отсюда последовательные входные данные, подаются в защёлки просмотра;
- TDO/TDI (выходной/входной) – выход тестовых данных. На этот контакт поступают последовательные выходные данные, извлекаемые из защелок просмотра. Может быть запрограммирован для приёма входных данных.

В данной лабораторной работе JTAG интерфейс используется для загрузки во *flash*-память микроконтроллера программ, которые явились результатом их разработки в среде IDE Embedded Workbench.

7. Описание целевой микропроцессорной системы

Целевая МП система, принципиальная схема которой приведена на рис. 7, выполнена в виде платы, на которой расположены:

- микросхема U1 микроконтроллера с подключенными к ней кварцевыми резонаторами Q1, Q2 и Q3;
- микросхема U2 интерфейса с последовательным коммуникационным портом RS232;
- простые устройства ввода/вывода: одноразрядный источник цифрового ввода кнопка BUT и одноразрядный цифровой индикатор – светодиод LED;
- кнопка аппаратного сброса RST;
- стабилизированный источник напряжения 3.3 В, питаемый от внешнего источника переменного напряжения 4.5 В через разъём PWR_JACK;
- разъём JTAG порта.

Внешний источник напряжения используется в автономном (независящем от инструментальной ЭВМ) режиме работы. Когда целевая система связана с инструментальной ЭВМ, то напряжение питания 3.3 В поступает через JTAG порт.

Поскольку в микроконтроллер интегрированы аналого-цифровой и цифро-аналоговый преобразователи, то для них требуется постоянное опорное напряжение V+. Это напряжение создаётся с использованием источника 3.3 В и поступает на соответствующие входы МК микросхемы.

Необходимо различать аналоговую и цифровую «землю». Это связано с тем, что на плате действуют как аналоговые, так и цифровые сигналы. На принципиальной схеме (рис. 7) для этих двух видов заземления используются разные обозначения и они имеют только одну общую точку. Эта точка находится там, где показано сопротивление R1 = 0.

Для выдачи сигнала LED на светодиод и приёма сигнала от кнопки BUT используется порт P6 микроконтроллера (контакты P6.0 и P6.1 соответственно). Связь с портом RS232 (сигналы TXD0 и RXD0) осуществляется через порт P3 (контакты P3.4 и P3.5).

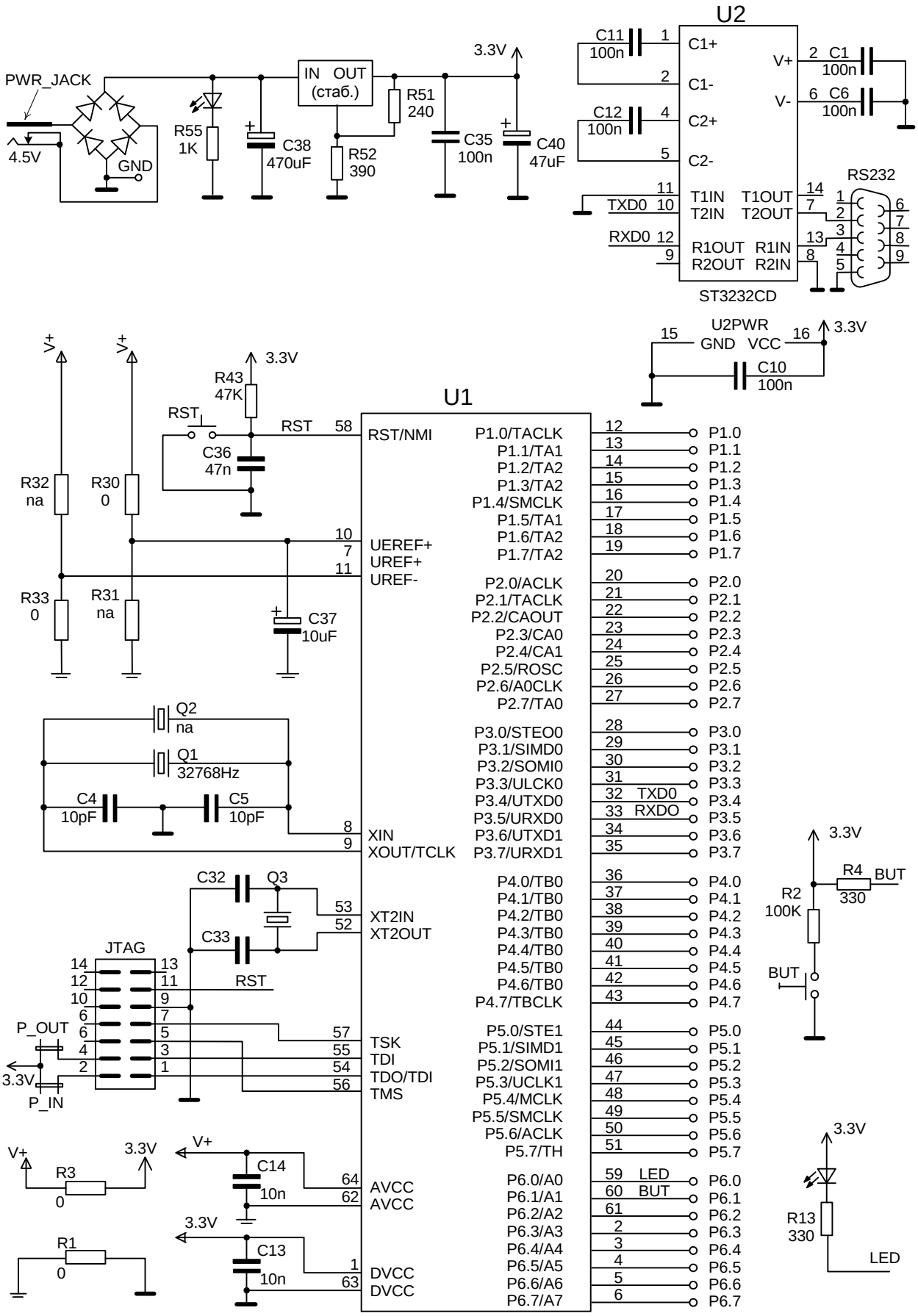

Рис. 7. Принципиальная схема целевой МП системы

8. Описание аппаратной и программной частей лабораторной установки

Общий вид лабораторной установки показан на рис. 1.

При работе с УВВ, входящими в состав целевой системы (таковыми являются кнопочный выключатель и светодиод), не требуется каких-либо дополнительных средств, кроме представленных на рис. 1. Связь с этими устройствами осуществляется через контакты P6.0 и P6.1. порта P6.

В дополнение к этому предусмотрена возможность ввода аналоговых сигналов, их цифровой обработки и вывода результата в аналоговом представлении. Для этой цели используются интегрированные в кристалл микроконтроллера АЦП и ЦАП. Сигналы на АЦП поступают от генератора стандартных сигналов (ГСС). Контроль сигналов, выводимых через ЦАП, осуществляется с помощью осциллографа. Для приёма сигналов от ГСС и передачи сигналов на осциллограф используются контакты порта P6 микроконтроллера, соединяемые с соответствующими измерительными приборами.

Для программирования целевой системы используется язык Ассемблера. Программа разрабатывается с помощью инструментальной ЭВМ и интегрированной среды разработки (IDE) IAR Embedded Workbench.

Создание проекта с использованием среды разработки начинается с того, что IDE предлагает шаблон (листинг 1) в соответствии, с которым можно построить программу на языке Ассемблера. Чтобы вызвать этот шаблон нужно, во-первых, создать новое "рабочее пространство" с помощью пункта меню **File->New->Workspace**, и, во-вторых, создать новый "проект" с помощью пункта меню **Project->CreateNewProject**. Из предложенного списка шаблонов нужно выбрать **asm** (ассемблерный проект). В шаблоне отмечены основные позиции, которых следует придерживаться при написании программы.

```
#include "msp430.h"           ; #define controlled include file
      NAME    main            ; module name
      PUBLIC  main            ; make the main label vissible
                              ; outside this module
      ORG   0FFFEh
      DC16  init              ; set reset vector to 'init' label
      RSEG  CSTACK            ; pre-declaration of segment
      RSEG  CODE              ; place program in 'CODE' segment
init: MOV   #SFE(CSTACK),SP    ; set up stack

main: NOP                      ; main program
      MOV.W   #WDTPW+WDTHOLD,&WDTCTL  ; Stop watchdog timer
      JMP $                    ; jump to current location '$'
                               ; (endless loop)
      END
```

Листинг 1

Вначале указан заголовочный файл msp430.h, который содержит необходимые определения констант. Константам в заголовочном файле соответствуют мнемонические обозначения адресов отображённых на память регистров ЦПЭ, регистров устройств ввода/вывода отдельных битов и битовых полей в регистрах ЦПЭ и УВВ. Введены также мнемонические обозначения адресов векторов прерывания.

Затем объявляется модуль main как доступный (PUBLIC) для других модулей. Делается указание компилятору установить адрес 0FFFEh (директива *ORG 0FFFEh*), соответствующий вектору прерывания при «сбросе» системы (RESET_VECTOR), с последующим размещением на этом месте точки входа init в процедуру инициализации (директива *DC16 init*).

Далее объявляется, что необходимо выделить память под стек (директива *RSEG CSTACK*) с указание на то, что выделенный под стек сегмент памяти будет перемещаемым.

В перемещаемой области памяти предлагается разместить и сегмент кода (директива *RSEG CODE*)

Программа инициализации микроконтроллера начинается с команды *MOV #SFE(CSTACK), SP* установки указателя стека SP на последнюю ячейку ранее выделенной области памяти с именем CSTACK, а основная программа – с метки main, где находится команда *NOP* (No Operation). В регистре управления сторожевым таймером WDTCTL устанавливаются биты WDTPW и WDTHOLD (см. их определения в файле msp430.h). В результате сторожевой таймер отключается (команда *MOV.W #WDTPW+WDTHOLD,&WDTCTL*). Возобновить его работу можно в процессе исполнения пользовательской программы.

Заканчивается шаблон кода переходом на метку (команда *JMP $*), которая является точкой входа в бесконечный цикл. Вместо него далее должна быть размещена основная часть программного кода для микроконтроллера.

Среда программирования IAR Embedded Workbench, как и многие другие, доступные для программирования микроконтроллеров, позволяет писать программы как на языке ассемблера, так и на более высоко организованных языках, например, C. В данной лабораторной работе будет использоваться язык ассемблера как наиболее прозрачный с точки зрения изучения архитектуры микроконтроллера.

9. Контрольные вопросы

1. Для чего нужны микроконтроллеры?
2. Чем отличается контроллер MSP430 от микропроцессоров архитектуры x86?
3. Чем отличается контроллер MSP430 от настольного персонального компьютера?
4. Чем определяется максимальная тактовая частота контроллера? Какова она? Сравните её с тактовой частотой современных процессоров для ПК, объясните причину такого различия.

5. Сравните систему команд MSP430 и современных процессоров для ПК. Какая из них богаче? Почему так сделано?
6. Какие периферийные устройства обычно размещают на одном чипе с микроконтроллером? Для чего это делается?
7. Что делает команда ассемблера **mov** и с аргументами какого типа она может работать?
8. Для чего микроконтроллеру нужны регистры? Какие регистры Вы знаете? Кратко поясните их функции.
9. Объясните, как используются шина адреса (ША) и шина данных (ШД), когда содержимое одной ячейки оперативной памяти пересылается в другую (например, при помощи команды mov)?
10. Чему равны константы WDTHOLD и BIT0? Как можно узнать значение любой константы из текста программы?

10. Задания

1. Ознакомьтесь с архитектурой микроконтроллера. Подготовьте ответы на контрольные вопросы.

2. Получите у преподавателя необходимое для выполнения работы оборудование, подключите его к компьютеру.

3. Запустите IAR Embedded Workbench и создайте шаблонный проект согласно рекомендациям параграфа 7. Объясните смысл появившихся на экране инструкций ассемблера.

Список литературы

1. Семенов Б.Ю. Микроконтроллеры MSP430. Первое знакомство, М.: Изд-во «Солон-пресс», 2006, 120 с.
2. Шкелёв Е.И. Электронные цифровые системы и микропроцессоры. Учебное пособие. //Н.Новгород: Изд.ННГУ, 2004, 152 с.
3. Цифровые процессоры обработки сигналов. Справочник. // Под редакцией А.Г. Остапенко. – М.: Радио и связь, 1994.
4. Электронная версия руководства пользователя (оригинал) // http://www.ti.com/lit/ug/slau144j/slau144j.pdf
5. Семейство микроконтроллеров MSP430x2xx: Архитектура. Программирование. Разработка приложений. М.:"Додека XXI". 2010.
6. Электронный учебный курс по MSP 430 // http://we.easyelectronics.ru/blog/msp430

Содержание

Составители:
Евгений Иванович **Шкелёв**
Владимир Андреевич **Калинин**
Владимир Владимирович **Пархачёв**

ЗНАКОМСТВО С МИКРОКОНТРОЛЛЕРОМ СЕРИИ MSP430

Практикум

Федеральное государственное автономное образовательное учреждение высшего профессионального образования «Нижегородский государственный университет им. Н.И. Лобачевского».
603950, Нижний Новгород, пр. Гагарина, 23.