Министерство образования Республики Беларусь
Учреждение образования
«Белорусский государственный университет
информатики и радиоэлектроники»

Кафедра электронной техники и технологии

С. К. Дик, М. В. Давыдов, А. С. Терех

ПРОЕКТИРОВАНИЕ НА ОСНОВЕ МИКРОКОНТРОЛЛЕРОВ

ЛАБОРАТОРНЫЙ ПРАКТИКУМ

для студентов специальности «Медицинская электроника»
дневной и заочной форм обучения

Минск БГУИР 2009

УДК 004.31(075.8)
ББК 32.973.26–018.2я73
Д45

Дик, С. К.

Д45 Проектирование на основе микроконтроллеров: лаб. практикум для студ. спец. «Медицинская электроника» днев. и заоч. форм обуч. / С. К. Дик, М. В. Давыдов, А. С. Терех. – Минск : БГУИР, 2009. – 44 с.: ил.

ISBN 978-985-488-405-9

Лабораторный практикум составлен в соответствии с программой дисциплины «Микроэлектронные схемы, микротехнологии и микропроцессоры в средствах медицинской электроники» и ставит своей целью изучение конструкции, принципов функционирования и программирования микропроцессоров. Состоит из четырех лабораторных работ и трех приложений к ним.

Предназначен для закрепления и углубления теоретических знаний, полученных на лекциях и в процессе самостоятельного изучения дисциплин, для приобретения практических навыков в области программирования микропроцессоров.

УДК 004.31(075.8)
ББК 32.973.26–018.2я73

ISBN 978-985-488-405-9

СОДЕРЖАНИЕ

Лабораторная работа №1

ГЕНЕРАЦИЯ ПРЯМОУГОЛЬНЫХ СИГНАЛОВ ЗАДАННОЙ ЧАСТОТЫ

1 Цель работы

Изучение структурной организации микроконтроллера Cygnal C8051F320, разработка программы для микроконтроллера Cygnal C8051F320 на языке программирования Ассемблер для генерации прямоугольных сигналов с заданной скважностью и частотой.

2 Теоретические сведения

Семейство микроконтроллеров C8051Fxxx оптимально подходит для построения устройств, требующих высокой производительности, точности измерений, большой степени интеграции и малого потребления. Они программно совместимы с 8051-м стандартом, но одновременно имеют высокую производительность – до 100 MIPS**.**

Энергонезависимая Flash-память программ может программироваться «в системе», т.е. на плате. В секторы по 512 байт Flash-памяти (размер Flash до 128K) могут записываться как программы, так и данные, которые становятся таким образом энергонезависимыми.

Микроконтроллеры имеют конструктивно встроенные интерфейсы: CAN (контроллеры серий F040 и F060), USB (серия F320), SMBus/I2C, UART, SPI. Имеется порт с повышенной нагрузочной способностью, 8, 10, 12, 16, 24-битные АЦП и 12-битные ЦАП. В моделях серии C8051F35X имеются сигма-дельта АЦП на 24 и 16 бит. Встроенная автономная отладочная система (JTAG) – полный внутрисхемный эмулятор «in-circuit» не задействует ресурсы кристалла и позволяет проверять и модифицировать память и регистры, устанавливать контрольные точки, временные точки, выполнять пошаговое выполнение и остановку программы.

В данной лабораторной работе будет использован микроконтроллер семейства C8051F320.

2.1 Краткий обзор микроконтроллера Cygnal семейства C8051F320

МК C8051F320 имеет встроенную схему сброса по включению питания, схему слежения за напряжением питания, регулятор напряжения, сторожевой таймер, тактовый генератор и представляет собой, таким образом, функционально законченную систему на кристалле. Имеется возможность внутрисхемного программирования Flash-памяти, что обеспечивает долговременное (энергонезависимое) хранение данных, а также позволяет осуществлять обновление программного обеспечения в готовых изделиях. Программа пользователя может полностью управлять всеми периферийными модулями, а также может ин-

дивидуально отключить любой из них с целью уменьшения энергопотребления. Встроенный двухпроводный Silicon Labs Development Interface (интерфейс C2) позволяет производить «неразрушающую» (не используются внутренние ресурсы) внутрисхемную отладку в режиме реального времени, используя МК, установленный в конечное изделие. Средства отладки обеспечивают проверку и модификацию памяти и регистров, расстановку точек остановок, пошаговое выполнение программы, а также поддерживают команды запуска и остановки. В процессе отладки с использованием интерфейса C2 все аналоговые и цифровые периферийные модули полностью сохраняют свою работоспособность. Два вывода интерфейса C2 могут использоваться для других пользовательских функций, что позволяет осуществлять внутрисистемную отладку, не занимая для этого отдельные выводы корпуса. Структурная схема МК C8051F320 изображена на рисунке 1.

Каждый МК предназначен для работы в промышленном температурном диапазоне (минус 40ºС…+85ºС) при напряжении питания 2,7 В…3,6 В (при этом для взаимодействия по шине USB требуется напряжение 3,0 В…3,6 В). На порты ввода/вывода и вывод /RST могут быть поданы входные сигналы напряжением до 5 В. МК C8051F320/1 выпускаются в 32-выводных корпусах типа LQFP и 28-выводных корпусах типа MLP.

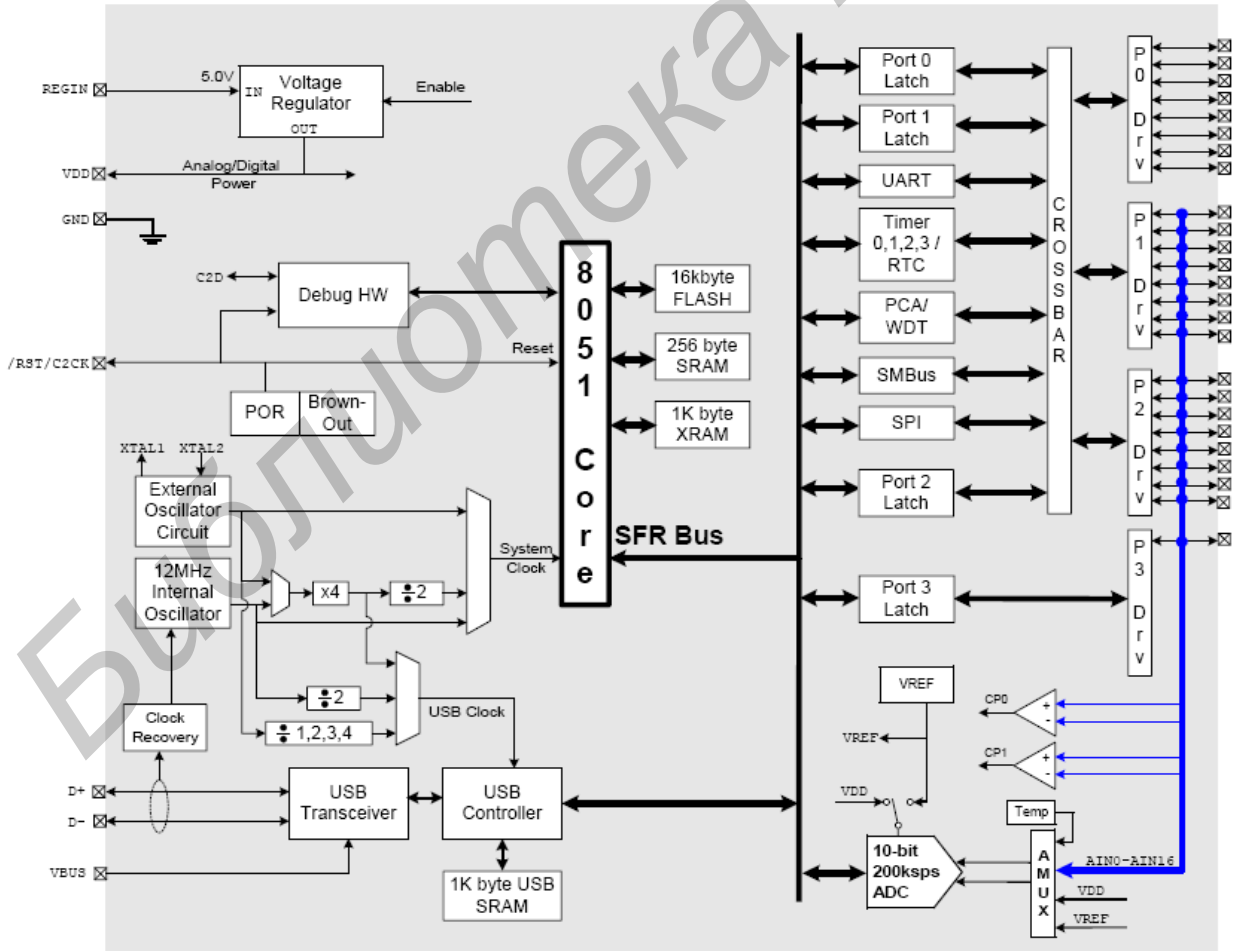

Рисунок 1 – Структурная схема C8051F320

Процессорное ядро CIP-51 по системе команд полностью совместимо с ядром MCS-51. Для разработки программного обеспечения могут использоваться стандартные 803x/805x ассемблеры и компиляторы. Процессорное ядро CIP-51 использует конвейерную архитектуру, что существенно повышает скорость выполнения команд по сравнению со стандартной архитектурой 8051. В МК с архитектурой 8051 все команды, кроме MUL и DIV, исполняются за 12 или 24 системных тактовых цикла при максимальной тактовой частоте 12…24 МГц. При работе на тактовой частоте 25 МГц производительность ядра CIP-51 может достигать 25 MIPS.

Ядро имеет стандартную (8051) структуру адресного пространства *памяти программ и данных*. В состав памяти входит ОЗУ объемом 256 байт, старшие 128 байт которого имеют двойную конфигурацию. В режиме косвенной адресации осуществляется доступ к старшим 128 байтам ОЗУ общего назначения, а в режиме прямой адресации осуществляется доступ к 128 байтам адресного пространства регистров специального назначения (SFR). Младшие 128 байт ОЗУ доступны как для прямой, так и для косвенной адресации. Из них первые 32 байта адресуются как четыре банка регистров общего назначения, а следующие 16 байт адресуются побайтно или побитно. Память программ МК состоит из 16 Кбайт Flash-памяти. Эта память может перепрограммироваться внутрисистемно секторами по 512 байт, не требуя при этом специального внешнего напряжения программирования. На рисунке 2 приведена карта распределения памяти МК.

Рисунок 2 – Карта распределения памяти

Программируемые цифровые порты ввода/вывода и матрица соединений

МК C8051F320 имеет 25 выводов, которые составляют три 8-разрядных порта и один 1-разрядный порт. Порты функционируют в соответствии со стандартом 8051 с некоторыми дополнительными возможностями. Каждый вывод порта можно настроить как аналоговый вход или как цифровой вход-выход. Выводы, настроенные как цифровые входы-выходы, можно, кроме этого, на-

строить как двухтактные цифровые выходы или выходы с открытым стоком. Также допускается глобальное отключение слаботоковых подтягивающих резисторов, что позволяет еще более снизить энергопотребление.

Цифровая матрица позволяет необходимым образом соединять внутренние цифровые системные ресурсы с выводами портов ввода/вывода. При помощи регистров управления матрицей на выводы портов могут быть выведены сигналы от внутренних таймеров/счетчиков, от последовательных интерфейсов, аппаратные прерывания, выходы компараторов и другие цифровые сигналы. Это позволяет выбрать точную комбинацию связей между портами ввода/вывода общего назначения и цифровыми ресурсами, необходимую для каждого конкретного приложения.

2.2 Обработка прерываний в микроконтроллерах Cygnal семейства C8051F320

Процессорное ядро CIP-51 имеет развитую систему прерываний, поддерживающую в общей сложности 16 источников прерываний с двумя уровнями приоритета. Распределение источников прерываний между встроенными периферийными модулями и внешними входными выводами зависит от конкретного типа МК. Каждый источник прерываний имеет один или несколько связанных с ним флагов прерываний, размещенных в SFR. Когда периферийный модуль или внешний источник прерываний регистрирует событие, удовлетворяющее условию прерывания, соответствующий флаг прерывания устанавливается в 1.

Если прерывание от источника прерываний разрешено, то при установке флага прерывания генерируется запрос прерывания. Как только выполнение текущей команды завершится, будет сгенерирована команда LCALL перехода по предопределенному адресу, откуда начнется исполнение процедуры обслуживания прерывания (Interrupt Service Routine – ISR). Каждая ISR должна заканчиваться командой RETI, которая возвращает управление прерванной программе и приводит к выполнению той команды, которая исполнилась бы, если бы запроса прерывания не было. Если прерывания не разрешены, флаг прерывания игнорируется и выполнение программы продолжается в нормальном режиме. (Флаг прерывания устанавливается в 1 независимо от того, разрешены прерывания или запрещены.)

Прерывание от каждого источника прерываний может быть разрешено или запрещено с помощью соответствующих бит разрешения прерываний в регистрах SFR (IE-EIE2). Прежде всего прерывания необходимо разрешить глобально установкой в 1 бита EA (IE.7), только после этого состояние индивидуальных флагов разрешения прерываний будет иметь силу. Сброс в 0 бита EA запрещает прерывания от всех источников прерываний независимо от состояния индивидуальных флагов разрешения прерываний.

Таблица 1 – Источники прерываний

Источник прерывания	**Вектор прерывания**	**Приоритет**	**Флаг прерывания**	**Битовая адресация?**	**Аппаратный сброс?**	**Бит разрешения**	**Управление приоритетом**
Сброс	0x0000	max	Нет	N/A	N/A	Разрешен всегда	Всегда наивысший
Внешнее прерывание 0 (/INT0)	0x0003	0	IE0 (TCON.1)	Y	Y	EX0 (IE.0)	PX0 (IP.0)
Переполнение Таймера 0	0x000B	1	TF0 (TCON.5)	Y	Y	ET0 (IE.1)	PT0 (IP.1)
Внешнее прерывание 1 (/INT1)	0x0013	2	IE1 (TCON.3)	Y	Y	EX1 (IE.2)	PX1 (IP.2)
Переполнение Таймера 1	0x001B	3	TF1 (TCON.7)	Y	Y	ET1 (IE.3)	PT1 (IP.3)
Последовательный порт УАПП0	0x0023	4	RI0 (SCON0.0) TI0 (SCON0.1)	Y	N	ES0 (IE.4)	PS0 (IP.4)
Переполнение Таймера 2	0x002B	5	TF2H (TMR2CN.7) TF2L (TMR2CN.6)	Y	N	ET2 (IE.5)	PT2 (IP.5)
Модуль SPI0	0x0033	6	SPIF (SPI0CN.7) WCOL (SPI0CN.6) MODF (SPI0CN.5) RXOVRN (SPI0CN.4)	Y	N	ESPI0 (IE.6)	PSPI0 (IP.6)
Модуль SMBus0	0x003B	7	SI (SMB0CN.0)	Y	N	ESMB0 (EIE1.0)	PSMB0 (EIP1.0)
Детектор Диапазона АЦП0	0x004B	9	ADWINT (ADC0CN.3)	Y	N	EWADC0 (EIE1.2)	PWADC0 (EIP1.2)
Завершение преобразования АЦП0	0x0053	10	ADC0INT (ADC0CN.5)	Y	N	EADC0 (EIE1.3)	PADC0 (EIP1.3)
Переполнение Таймера 3	0x0073	14	TF3H (TMR3CN.7) TF3L (TMR3CN.6)	N	N	ET3 (EIE1.7)	PT3 (EIP1.7)
Уровень VBUS	0x007B	15	N/A	N/A	N/A	EVBUS (EIE2.0)	PVBUS (EIP2.0)

Некоторые флаги прерываний сбрасываются автоматически аппаратными средствами при переходе к процедуре ISR. Однако большинство флагов прерываний не сбрасываются аппаратно и должны быть сброшены программно до возвращения из процедуры ISR. Если флаг прерывания остается установленным после завершения выполнения команды RETI, то сразу же будет сгенерирован новый запрос прерывания и после завершения выполнения следующей команды произойдет повторный переход к процедуре ISR.

Данное семейство МК поддерживает 16 источников прерываний. Программа может симулировать прерывание установкой в 1 любого флага прерывания. Если прерывание для этого флага разрешено, будут сгенерирован запрос прерывания и произойдет переход по адресу процедуры ISR, связанной с этим флагом прерывания. Источники прерываний МК, соответствующие им адреса прерываний, уровень приоритета и биты управления перечислены в таблице 1.

Задержка обработки прерывания

Время реакции на прерывание зависит от состояния процессорного ядра в момент возникновения прерывания. Опрос флага прерывания и декодирование приоритета осуществляется каждый системный тактовый цикл. Наименее возможное время реакции на прерывание составляет 5 тактовых циклов: 1 цикл для определения прерывания и 4 – для выполнения команды LCALL перехода к процедуре ISR.

Если в момент выполнения команды RETI появляется прерывание, то до выполнения команды LCALL перехода на процедуру обслуживания этого прерывания будет исполнена одна команда основной программы. Поэтому максимальное время реакции на прерывание (если в настоящий момент не обслуживается другое прерывание или если новое прерывание имеет более высокий приоритет) будет тогда, когда выполняется команда RETI, а следом за ней должна выполняться команда DIV. В этом случае время реакции составляет 18 тактовых циклов: 1 цикл для определения прерывания, 5 – для выполнения команды RETI, 8 – для выполнения команды DIV и 4 – для выполнения команды LCALL перехода на процедуру ISR.

Если выполняется процедура ISR для прерывания с равным или более высоким приоритетом, новое прерывание не будет обслужено до тех пор, пока не завершится текущая процедура ISR, включая команду RETI и следующую команду.

2.3. Таймер/счетчики в микроконтроллерах Cygnal семейства C8051F320

МК содержит четыре таймер/счетчика (Т/С): два из них представляют собой 16-разрядные Т/С, совместимые с Т/С стандартной архитектуры 8051, а два других являются 16-разрядными Т/С с режимом автоперезагрузки и предназначены для использования совместно с АЦП, SMBus, USB (для измерения фреймов), а также в качестве Т/С общего назначения. Эти Т/С можно использовать для измерения временных интервалов, подсчета внешних событий, а также для

генерации периодических запросов прерываний. Таймеры 0 и 1 почти идентичны и имеют четыре основных режима работы. Таймеры 2 и 3 могут работать (каждый) как один 16-разрядный таймер или как два 8-разрядных таймера, причем во всех случаях поддерживается режим автоперезагрузки. Рассмотрим подробно Таймеры 0 и 1.

***Режимы Таймеров* 0 *и* 1:**

13-разрядный Т/С
16-разрядный Т/С
8-разрядный Т/С с автоперезагрузкой
Два 8-разрядных Т/С (только Таймер 0)

Таймеры 0 и 1 могут тактироваться от одного из пяти источников, выбор которых осуществляется с помощью бит выбора режима таймера (T1M – T0M) и бит выбора коэффициента деления тактовой частоты (SCA1 – SCA0). Биты выбора коэффициента деления тактовой частоты настраивают предварительный делитель тактовой частоты, сигнал с выхода которого может использоваться для тактирования Таймеров 0 и/или 1 (таблица 1).

В качестве сигнала тактирования Таймеров 0 и 1 можно выбрать либо сигнал с выхода предварительного делителя тактовой частоты, либо системный тактовый сигнал. Таймеры 0 и 1 могут также функционировать как счетчики. В этом случае регистр таймер/счетчика инкрементируется под воздействием каждого перехода внешнего сигнала на выбранном входном выводе (T0 или T1) из состояния лог. 1 в состояние лог. 0. Могут подсчитываться импульсы с частотой до 1/4 системной тактовой частоты. Входной сигнал не обязательно должен быть периодическим, однако он должен удерживаться на заданном уровне как минимум в течение двух полных системных тактовых циклов, чтобы гарантировать его корректную выборку.

Каждый таймер реализован в виде 16-разрядного регистра, доступного как два отдельных байта: младший байт (TL0 или TL1) и старший байт (TH0 или TH1). Регистр управления Т/С (TCON) используется для включения Таймеров 0 и 1, а также для индикации их состояния. Прерывания от Таймера 0 можно включить установкой в 1 бита ET0 в регистре IE. Прерывания от Таймера 1 можно включить установкой в 1 бита ET1 в регистре IE. Оба таймера/счетчика работают в одном из четырех основных режимов, задаваемых битами выбора режима T1M1-T0M0 в регистре режима Т/С (TMOD). Каждый Т/С может быть настроен независимо от другого.

Рассмотрим подробно Режимы 0,1 и 2 (13-разрядный Т/С, 16-разрядный Т/С, 8-разрядный таймер/счетчик с автоперезагрузкой), которые будут использованы при выполнении лабораторной работы.

В *режиме 0* Таймеры 0 и 1 работают как 13-разрядный таймер/счетчик. Ниже приводится описание настройки и функционирования Таймера 0. Однако оба таймера идентичны, и Таймер 1 настраивается точно так же, как и Таймер 0 (рисунок 3).

Регистр TH0 содержит восемь старших бит 13-разрядного значения регистра T/C. Регистр TL0 содержит в разрядах TL0.4-TL0.0 пять младших бит 13-разрядного значения регистра T/C. Три старших бита регистра TL0 (TL0.7-TL0.5) не определены и должны маскироваться или игнорироваться при чтении регистра TL0. При инкрементировании 13-разрядного таймера и переполнении его из состояния 0x1FFF (все единицы) в состояние 0x0000 устанавливается в 1 флаг переполнения таймера TF0 (TCON.5) и будет сгенерировано прерывание, если оно разрешено. Бит C/T0 (TMOD.2) выбирает источник сигнала тактирования T/C0. Если бит C/T0 установлен в 1, то инкрементирование регистра таймера осуществляется под воздействием перехода внешнего сигнала на выбранном входном выводе (T0) из состояния лог. 1 в состояние лог. 0. Если бит C/T0 сброшен в 0, то в качестве источника тактирования T/C0 будет использоваться сигнал, определяемый битом T0M (CKCON.3). Если бит T0M установлен в 1, то Таймер 0 тактируется системным тактовым сигналом. Если бит T0M сброшен в 0, то в качестве источника тактирования T/C0 будет использоваться сигнал, определяемый битами настройки предварительного делителя в регистре CKCON (рисунок 3).

Установка в единицу бита TR0 (TCON.4) включит таймер, если либо бит GATE0 (TMOD.3) равен нулю, либо на внешнем выводе /INT0 присутствует сигнал с активным логическим уровнем, который определяется битом IN0PL в регистре INT01CF. После установки в единицу бита GATE0 управление таймером передается внешнему сигналу /INT0, что позволяет легко осуществлять измерение ширины импульсов (таблица 2).

Таблица 2 – Управление работой таймер-счетчика 0

TR0	**GATE0**	**/INT0**	**Таймер/Счетчик**
0	X	X	Отключен
1	0	X	Включен
1	1	0	Отключен
1	1	1	Включен

Примечание – X = не имеет значения

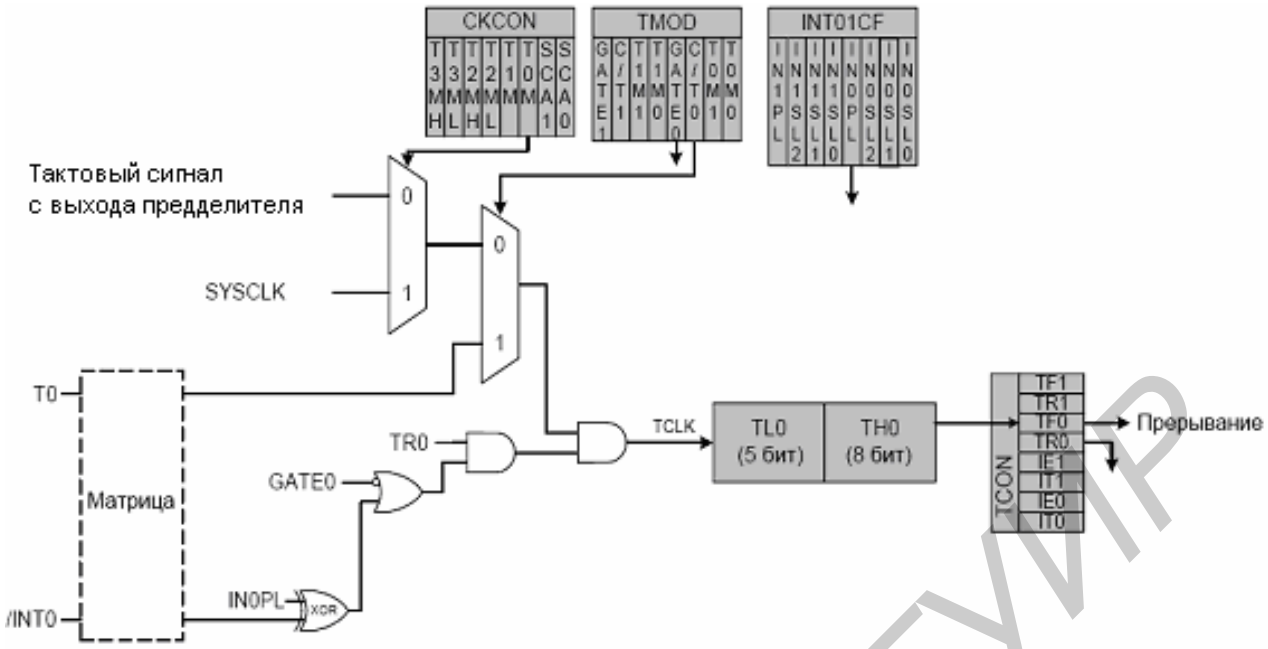

Рисунок 3 – Структурная схема Таймера 0 в режиме 0

Установка TR0 не сбрасывает таймер. Регистры таймера следует инициализировать необходимыми значениями до его включения.

TL1 и TH1 образуют 13-разрядный регистр Таймера 1 точно так же, как описано выше для регистров TL0 и TH0. Для настройки Таймера 1 и управления им используются соответствующие биты регистров TCON и TMOD таким же образом, как и для Таймера 0. Входной сигнал /INT1 используется совместно с Таймером 1; полярность /INT1 определяется битом IN1PL в регистре INT01CF.

Режим 1 аналогичен режиму 0 с тем лишь исключением, что регистры T/C используют все 16 бит. Таймеры/счетчики включаются и настраиваются в режиме 1 точно так же, как в режиме 0.

В режиме 2 Таймеры 0 и 1 настраиваются для работы в качестве 8-разрядных таймеров/счетчиков с автоматической перезагрузкой начального значения. Регистр TL0 содержит значение счетчика, а регистр TH0 содержит перезагружаемое значение. Когда счетчик в регистре TL0 переполняется (переходит из состояния 0xFF в состояние 0x00), флаг переполнения таймера TF0 (TCON.5) устанавливается в 1 и значение регистра TH0 загружается в регистр TL0. При установке флага TF0 будет сгенерировано прерывание, если оно разрешено. Перезагружаемое значение в регистре TH0 не изменяется. Чтобы первый отсчет был корректным, необходимо проинициализировать регистр TL0 требуемым значением до включения таймера. Таймер 1 в режиме 2 работает точно так же, как Таймер 0 (рисунок 4).

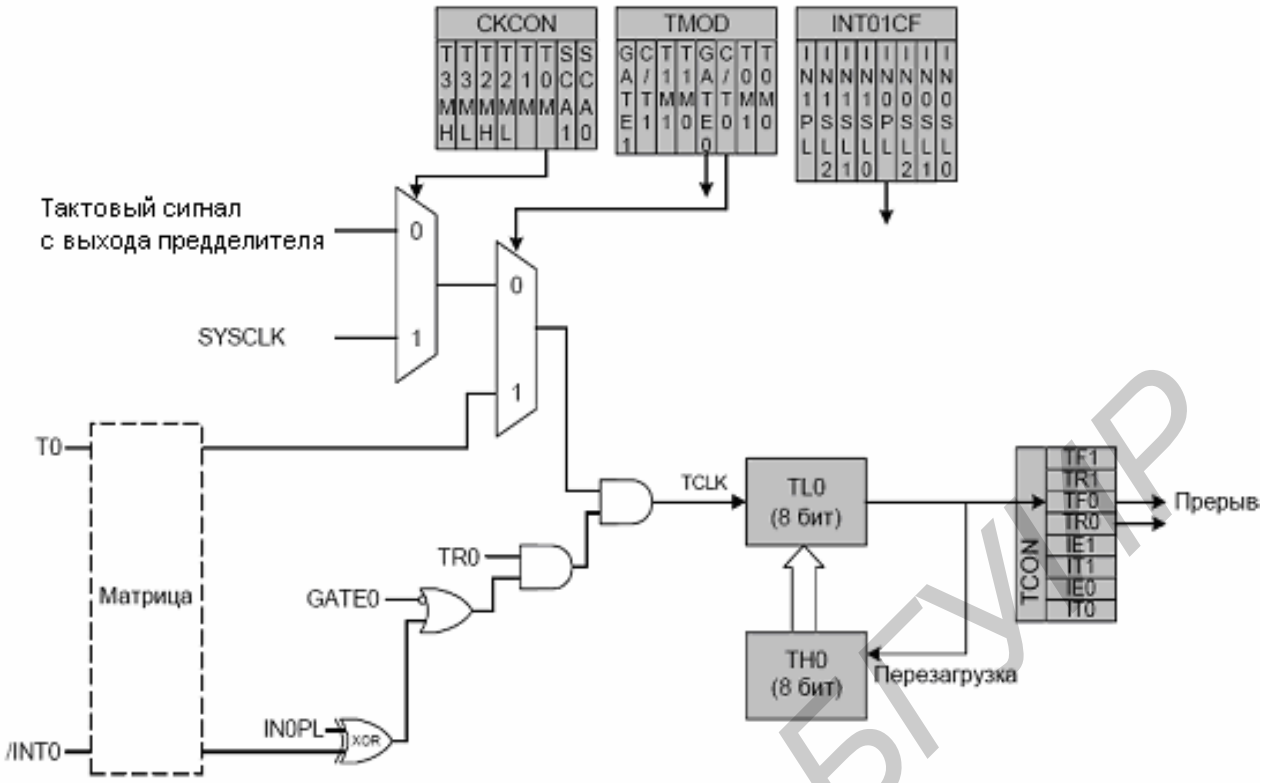

Рисунок 4 – Структурная схема Таймера 0 в режиме 2

В режиме 2 оба T/C включаются и настраиваются точно так же, как и в режиме 0. Установка в 1 бита TR0 (TCON.4) включит таймер, если либо бит GATE0 (TMOD.3) равен нулю, либо на внешнем выводе /INT0 присутствует сигнал с активным логическим уровнем, который определяется битом IN0PL в регистре INT01CF.

3 Приборы и оборудование

В данной лабораторной работе используется:

1 Комплект разработки для микроконтроллера C8051F320.

2 Персональный компьютер с USB портом.

3 Программное обеспечение.

4 Порядок выполнения работы

1 Запустить среду разработки программ для микроконтроллеров SiLabs.

2 Подготовить комплект отладки и ПК. Подключить плату к программатору, подключить программатор к компьютеру по интерфейсу USB. Подать питание на плату с помощью 9В блока питания. Установить связь МК с компьютером.

3 Включить программу main_TimerWork.asm в проект.

4 Провести инициализацию микроконтроллера.

4.1 Инициализация кварцевого резонатора (регистр OSCICN).

4.2 Инициализация порта (регистры XBR0, XBR1, PnMDIN, PnMDOUT, PnSKIP).

4.3 Инициализация таймера (регистры TCON, TMOD).

4.4 Инициализация прерывания (регистры IE, IP).

5 Установить значения в таймеры для генерации сигнала необходимой частоты (регистры TH0, TL0), рассчитанные исходя из заданной частоты мерцания светодиода (частота задается в таблице 2).

6 Скомпилировать программу.

7 Выполнить построение проекта.

8 Записать программу во внутреннюю память МК.

9 Запустить работу программы (кнопка Run среды SiLabs).

Таблица 2 – Варианты заданий

Вариант, №	1	2	3	4	5	6	7
Частота, Гц	1	2	0,5	1,3	1,5	2,5	3

5 Содержание отчета

1 Цель работы.

2 Листинг программы генерации сигнала.

3 Выводы.

6 Контрольные вопросы

1 Какова структура встроенной памяти микроконтроллера?

2 Перечислите источники прерываний микроконтроллера и опишите регистры прерываний.

3 Какие существуют порты ввода/вывода, какие регистры используются для их управления?

4 Как осуществляется инициализация портов ввода/вывода?

5 Какие существуют режимы работы Таймеров 0 и 1?

6 Перечислите основные регистры Таймеров 0 и 1, опишите их назначение.

Лабораторная работа №2

ИЗУЧЕНИЕ РАБОТЫ ПОРТОВ ВВОДА/ВЫВОДА

1 Цель работы

Изучение организации работы портов ввода/вывода микроконтроллера Cygnal C8051F320 и разработка программы для подключения кнопок управления.

2 Теоретические сведения

Порты ввода/вывода в микроконтроллерах Cygnal семейства C8051F320

Для доступа к аналоговым и цифровым ресурсам МК используются 17 внешних линий ввода/вывода, которые организованы как два 8-разрядных порта и один 1-разрядный порт. Каждый из выводов портов можно определить как порт ввода/вывода общего назначения (GPIO) или аналоговый вход; выводы портов P0.0 – P1.7 можно назначить одному из внутренних цифровых модулей, как показано на рисунках 1 и 3. Разработчик системы сам определяет, какие цифровые ресурсы будут назначены внешним выводам, ограничиваясь только количеством доступных выводов. Такая гибкость при распределении ресурсов достигается благодаря использованию приоритетного декодера матрицы. Следует иметь в виду, что состояние на внешнем выводе порта всегда можно прочитать из соответствующего регистра-защелки порта независимо от настройки матрицы.

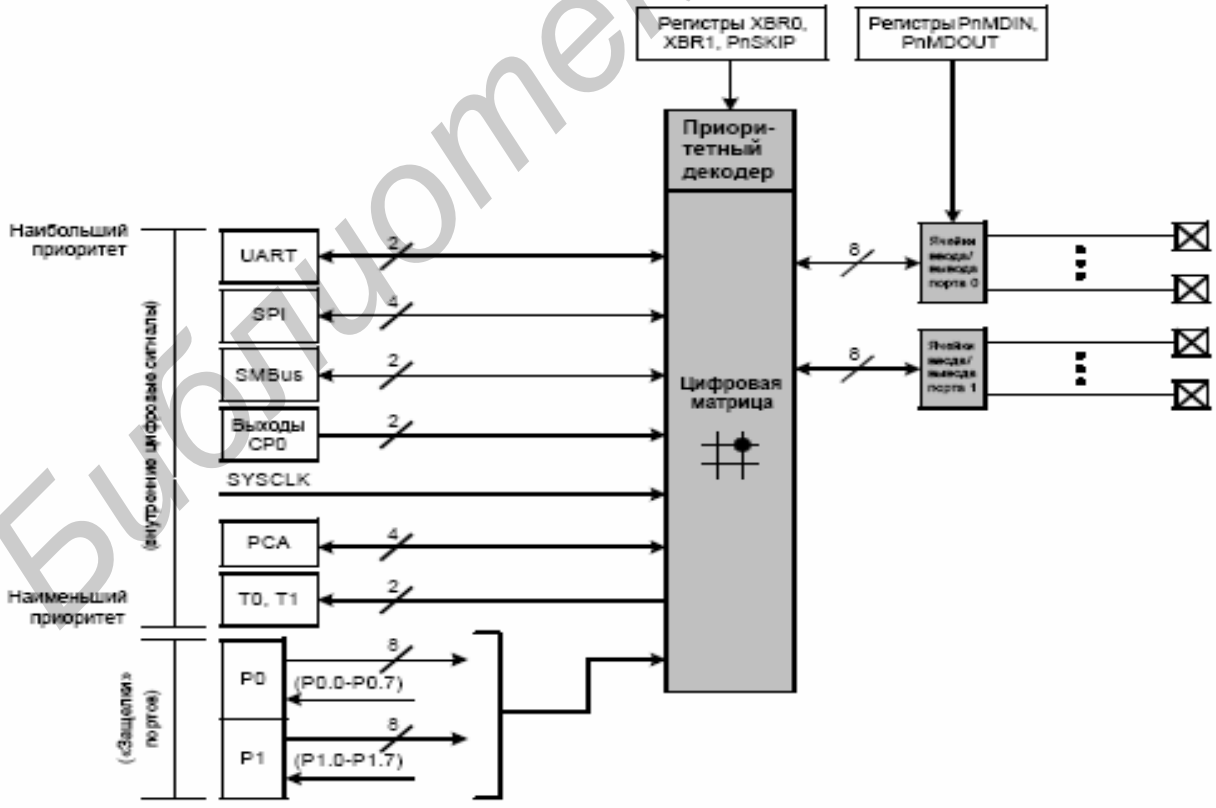

Рисунок 1 – Функциональная схема портов ввода/вывода

Матрица назначает внешние порты ввода/вывода выбранным внутренним цифровым ресурсам, используя приоритетный декодер (см. рисунки 3 и 4). Для выбора внутренних цифровых ресурсов используются регистры XBR0 и XBR1.

Допустимое напряжение на внешних выводах портов составляет 5В (см. схему ячейки порта на рисунке 2). С помощью регистров настройки выходов портов (PnMDOUT, где n = 0,1) можно настроить выходные драйверы портов ввода/вывода либо как обычные цифровые выходы, либо как выходы с открытым стоком.

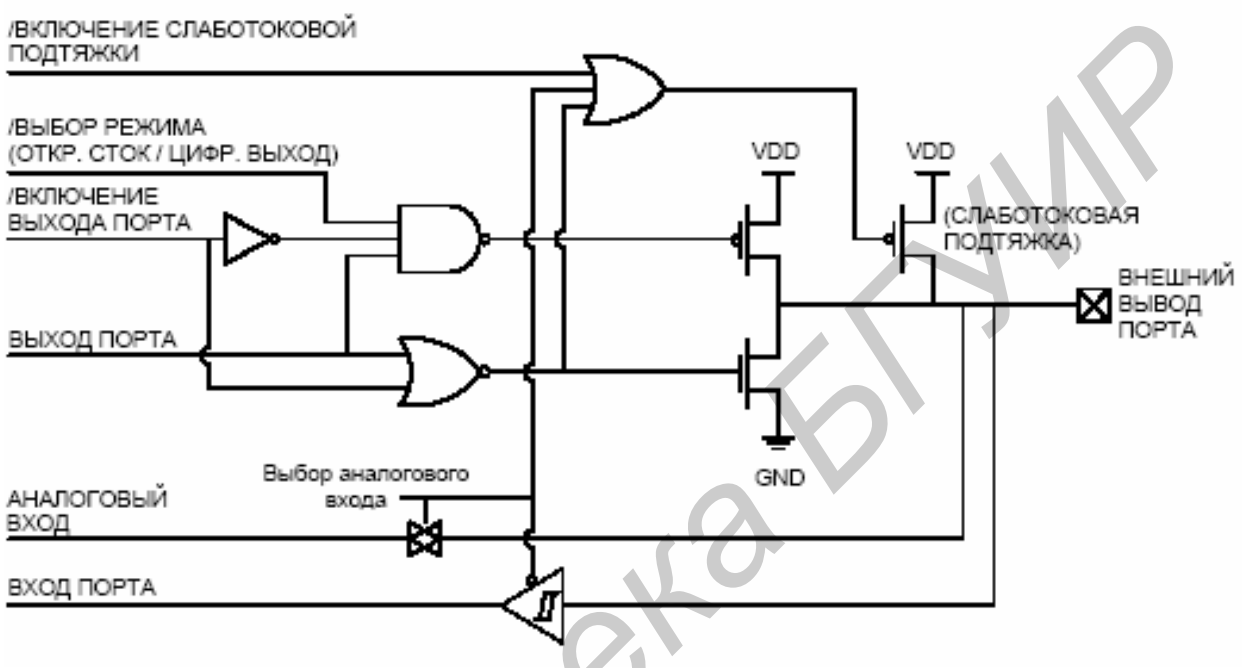

Рисунок 2 – Структурная схема ячейки порта ввода/вывода

Приоритетный декодер матрицы

Приоритетный декодер матрицы (см. рисунок 3) назначает приоритет каждой функции ввода/вывода, начиная с выводов UART0. Если какой-либо цифровой ресурс выбран, то этому ресурсу назначается еще не назначенный внешний вывод с наименьшим приоритетом (кроме UART0, которому всегда назначаются выводы P0.4 и P0.5). Если вывод порта назначен, то матрица пропускает этот вывод при назначении выводов следующему выбранному ресурсу. Кроме того, матрица будет пропускать выводы портов, если соответствующие им биты в регистрах PnSKIP установлены в 1. Регистры PnSKIP позволяют программе настроить матрицу таким образом, чтобы она пропускала выводы портов, которые используются в качестве аналоговых входов, портов ввода/вывода общего назначения или для реализации специальных функций.

Важное замечание относительно конфигурации матрицы: если вывод порта закреплен за периферийным модулем без использования матрицы, то соответствующий ему бит в регистрах PnSKIP должен быть установлен в 1. Это касается P0.0, если используется VREF, P0.3 и/или P0.2, если включена схема возбуждения внешнего генератора, P0.6, если АЦП или ЦАП настроены

на использование внешнего сигнала запуска преобразования (CNVSTR), а также любых выбранных входов АЦП или компаратора. Матрица пропускает выбранные выводы, как если бы они были уже назначены, и переходит к следующему неназначенному выводу. На рисунке 3 показаны приоритеты декодера матрицы без пропуска каких-либо выводов портов (P0SKIP, P1SKIP = 0x00); на рисунке 4 показаны приоритеты декодера матрицы с пропуском выводов XTAL1 (P0.2) и XTAL2 (P0.3) (P0SKIP = 0x0C).

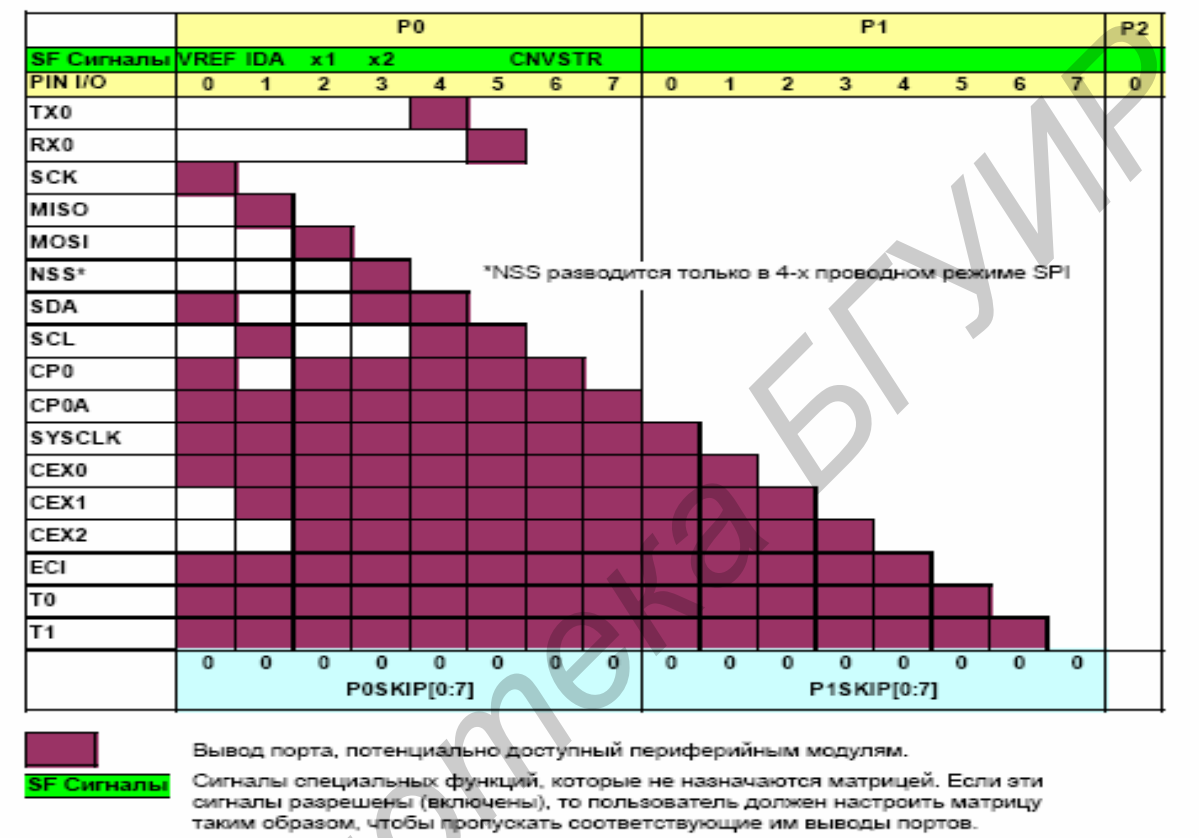

Рисунок 3 – Приоритетный декодер матрицы (нет пропускаемых выводов)

Регистры XBR0 и XBR1 используются для назначения цифровых ресурсов внешним портам ввода/вывода. Следует иметь в виду, что если выбран SMBus, то матрица назначает оба вывода, связанные с модулем SMBus (SDA и SCL); если выбран UART0, то матрица назначает оба вывода, связанные с модулем UART0 (TX и RX). Назначение выводов UART0 фиксировано с целью обеспечения возможности самозагрузки: TX0 всегда назначается выводу P0.4; RX0 всегда назначается выводу P0.5. После назначения приоритетных функций следуют стандартные порты ввода/вывода.

Важное замечание: модуль SPI может функционировать в 3- или 4-проводном режиме в зависимости от состояния бит NSSMD1–NSSMD0 в регистре SPI0CN. В соответствии с режимом работы SPI сигнал NSS либо разводится, либо не разводится на внешний порт.

Порт ввода/вывода общего назначения

Выводы портов, которые не назначены матрицей и не используются аналоговыми периферийными модулями, можно использовать в качестве выводов ввода/вывода общего назначения. Порты 3–0 доступны с помощью соответствующих SFR-регистров как в побайтном, так и в побитном режимах адресации. При записи в порт значение, записываемое в SFR-регистр, «защелкивается»; это позволяет удерживать на каждом выводе порта выходное значение. При чтении логические уровни входных выводов портов возвращаются независимо от значений регистров XBRn (т.е. даже если вывод назначен матрицей другому сигналу, регистр порта все равно может прочитать логическое состояние на соответствующем входе). Исключением являются команды типа *чтение-модификация-запись*. При работе с SFR-регистром порта командами типа *чтение-модификация-запись* являются следующие команды: ANL, ORL, XRL, JBC, CPL, INC, DEC, DJNZ, а также MOV, CLR или SETB, если они адресуют отдельный бит в SFR-регистре порта. В случае использования этих команд считывается, модифицируется и записывается обратно значение регистра (а не вывода).

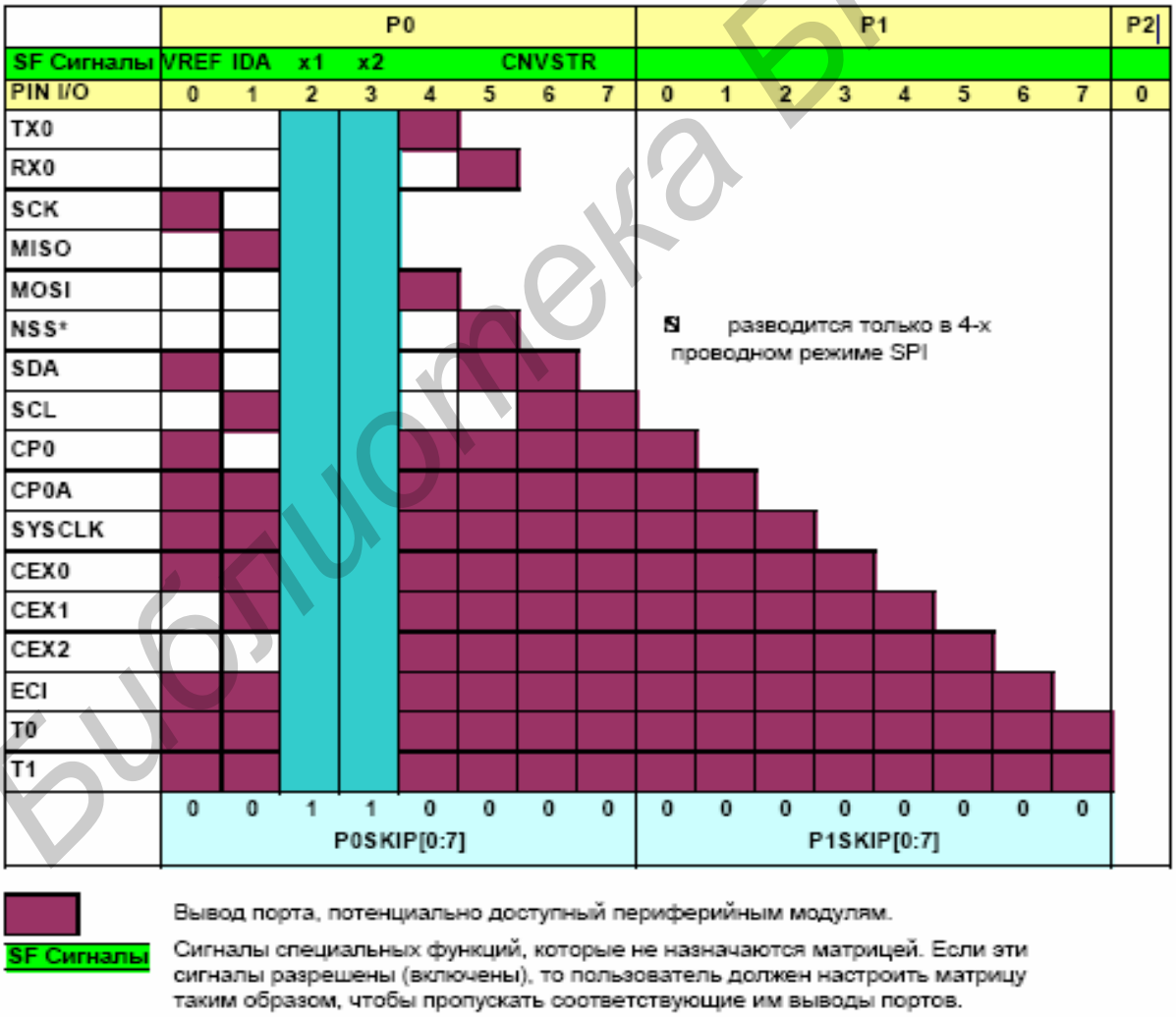

Рисунок 4 – Приоритетный декодер матрицы (пропускаются выводы подключения кварцевого резонатора)

Инициализация порта ввода/вывода осуществляется следующим образом:

1 Выбрать тип входа (аналоговый или цифровой) для всех выводов порта, используя регистр настройки входов порта (PnMDIN).

2 Выбрать тип выхода (с открытым стоком или двухтактный цифровой) для всех выводов порта, используя регистр настройки выходов порта (PnMDOUT).

3 Выбрать все выводы, которые должны пропускаться матрицей при назначении выводов, используя регистры выбора пропускаемых выводов (PnSKIP).

4 Назначить выводы порта требуемым периферийным модулям (XBR0, XBR1).

5 Включить матрицу (XBARE = ‘1’).

Все выводы порта должны быть настроены как аналоговые или как цифровые входы. Любые выводы, используемые в качестве входов компаратора или АЦП, должны быть настроены как аналоговые входы. Если вывод настроен как аналоговый вход, то его слаботоковая подтяжка, цифровой драйвер и цифровой приемник отключаются. Это позволяет снизить энергопотребление и уменьшить уровень шумов на аналоговом входе.

Выводы, настроенные как цифровые входы, могут использоваться аналоговыми периферийными модулями, однако это не рекомендуется. Чтобы настроить вывод порта как цифровой вход, следует сбросить в 0 соответствующий бит в регистре PnMDOUT и установить в 1 соответствующий бит регистра-защелки порта (регистр Pn).

Кроме этого, все аналоговые входы необходимо настроить таким образом, чтобы они пропускались матрицей при назначении выводов (это достигается установкой в 1 соответствующих бит в регистрах PnSKIP).

Тип входа устанавливается с помощью соответствующих бит регистра PnMDIN (1 – цифровой вход, 0 – аналоговый вход). При сбросе все выводы настраиваются по умолчанию как цифровые входы. Параметры выходных драйверов выводов порта задаются с помощью регистров настройки выходов порта (PnMDOUT). Выходной драйвер каждого порта можно настроить либо как цифровой двухтактный выход, либо как выход с открытым стоком. Такая настройка не осуществляется автоматически, ее необходимо выполнить даже для цифровых ресурсов, выбранных в регистрах XBRn. Единственным исключением из этого правила являются выводы SMBus (SDA, SCL), которые настраиваются как выходы с открытым стоком независимо от значения PnMDOUT. Если бит WEAKPUD в регистре XBR1 сброшен в 0, то слаботоковая подтяжка отключается у всех выводов портов, настроенных как выходы с открытым стоком. Бит WEAKPUD не влияет на выводы, настроенные как цифровые двухтактные выходы. Более того, слаботоковая подтяжка отключается у выхода, на который выведен лог. 0, чтобы предотвратить нежелательное увеличение энергопотребления.

Для выбора цифровых ресурсов, требуемых для конкретного проекта, необходимо загрузить регистры XBR0 и XBR1 соответствующими значениями. Установка в единицу бита XBARE в регистре XBR1 включает матрицу. До

включения матрицы внешние выводы остаются стандартными портами ввода/вывода (настроенными на вход) независимо от значений регистров XBRn. Имея значение регистров XBRn, можно определить разводку выводов, используя таблицу декодирования приоритетов.

Важное замечание: чтобы использовать порты P0, P1 и P2.0 – P2.3 как стандартные порты ввода/вывода в режиме выходов, необходимо включить матрицу. Выходные драйверы этих портов отключаются при выключении матрицы. P2.4 – P2.7 и P3.0 всегда функционируют как стандартные порты ввода/вывода общего назначения.

3 Приборы и оборудование

В данной лабораторной работе используется:

1 Комплект разработки для микроконтроллера C8051F320.

2 Персональный компьютер с USB портом.

3 Программное обеспечение.

4 Порядок выполнения работы

1 Запустить среду разработки программ для микроконтроллеров SiLabs.

2 Подготовить комплект отладки и ПК. Подключить плату к программатору, подключить программатор к компьютеру по интерфейсу USB. Подать питание на плату с помощью 9В блока питания. Установить связь МК с компьютером.

3 Включить программу main_TimerWork2.asm в проект.

4 Провести инициализацию микроконтроллера.

4.1 Инициализация внутреннего кварцевого резонатора (регистр OSCICN).

4.2 Инициализация порта (регистры XBR0, XBR1, PnMDIN, PnMDOUT, PnSKIP) для подключения светодиода и кнопки.

4.3 Инициализация таймеров 0 и 1 (регистры TCON, TMOD) в режимах 1 и 2.

4.4 Инициализация прерывания (регистры IE, IP) от таймера/счетчика

5 Установить значения в таймеры для генерации сигнала необходимой частоты (регистры TH0, TL0).

6 Запустить работу программы (кнопка Run среды).

7 Разработать алгоритм управления кнопкой (при нажатии частота мерцания светодиода изменяется в соответствии с заданием (таблица 1)).

8 Написать программу управления кнопкой, включающую подпрограмму гашения дребезга контактов.

9 Скомпилировать программу.

10 Выполнить построение проекта.

11 Записать программу во внутреннюю память МК.

12 Запустить работу программы (кнопка Run среды SiLabs).

Таблица 1 – Варианты заданий

Вариант, №	1	2	3	4	5	6	7
Частота, Гц	1–2	2–3	0,5–3	1–3	2–0,5	1–2,5	3–1,5

5 Содержание отчета

1 Цель работы.

2 Алгоритм программы управления кнопкой.

3 Листинг программы управления кнопкой.

4 Выводы.

6 Контрольные вопросы

1 Какие существуют порты ввода/вывода, какие регистры используются для их управления?

2 Принцип работы приоритетного декодера матрицы.

3 Как осуществляется инициализация портов ввода/вывода?

4 Какие выводы используются в качестве портов ввода/вывода общего назначения, и с помощью каких регистров осуществляется к ним доступ?

5 Для чего применяются алгоритмы гашения дребезга контактов?

Лабораторная работа №3

АНАЛОГО-ЦИФРОВОЕ ПРЕОБРАЗОВАНИЕ СИГНАЛА И СОХРАНЕНИЕ ЕГО В ПАМЯТИ МИКРОКОНТРОЛЛЕРА

1 Цель работы

Изучение АЦП и структуры внутренней памяти данных микроконтроллера Cygnal C8051F320, разработка программы на языке программирования Ассемблер для аналого-цифрового преобразование сигнала и сохранения этого сигнала в памяти микроконтроллера.

2 Теоретические сведения

10-разрядный АЦП (АЦП0). Модуль АЦП0 МК C8051F320 состоит из двух 17-канальных аналоговых мультиплексоров (обозначаются вместе как AMUX0) и 10-разрядного АЦП последовательного приближения (максимальная производительность – 200 тыс. преобразований в секунду) с интегрированным устройством выборки-хранения (УВХ) и программируемым детектором диапазона. AMUX0, режимы преобразования и детектор диапазона настраиваются программным путем при помощи регистров специального назначения (рисунок 1). АЦП0 функционирует как в однофазном, так и в дифференциальном режимах, и может быть настроен на измерение напряжения на выводах P1.0 – P3.0, выходного напряжения датчика температуры или напряжения VDD относительно P1.0 – P3.0, VREF или GND. Модуль АЦП0 включен только тогда, когда бит AD0EN регистра управления АЦП0 (ADC0CN) установлен в 1. Сброс этого бита в 0 переводит АЦП0 в режим отключения с пониженным энергопотреблением.

Аналоговый мультиплексор. AMUX0 осуществляет выбор положительного и отрицательного входов АЦП. В качестве положительного входа можно выбрать:

– P1.0 – P3.0;
– выходной сигнал встроенного датчика температуры;
– положительное напряжение питания (VDD).

В качестве отрицательного входа можно выбрать:

– P1.0 – P3.0;
– VREF;
– общий вывод питания GND.

Если в качестве отрицательного входа выбран GND, то АЦП0 функционирует в однофазном режиме; в остальных случаях АЦП0 функционирует в дифференциальном режиме. Входные каналы АЦП0 выбираются в регистрах AMX0P и AMX0N (см. ниже).

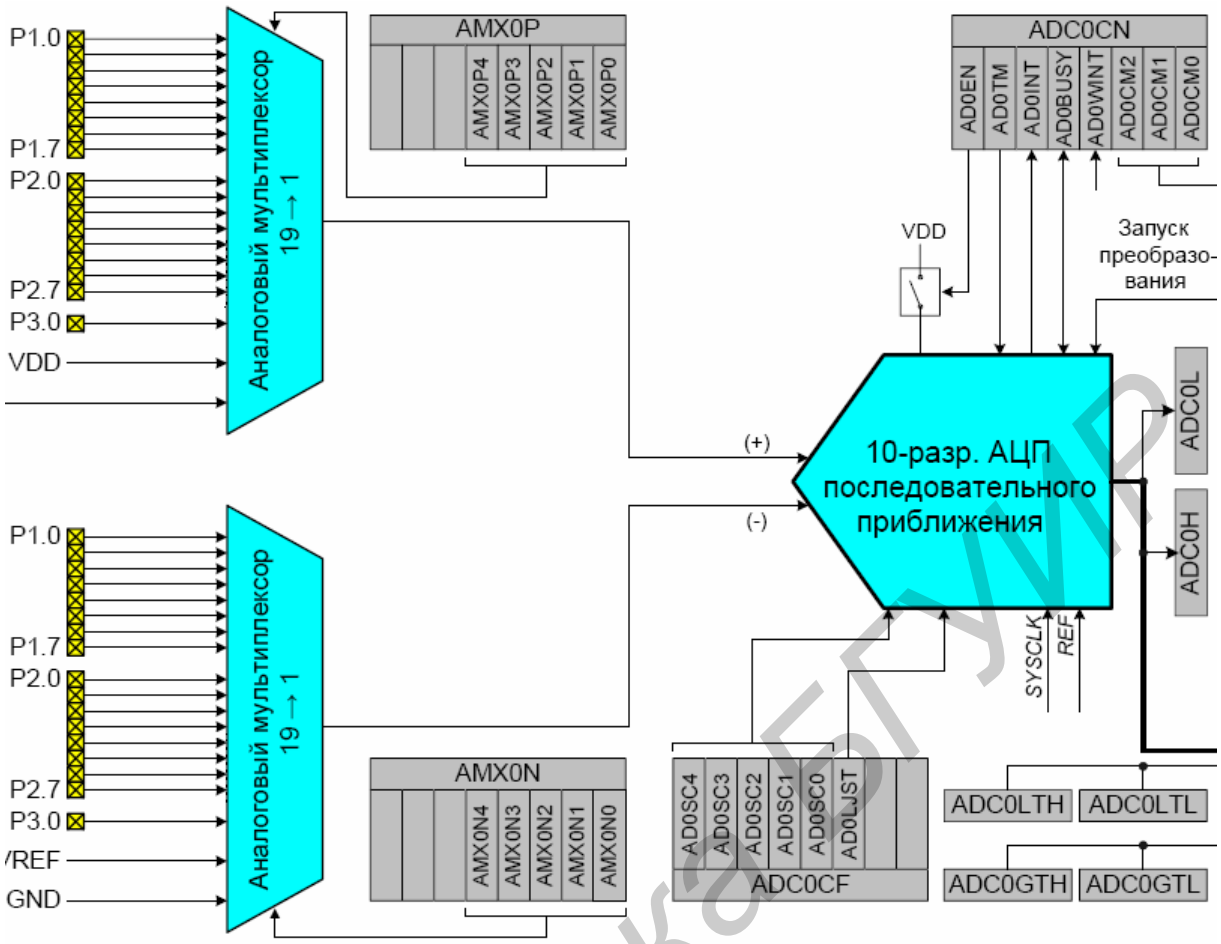

Рисунок 1– Функциональная схема АЦП0

Формат получаемого результата преобразования различен для однофазного и дифференциального режимов. По окончании каждого преобразования в регистры ADC0H и ADC0L записываются соответственно старший и младший байты результата преобразования. Данные могут быть выровнены вправо или влево в зависимости от значения бита AD0LJST (ADC0CN.0). В однофазном режиме результаты преобразований представлены в виде 10-разрядных целых чисел без знака. Диапазон измерения входных напряжений: 0…VREF*1023/1024. В дифференциальном режиме результаты преобразований представлены в виде 10-разрядных чисел в дополнительном коде со знаком. Диапазон измерения входных напряжений: – VREF … VREF*511/512.

Важное замечание относительно настройки входов АЦП0: выводы порта, выбранные в качестве входов АЦП0, должны быть настроены как аналоговые входы, и должны пропускаться матрицей при назначении выводов. Чтобы настроить вывод порта как аналоговый вход, следует сбросить в 0 соответствующий бит в регистре PnMDIN (для n = 0, 1, 2, 3). Чтобы заставить матрицу пропускать вывод порта при назначении выводов, следует установить в 1 соответствующий бит в регистре PnSKIP (для n = 0, 1, 2). Более подробная информация о настройке порта ввода/вывода приведена в лабораторной работе №1.

Максимальная скорость преобразования АЦП0 – 200 тыс. преобразований в секунду. Частота дискретизации АЦП0 определяется частотой системного тактового сигнала, деленной на значение, задаваемое битами AD0SC регистра ADC0CF, т.е. SYSCLK/(AD0SC + 1) для $0 \leq AD0SC \leq 31$.

Запуск преобразования может быть осуществлен одним из шести способов в зависимости от состояния бит режима запуска преобразования АЦП0 (AD0CM2-0) в регистре ADC0CN. Преобразование может быть инициировано:
1) установкой в единицу бита AD0BUSY в регистре ADC0CN;
2) переполнением Таймера 0 (т.е. непрерывное по времени преобразование);
3) переполнением Таймера 2;
4) переполнением Таймера 1;
5) нарастающим фронтом внешнего входного сигнала CNVSTR (вывод P0.6);
6) переполнением Таймера 3.

Установка в 1 (единицу) бита AD0BUSY позволяет осуществлять программное управление АЦП0, т.е. выполнять преобразования «по требованию». Бит AD0BUSY устанавливается в 1 во время преобразования и сбрасывается в 0 после окончания преобразования. При сбросе бита AD0BUSY инициируется прерывание (если оно разрешено) и устанавливается флаг прерывания от АЦП0 (AD0INT). (При определении окончания преобразования методом опроса следует использовать флаг прерывания от АЦП0 (AD0INT)). Преобразованные данные доступны в регистрах старшего и младшего слова данных АЦП0, ADC0H и ADC0L соответственно, когда AD0INT=1. Следует иметь в виду, что если запуск преобразования осуществляется переполнением Таймера 2 или 3, то используются переполнения младшего байта, когда Таймер 2 или 3 работает в 8-разрядном режиме, и переполнения старшего байта, когда Таймер 2 или 3 работает в 16-разрядном режиме.

Важное замечание относительно использования CNVSTR: входной вывод CNVSTR функционирует так же, как вывод порта P0.6. Если вход CNVSTR используется для запуска преобразования АЦП0, то вывод порта P0.6 должен пропускаться матрицей при назначении выводов. Чтобы заставить матрицу пропускать P0.6, следует установить в единицу бит 6 в регистре P0SKIP.

Время установления. Если конфигурация входов АЦП0 изменяется (т.е. изменяются настройки AMUX0), то после этого для обеспечения точности преобразования необходимо выдержать паузу длительностью не менее минимального времени установления сигнала. Время установления определяется сопротивлением AMUX0, емкостью накопительного конденсатора УВХ, сопротивлением внешнего источника сигнала и требуемой точностью преобразования. Следует отметить, что в энергосберегающем режиме выборки-хранения после запуска каждого преобразования выборка длится три периода сигнала дискретизации АЦП. Для большинства приложений эти три периода сигнала дискретизации будут соответствовать требованиям, предъявляемым ко времени установления.

На рисунке 2 показаны эквивалентные схемы входов АЦП0 как для дифференциального, так и для однофазного режимов работы. Следует отметить, что эквивалентная постоянная времени для обоих схем одинакова. Требуемое время установления для заданной точности установления (settling accuracy – SA) можно приблизительно определить из уравнения t = ln(2n/SA) x RTOTALCSAMPLE. Если измеряется выходное напряжение датчика температуры или напряжение VDD относительно GND, то RTOTAL = RMUX, где SA – точность установления, задаваемая в долях МЗР (например, 0.25 для установления в пределах ¼ МЗР); t – требуемое время установления в секундах RTOTAL – сумма сопротивления AMUX0 и сопротивления внешнего источника сигнала; n – разрешение АЦП в битах (10).

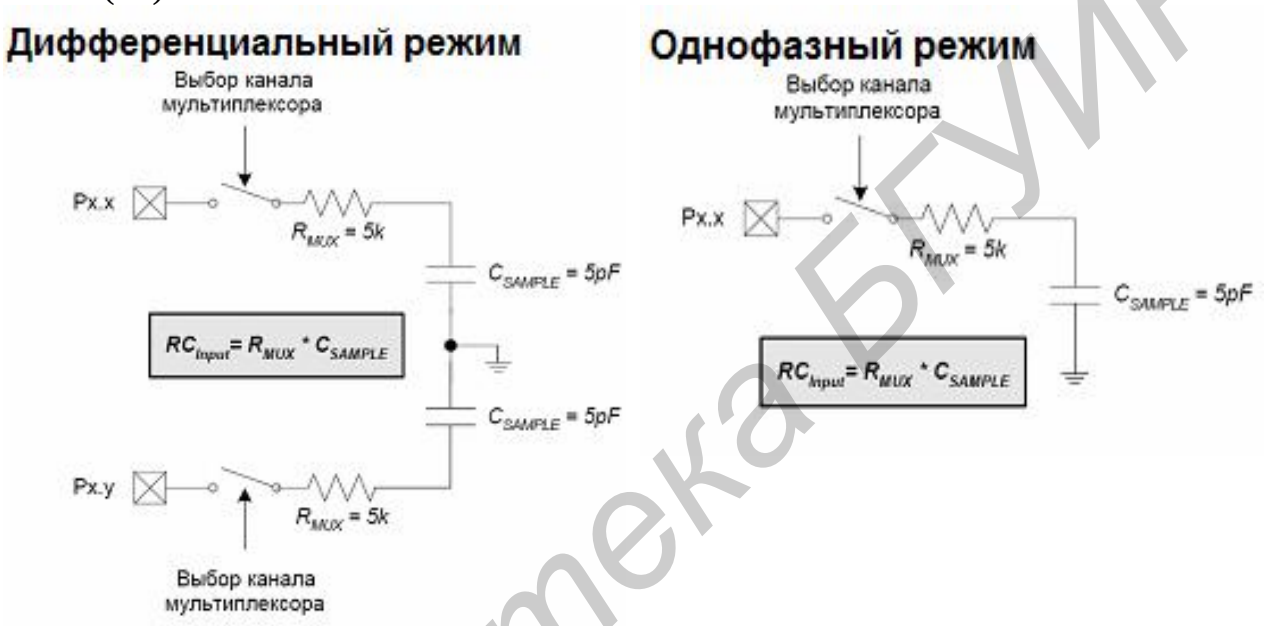

Рисунок 2 – Эквивалентные схемы входов АЦП0

Необходимые регистры АЦП перечислены в таблице 1 и подробно описаны в приложении.

Таблица 1 – Регистры АЦП

AMX0P	Регистр выбора положительного канала AMUX0
AMX0N	Регистр выбора отрицательного канала AMUX0
ADC0CF	Регистр конфигурации АЦП0
ADC0H	Регистр старшего байта слова данных АЦП0
ADC0L	Регистр младшего байта слова данных АЦП0
ADC0CN	Регистр управления АЦП0

Организация памяти. Организация памяти МК с ядром CIP-51 соответствует стандарту 8051. Имеются две отдельные области памяти, память программ и память данных, которые разделяют одно и то же адресное пространство, но доступ к ним осуществляется командами различного типа. Организация памяти CIP-51 показана на рисунке 3.

Память программ. CIP-51 имеет адресное пространство памяти программ 64 Кбайт. В МК C8051F320/1 физически реализовано 16 Кбайт этой памяти программ, которая является внутрисистемной перепрограммируемой Flash-памятью, занимающей непрерывный блок адресов от 0x0000 до 0x3FFF. Адреса, превышающие 0x3FFF, зарезервированы. По умолчанию память программ настраивается только для чтения. Однако CIP-51 может записывать данные в память программ (с использованием команды MOVX), для чего необходимо установить в 1 бит разрешения записи памяти программ (PSCTL.0). Эта возможность позволяет CIP-51 обновлять программный код и использовать память программ для долговременного хранения данных.

Рисунок 3 – Карта распределения памяти

Память данных. Физически реализовано 256 байт внутреннего ОЗУ, отображенного в пространстве памяти данных с адресами от 0x00 до 0xFF. Младшие 128 байт памяти данных используются для регистров общего назначения (РОН) и сверхоперативного ЗУ (СОЗУ). Для доступа к младшим 128 байтам памяти данных можно использовать либо прямую, либо косвенную адресацию. Ячейки с адресами от 0x00 до 0x1F разбиты на четыре банка РОН, каждый банк состоит из восьми однобайтовых регистров. Следующие 16 байт (0x20 – 0x2F) могут адресоваться побайтно или побитно как 128 бит, доступные в режиме прямой битовой адресации. Старшие 128 байт памяти данных доступны только в режиме косвенной адресации. Эта область памяти занимает то же самое адресное пространство, что и регистры специального назначения (Special Function Registers – SFR), но физически отделена от них. При обращении к ячейкам памяти с адресами 0x7F – 0xFF использующийся в команде режим адресации определяет, к чему осуществляется доступ: к старшим 128 байтам памяти данных или к SFR. Команды, которые используют режим прямой адресации, будут обращаться к SFR. Команды, использующие режим косвенной адресации, будут обращаться к старшим 128 байтам памяти данных.

Регистры общего назначения. Младшие 32 байта памяти данных (0x00 – 0x1F) разбиты на четыре банка регистров общего назначения. Каждый банк состоит из восьми однобайтовых регистров, обозначаемых R0-R7. В конкретный момент времени может быть активен лишь один банк, определяемый битами RS0 (PSW.3) и RS1 (PSW.4) в слове состояния программы (program status word) PSW. Это позволяет осуществлять быстрое переключение контекста при вызове подпрограмм и процедур обработки прерываний. Режимы косвенной адресации используют регистры R0 и R1 в качестве индексных регистров.

Ячейки памяти с битовой адресацией. Кроме прямого (побайтного) доступа к памяти данных 16 ячеек этой памяти с адресами 0x20 – 0x2F доступны так же, как 128 индивидуально адресуемых бит. Каждый бит имеет битовый адрес от 0x00 до 0x7F. Бит 0 байта 0x20 имеет битовый адрес 0x00, а бит 7 байта 0x20 имеет битовый адрес 0x07. Бит 7 байта 0x2F имеет битовый адрес 0x7F. Битовый доступ можно отличить от байтового доступа по типу используемой команды (операнды исходных данных и результата в первом случае являются битами, во втором – байтами).

Ассемблер MCS-51™ допускает альтернативную запись для режима битовой адресации в форме XX.B, где XX – адрес байта, а B – позиция бита внутри этого байта. Например, команда MOV C, 22h.3 присваивает значение бита 0x13 (бит 3 в ячейке с адресом 0x22) флагу переноса.

Стек. Программный стек может быть размещен в любом месте 256-байтной памяти данных. Область стека определяется с использованием указателя стека (Stack Pointer – SP, 0x81). SP будет указывать на последнюю использованную ячейку. Следующее значение, загружаемое в стек, размещается по адресу SP+1 и затем SP инкрементируется. При сбросе SP инициализируется значением 0x07. Поэтому первое значение, загружаемое в стек, размещается по адресу 0x08, которое также является первым регистром (R0) регистрового банка 1. Таким образом, если требуется использовать более одного банка регистров, SP следует инициализировать адресом ячейки ОЗУ, не используемой для хранения данных. Стек может иметь глубину до 256 байт.

Регистры специального назначения. Ячейки памяти данных с адресами от 0x80 до 0xFF, доступные в режиме прямой адресации, образуют регистры специального назначения (Special Function Registers – SFR). SFR позволяют управлять ресурсами ядра CIP-51 и периферийными модулями, а также осуществлять обмен данными с ними.

Регистры SFR доступны в любое время, когда для доступа к ячейкам памяти с адресами от 0x80 до 0xFF используется режим прямой адресации. SFR с адресами, оканчивающимися на 0x0 или 0x8 (т.е. P0, TCON, SCON, IE, и т.д.), адресуются как побайтно, так и побитно. Все другие SFR адресуются только побайтно. Незанятые адреса в области SFR зарезервированы для дальнейшего использования. Обращение к ячейкам из этой области даст неопределенный результат и должно быть исключено.

3 Приборы и оборудование

В данной лабораторной работе используется:

1 Комплект разработки для микроконтроллера C8051F320.

2 Персональный компьютер с USB портом.

3 Генератор.

4 Программное обеспечение.

4 Порядок выполнения работы

1 Запустить среду разработки программ для микроконтроллеров SiLabs.

2 Подготовить комплект отладки и ПК. Подключить плату к программатору, подключить программатор к компьютеру по интерфейсу USB. Подать питание на плату с помощью 9В блока питания. Установить связь МК с компьютером.

3 Включить программу main_TimerWork2.asm в проект.

4 Провести инициализацию микроконтроллера.

4.1 Инициализация внутреннего кварцевого резонатора (регистр OSCICN).

4.2 Инициализация порта (регистры XBR0, XBR1, PnMDIN, PnMDOUT, PnSKIP) для подключения светодиода и кнопки.

4.3 Инициализация таймеров 0 и 1 (регистры TCON, TMOD) в режимах 1 и 2.

4.4 Инициализация прерывания (регистры IE, IP) от таймера/счетчика.

4.5 Инициализация АЦП (регистры AMX0P, AMX0N, ADC0CN, ADC0CF)

5 Установить значения в таймеры для генерации сигнала необходимой частоты (регистры TH0, TL0).

6 Запустить работу программы (кнопка Run среды).

7 Разработать алгоритм управления кнопкой (в соответствии с заданием).

8 Написать программу управления кнопкой, включающую подпрограмму гашения дребезга контактов.

9 Скомпилировать программу.

10 Выполнить построение проекта.

11 Записать программу во внутреннюю память МК.

12 Запустить работу программы (кнопка Run среды SiLabs).

13 Подать на АЦП сигнал с потенциометра.

14 Выполнить аналого-цифровое преобразование сигнала с частотой дискретизации 5 Гц.

15 Изменять частоту мерцания светодиода в зависимости от значения напряжения на выводе потенциометра (значения старшего регистра АЦП см. в таблице 2, частоту мерцания см. в таблице 1, лабораторной работы №2).

Таблица 2 – Варианты заданий

Вариант, №	1	2	3	4	5	6	7
Частота, Гц	F0	A5	EF	1F	7A	BC	3

5 Содержание отчета

1 Цель работы.

2 Алгоритм программы управления АЦП.

3 Листинг программы управления АЦП.

4 Выводы.

6 Контрольные вопросы

1 Перечислите основные составляющие модуля АЦП микроконтроллера C8051F320.

2 В каких режимах функционирует АЦП, в чем их отличие?

3 Как осуществляется настройка входов АЦП и какие регистры при этом используются?

4 Как может быть инициировано преобразование, назовите основные особенности каждого способа.

5 Какая структура памяти микроконтроллера? Назовите основные регистры, необходимые для работы с памятью данных.

Лабораторная работа №4

УСТАНОВКА СВЯЗИ МК С КОМПЬЮТЕРОМ

1 Цель работы

Получить практические навыки установки связи между ПК и МК посредством порта UART. Определить скорость передачи данных.

2 Теоретические сведения

UART0 представляет собой асинхронный полнодуплексный последовательный порт, способный работать в режимах 1 и 3 стандартного (для архитектуры 8051) UART. Поддержка усовершенствованного режима генерации скорости передачи данных позволяет использовать для генерации стандартных скоростей обмена различные источники тактирования. Буферизация принимаемых данных позволяет UART0 начать прием второго входящего байта данных до того, как программа закончит чтение предыдущего байта данных.

С работой UART0 (рисунок 1) связаны следующие регистры специального назначения: регистр управления UART0 (SCON0) и буфер данных UART0 (SBUF0). Одна и та же ячейка памяти, адресуемая как SBUF0, обеспечивает доступ и к регистру передатчика, и к регистру приемника. Операции записи в SBUF0 всегда обращаются к регистру передатчика**.** Операции чтения из SBUF0 всегда обращаются к буферизованному регистру приемника; невозможно прочитать данные из регистра передатчика.

Если прерывания от модуля UART0 разрешены, то запрос прерывания генерируется каждый раз при завершении передачи байта данных (установка в 1 флага TI0 в регистре SCON0) или при получении байта данных (установка в 1 флага RI0 в регистре SCON0). Флаги прерываний от UART0 не сбрасываются аппаратно при переходе к процедуре обслуживания прерывания. Они должны сбрасываться программно. Это позволяет программе определить причину, вызвавшую прерывание от UART0 (завершение передачи или завершение приема).

2.1 Усовершенствованный режим генерации скорости передачи данных

Скорость передачи данных UART0 генерируется Таймером 1, работающим в 8-разрядном режиме с автоперезагрузкой. Частота передатчика (TX) определяется переполнением регистра TL1; частота приемника определяется переполнением регистра-копии регистра TL1 (обозначенного как «RX-Таймер» на рисунке 2), который недоступен из программы пользователя. Скорость передачи данных передатчика и приемника равна деленной на два частоте переполнения регистров TL1 и RX-Таймер соответственно. RX-Таймер работает тогда, когда включен Таймер 1 и использует то же самое значение перезагрузки (TH1). Однако перезагрузка регистра RX-Таймер происходит в тот момент, ко-

гда на выводе RX обнаруживается событие START. Это позволяет начать прием данных в любой момент при обнаружении события START независимо от состояния Таймера TX.

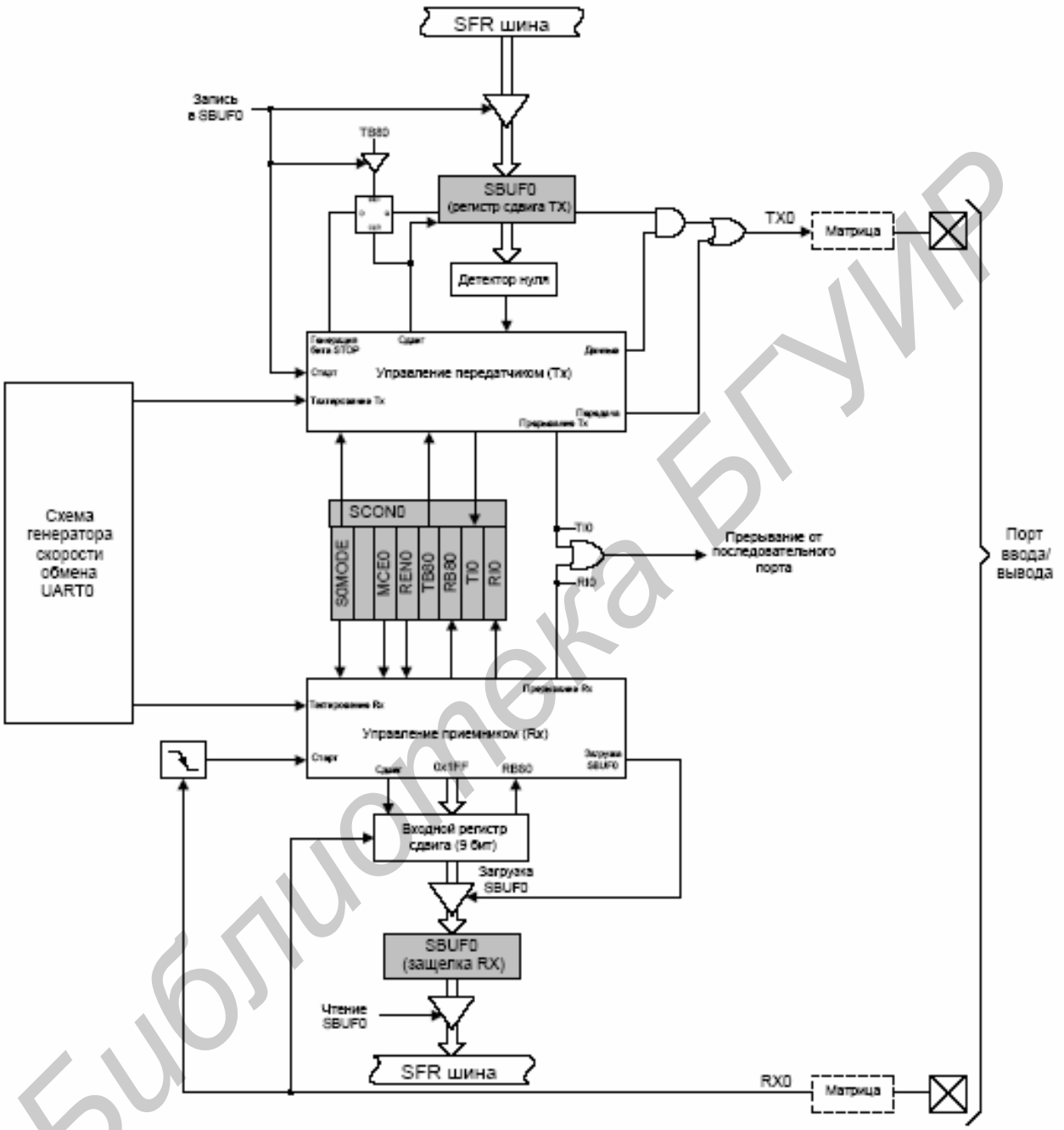

Рисунок 1 – Структурная схема UART0

Таймер 1 следует настроить для работы в режиме 2, т. е. как 8-разрядный таймер с автоперезагрузкой. Значение перезагрузки Таймера 1 следует установить таким образом, чтобы частота переполнений таймера была в два раза больше необходимой скорости передачи данных. Частота тактового сигнала Таймера 1 может быть одной из следующих:

1) SYSCLK;
2) SYSCLK/4;
3) SYSCLK/12;
4) SYSCLK/48;
5) частота внешнего генератора / 8;
6) частота внешнего сигнала на входе T1.

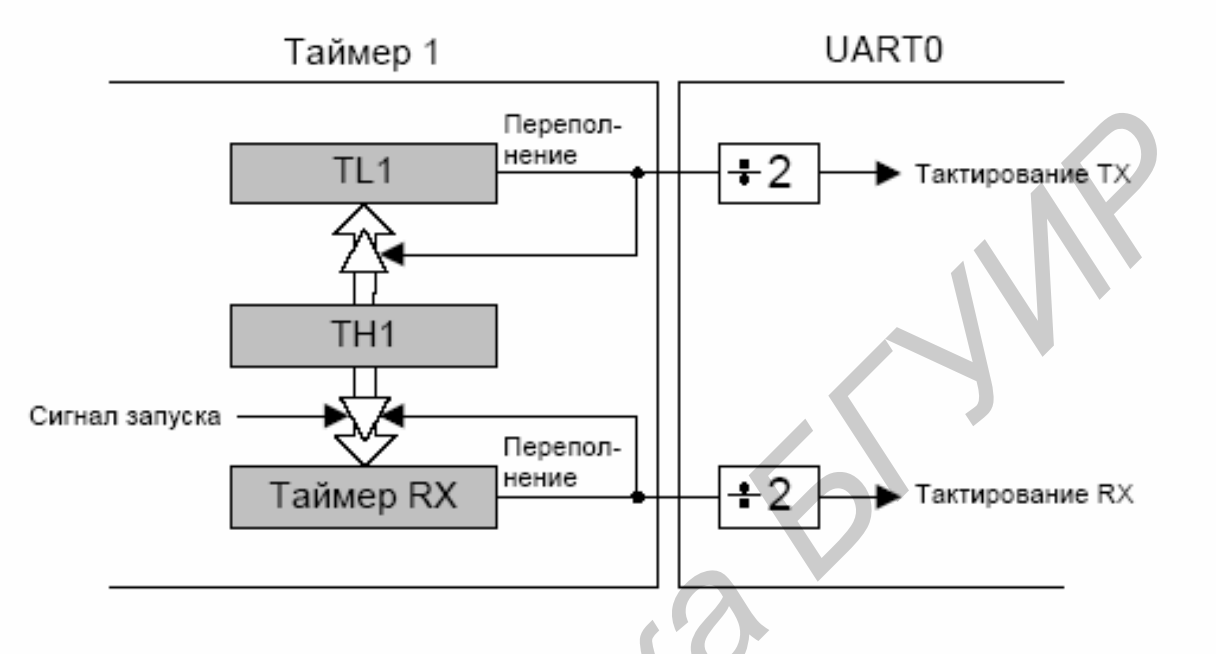

Рисунок 2 – Логика генератора скорости передачи данных UART0

Скорость передачи данных UART0 определяется из равенства

$$V_{UART0} = \frac{F_{T1}}{2}, \quad (1)$$

где V_{UART0} – скорость передачи данных UART0; F_{T1} – частота переполнения Таймера 1.

Расчет частоты переполнения Таймера 1 F_{T1} производим по формуле

$$F_{T1} = \frac{CLK_{T1}}{256 - H_{T1}}, \quad (2)$$

где CLK_{T1} – частота тактирования Таймера 1; H_{T1} – старший байт Таймера 1 (8-разрядное значение перезагрузки).

Следует отметить, что внутренний генератор может генерировать системный тактовый сигнал, в то время как выходной сигнал внешнего генератора подается на Таймер 1.

2.2 Режимы работы UART0

UART0 обеспечивает стандартный асинхронный полнодуплексный обмен данными. Режим работы UART0 (8- или 9-разрядный) выбирается при помощи бита S0MODE (SCON0.7). Типичные варианты использования UART приведены на рисунке 3.

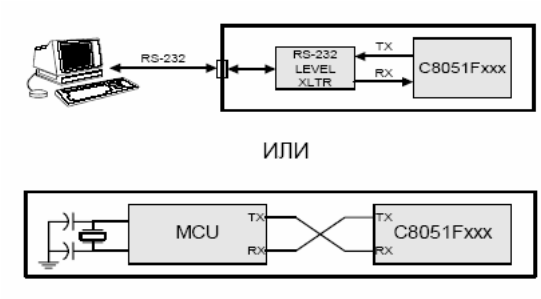

Рисунок 3 – Примеры использования UART

2.2.1 8-разрядный UART

В режиме 8-разрядного UART для передачи одного байта данных используются 10 бит: один стартовый бит, восемь бит данных (МЗР вперед) и один стоповый бит. Данные передаются МЗР вперед через внешний вывод TX0 и принимаются через внешний вывод RX0. При приеме в регистре SBUF0 сохраняются 8 бит данных, а бит RB80 (SCON0.2) принимает значение стопового бита.

Передача данных начинается, когда происходит запись байта данных в регистр SBUF0. Флаг прерывания от передатчика TI0 (SCON0.1) устанавливается в 1 в конце передачи (в начале передачи стопового бита). Прием данных может быть начат в любое время после установки в 1 флага включения приемника REN0 (SCON0.4). После приема стопового бита байт данных будет загружен в регистр приемника SBUF0, если соблюдаются следующие условия: RI0 должен быть равен лог.0, и если MCE0 = 1, то стоповый бит должен быть равен лог.1. В случае переполнения данных при приеме первые принятые 8 бит защелкиваются в регистре приемника SBUF0, а следующие данные, вызвавшие переполнение, теряются.

Если эти условия соблюдаются, то восемь бит данных сохраняются в регистре SBUF0, стоповый бит сохраняется в бите RB80 и устанавливается в 1 флаг RI0. Если эти условия не соблюдаются, то SBUF0 и RB80 не будут загружаться и флаг RI0 не устанавливается. При установке флагов TI0 или RI0 будет сгенерировано прерывание, если оно разрешено.

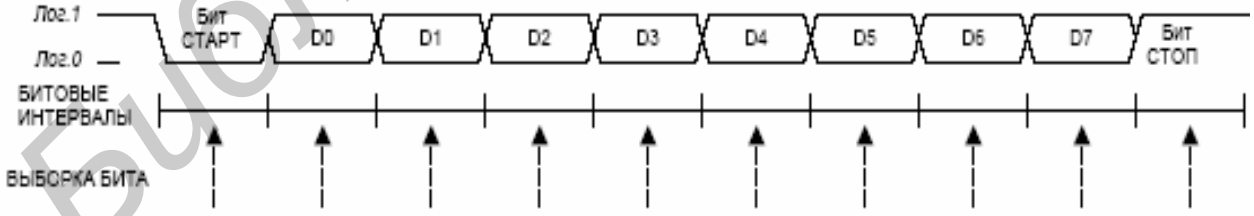

Рисунок 4 – Временные диаграммы в режиме 8-разрядного UART

2.2.2 9-разрядный UART

В режиме 9-разрядного UART для передачи одного байта данных используются 11 бит: один стартовый бит, восемь бит данных (МЗР вперед), программируемый девятый бит данных и один стоповый бит. При передаче значение девятого бита данных определяется значением бита TB80 (SCON0.3), который

устанавливается/сбрасывается программой пользователя. Значение девятого бита может либо соответствовать значению флага четности «P» регистра PSW (применяется для обнаружения ошибок), либо использоваться для организации связи с несколькими МК. При приеме значение девятого бита сохраняется в бите RB80 (SCON0.2), а стоповый бит игнорируется.

Передача данных начинается, когда происходит запись байта данных в регистр SBUF0. Флаг прерывания от передатчика TI0 (SCON0.1) устанавливается в 1 в конце передачи (в начале передачи стопового бита). Прием данных может быть начат в любое время после установки в 1 флага включения приемника REN0 (SCON0.4). После приема стопового бита байт данных будет загружен в регистр приемника SBUF0, если соблюдаются следующие условия: RI0 должен быть равен лог. 0, и, если MCE0 = 1, то стоповый бит должен быть равен лог.1 (когда MCE0 = 0, состояние девятого бита данных не имеет значения).

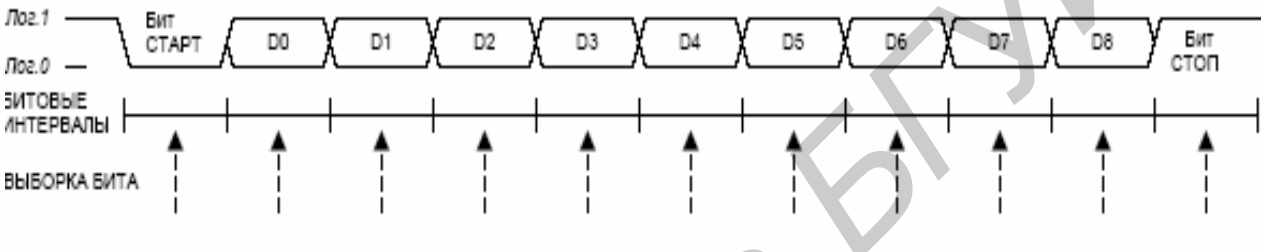

Рисунок 5 – Временные диаграммы в режиме 9-разрядного UART

Если эти условия соблюдаются, то 8 бит данных сохраняются в регистре SBUF0, девятый бит данных сохраняется в бите RB80 и устанавливается в 1 флаг RI0. Если эти условия не соблюдаются, то SBUF0 и RB80 не будут загружаться и флаг RI0 не будет устанавливаться. При установке флагов TI0 или RI0 будет сгенерировано прерывание от модуля UART0, если оно разрешено.

Таблица 1 – Параметры настройки таймера для стандартных скоростей передачи данных при тактировании от внутреннего генератора (11,0592 МГц)

	Частота: 11,0592 МГц						
	Требуемая скорость передачи данных (бит/сек)	Погрешность установки скорости передачи данных	Коэффициент деления генератора	Частота сигнала тактирования	SCA1-SCA0 (выбор коэффициента предварительного деления)*	T1M*	Значение перезагрузки Таймера 1
SYSCLK от внешнего генератора	230400	0,00%	48	SYSCLK	XX	1	0xE8
	115200	0,00%	96	SYSCLK	XX	1	0xD0
	57600	0,00%	192	SYSCLK	XX	1	0xA0
	28800	0,00%	384	SYSCLK	XX	1	0x40
	14400	0,00%	768	SYSCLK/12	00	0	0xE0
	9600	0,00%	1152	SYSCLK/12	00	0	0xD0
	2400	0,00%	4608	SYSCLK/12	00	0	0x40
	1200	0,00%	9216	SYSCLK/48	10	0	0xA0
SYSCLK от внутреннего генератора	230400	0,00%	48	EXTCLK/8	11	0	0xFD
	115200	0,00%	96	EXTCLK/8	11	0	0xFA
	57600	0,00%	192	EXTCLK/8	11	0	0xF4
	28800	0,00%	384	EXTCLK/8	11	0	0xE8
	14400	0,00%	768	EXTCLK/8	11	0	0xD0
	9600	0,00%	1152	EXTCLK/8	11	0	0xB8

Примечание – X – Не имеет значения

3 Приборы и оборудование

В данной лабораторной работе используется:

1 Комплект разработки для микроконтроллера C8051F320.

2 Персональный компьютер с USB портом.

3 Программное обеспечение.

4 Порядок выполнения работы

1 Запустить среду разработки программ для микроконтроллеров SiLabs.

2 Подготовить комплект отладки и ПК. Подключить плату к программатору, подключить программатор к компьютеру по интерфейсу USB. Подать питание на плату с помощью 9В блока питания. Установить связь МК с компьютером.

3 Включить программу main_TimerWork2.asm в проект.

4 Провести инициализацию микроконтроллера.

4.1 Инициализация внутреннего кварцевого резонатора (регистр OSCICN).

4.2 Инициализация порта (регистры XBR0, XBR1, PnMDIN, PnMDOUT, PnSKIP) для подключения светодиода и кнопки.

4.3 Инициализация таймеров 0 и 1 (регистры TCON, TMOD) в режимах 1 и 2.

4.4 Инициализация прерывания (регистры IE, IP) от таймер/счетчика.

4.5 Инициализация АЦП (регистры AMX0P, AMX0N, ADC0CN, ADC0CF).

4.6 Инициализация UART (регистр SCON0).

5 Установить значения в таймеры для генерации сигнала необходимой частоты (регистры TH0, TL0).

6 Разработать алгоритм программы, включающий управление аналого-цифровым преобразованием (в соответствии с заданием), сохранение результатов в памяти микроконтроллера, передачу данных с помощью универсального асинхронного приемопередатчика в ПК.

7 Написать программу, реализующую разработанный алгоритм.

8 Скомпилировать программу.

9 Выполнить построение проекта.

10 Записать программу во внутреннюю память МК.

11 Запустить работу программы (кнопка Run среды SiLabs).

12 Подать на АЦП сигнал частотой 1 Гц.

13 Выполнить аналого-цифровое преобразование сигнала с частотой дискретизации 10 Гц и сохранить 25 байт данных в памяти МК.

14 Передать полученные данные на ПК.

5 Содержание отчета

1 Цель работы.
2 Листинг программы
3 Выводы.

6 Контрольные вопросы

1 Определите назначение UART.
2 Назовите основные регистры, используемые портом UART.
3 Дайте описание бит регистра управления UART0 (SCON0).
4 Назначение генератора скорости передачи данных в UART.
5 Назовите основные параметры, определяющие скорость передачи данных.
6 Назовите режимы работы UART и их особенности.

ЛИТЕРАТУРА

1 Гладштейн, М. А. Микроконтроллеры смешанного сигнала C8051Fxxx фирмы Silicon Laboratories и их применение / М. А. Гладштейн. – Додэка XXI; серия : Мировая электроника. – 2008.

ПРИЛОЖЕНИЕ А

ПРИМЕР ЛИСТИНГА ПРОГРАММЫ ДЛЯ МИКРОКОНТРОЛЛЕРА CYGNAL СЕМЕЙСТВА C8051F320

```
; Рассмотрена работа со сторожевым таймером (WDT), таймером 0
;(Timer0),
; тактовым генератором (Oscillator) прерываниями (Interrupts), портами
;(P0.4)
$include (C8051F320.inc)          ; Include regsiter definition file.
;************************ define interrupts vector ****************
d_ResetVect       equ       000h
d_Timer0Vect      equ       00bh
d_EndVectors      equ       0b3h      ; End of Interrupt Vector space
;********************** my define *******************************
d_Timer0          equ       0bdch  ;!!!!   ; частота 2 Гц (деление на 48)
d_Timer1          equ       0fb1eh  ;!!!!   ; частота 100 Гц (деление на 48)
;************************** define data ***************************
CntTime     data      030h     ; счет времени между двумя
                               ; прерываниями INT0

;************************** define pins ***************************
pLED              bit       P2.2
pButton           bit       P2.0
;************************ programm code ***************************
      org d_ResetVect
      ljmp FnMain

      org d_Timer0Vect
      ljmp IntTimer0

      org d_EndVectors
;********************* Процедуры ******************************
;Вход R0 - время задержки в десятках миллисекунд
TimeWeit:
lb_TmrW_TimeCount:
    mov TL1,#LOW(d_Timer1)     ; длительность 1/100
    mov TH1,#HIGH(d_Timer1)
     clr TF1
    setb TR1
lb_TmrW_T1Work:
   nop
   jnb TF1,lb_TmrW_T1Work
   djnz R0,lb_TmrW_TimeCount
```

```
   ret
;************ Процедуры обработки прерываний ****************
IntTimer0:
   push PSW
   push ACC
   setb RS0                    ; устанавливаем банк регистров №1
  clr TR0
  mov TL0,#LOW(d_Timer0)      ; длительность 1/2
  mov TH0,#HIGH(d_Timer0)
  setb TR0
   ;*** Рабочий цикл ***

    cpl pLED

   ;*** Рабочий цикл ***
lb_iT0_end:
   pop ACC
   pop PSW
   reti

;********************** инициализация ************************
Init8051:
   lcall PCA_Init            ; инициализация WDT
   lcall Timer_Init          ; инициализация Таймера 0
   lcall Oscillator_Init      ; инициализация внутреннего генератора
   lcall Interrupts_Init     ; инициализация прерываний
   lcall Port_IO_Init
   lcall RegInit       ; Инициализация регистров
   ret

PCA_Init:
   anl  PCA0MD,    #0BFh      ; отключаем WDT
   mov  PCA0MD,    #000h
   ret

 Timer_Init:
   mov  TMOD,      #011h      ; Работаем с T0 режим 1; T1 режим 1
   mov  CKCON,     #002h      ; Используем делитель. Делим системную ;
                               частоту на 48
  mov TL0,#LOW(d_Timer0)      ; длительность 1/2с
  mov TH0,#HIGH(d_Timer0)
  mov TL1,#LOW(d_Timer1)      ; длительность 1/100с
  mov TH1,#HIGH(d_Timer1)
  setb TR0
```

```
   ret

Oscillator_Init:
   mov  OSCICN,    #082h       ; Используем внутренний тактовый
                               ; генератор
   ret

Interrupts_Init:
   mov  IE,        #082h       ; Разрешаем общие прерывания и прерывания
                               ; от Таймера 0
   ret

Port_IO_Init:
   mov  P2MDOUT,   #006h       ; Включаем подтяжку
   mov  XBR1, #040h ; Включаем Crossbar
   ret

;*************** ГОЛОВНАЯ ПРОГРАММА **********************
FnMain:
   mov SP,#010h            ; Инициализируем стек
   lcall Init8051

lb_main_Loop:
      nop
     jb pButton,lb_main_Loop

     mov R0,#30
     lcall TimeWeit
     cpl TR0

lb_main_WeitOffButt:
      nop
     jnb pButton,lb_main_WeitOffButt
     mov R0,#30
     lcall TimeWeit
   sjmp lb_main_Loop
end
```

ПРИЛОЖЕНИЕ Б

РЕГИСТРЫ СПЕЦИАЛЬНОГО НАЗНАЧЕНИЯ (SPECIAL FUNCTION REGISTERS – SFR)

PSW: Слово состояния программы

R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
CY	AC	F0	RS1	RS0	OV	F1	PAR-ITY
Бит 7	Бит 6	Бит 5	Бит 4	Бит 3	Бит 2	Бит 1	Бит 0 (доступен в битовом режиме адресации)

Значение при сбросе: 00000000
SFR Адрес: 0xD0

Бит 7: **CY**: Флаг переноса.

Этот бит устанавливается, если в результате последней арифметической операции произошел перенос (сложение) или заем (вычитание). Он сбрасывается в 0 всеми другими арифметическими операциями.

Бит 6: **AC**: Флаг десятичного переноса.

Этот бит устанавливается, если в результате последней арифметической операции произошел перенос (сложение) в старший полубайт или заем (вычитание) из старшего полубайта. Он сбрасывается в 0 всеми другими арифметическими операциями.

Бит 5: **F0**: Флаг пользователя 0.

Это доступный в битовом режиме адресации флаг общего назначения, предназначенный для использования под управлением программы.

Биты 4-3: **RS1-RS0**: Биты выбора банка регистров.

Эти биты определяют активный банк регистров.

Бит 2: **OV**: Флаг переполнения.

Этот бит устанавливается в 1 в следующих случаях:

– если в результате выполнения команд ADD, ADDC или SUBB произошло переполнение с изменением знака;

– если в результате выполнения команды MUL произошло переполнение (результат превышает значение 255);

– если при выполнении команды DIV произошло деление на ноль.

Бит OV сбрасывается в 0 командами ADD, ADDC, SUBB, MUL и DIV во всех других случаях.

Бит 1: **F1**: Флаг пользователя 1.

Это доступный в битовом режиме адресации флаг общего назначения, предназначенный для использования под управлением программы.

Бит 0: **PARITY**: Флаг четности.

Этот бит устанавливается в 1, если сумма восьми бит в аккумуляторе нечетная, и сбрасывается, если сумма четная.

TMOD: Регистр режима Таймеров 0 и 1

R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Значение при сбросе: 00000000 SFR Адрес: 0x89
GATE 1	C/T1	T1M1	T1M0	GATE 0	C/T0	T0M1	T0M0	
Бит 7	Бит 6	Бит 5	Бит 4	Бит 3	Бит 2	Бит 1	Бит 0	

Бит 7: **GATE1**: Управление блокировкой Таймера 1.

0: Таймер 1 включен, если TR1 = 1, независимо от логического уровня на входе /INT1.

1: Таймер 1 включен только тогда, когда TR1 = 1 и на входе /INT1 активный логический уровень определяется битом IN1PL в регистре INT01CF.

Бит 6: **C/T1**: Выбор режима таймера или счетчика для T/C1.

0: T/C1 работает как таймер: Таймер 1 инкрементируется от внутреннего сигнала тактирования, который задается битом T1M (CKCON.4).

1: T/C1 работает как счетчик: Таймер 1 инкрементируется под воздействием перехода из 1 в 0 внешнего входного сигнала (T1).

Биты 5-4: **T1M1-T1M0**: Выбор режима работы Таймера 1.

Эти биты определяют режим работы Таймера 1.

T1M1	T1M0	Режим
0	0	Режим 0: 13-разрядный таймер/счетчик
0	1	Режим 1: 16-разрядный таймер/счетчик
1	0	Режим 2: 8-разрядный таймер/счетчик с автоперезагрузкой
1	1	Режим 3: Таймер 1 неактивен

Бит 3: **GATE0**: Управление блокировкой Таймера 0.

0: Таймер 0 включен, если TR0 = 1, независимо от логического уровня на входе /INT0.

1: Таймер 0 включен только тогда, когда TR0 = 1 и на входе /INT0 активный логический уровень определяется битом IN0PL в регистре INT01CF.

Бит 2: **C/T0**: Выбор режима таймера или счетчика для T/C0.

0: T/C0 работает как таймер: Таймер 0 инкрементируется от внутреннего сигнала тактирования, который задается битом T0M (CKCON.3).

1: T/C0 работает как счетчик: Таймер 0 инкрементируется под воздействием перехода из 1 в 0 внешнего входного сигнала (T0).

Биты 1-0: **T0M1-T0M0**: Выбор режима работы Таймера 0.

Эти биты определяют режим работы Таймера 0.

T0M1	T0M0	Режим
0	0	Режим 0: 13-разрядный таймер/счетчик
0	1	Режим 1: 16-разрядный таймер/счетчик
1	0	Режим 2: 8-разрядный таймер/счетчик с автоперезагрузкой
1	1	Режим 3: Два 8-разрядных таймера/счетчика

ПРИЛОЖЕНИЕ В

ОСНОВНЫЕ КОМАНДЫ МИКРОКОНТРОЛЛЕРА

Мнемоника команды	Описание	Байты	Циклы
1	2	3	4
АРИФМЕТИЧЕСКИЕ ОПЕРАЦИИ			
ADD A,Rn	Сложение аккумулятора с регистром (n = 0…7)	1	1
ADD A,direct	Сложение аккумулятора с прямо адресуемым байтом	2	2
ADD A,@Ri	Сложение аккумулятора с косвенно адресуемым байтом ОЗУ	1	2
ADD A,#data	Сложение аккумулятора с константой	2	2
ADDC A,Rn	Сложение аккумулятора с регистром и переносом	1	1
ADDC A,direct	Сложение аккумулятора с прямо адресуемым байтом и переносом	2	2
ADDC A,@Ri	Сложение аккумулятора с косвенно адресуемым байтом ОЗУ и переносом	1	2
ADDC A,#data	Сложение аккумулятора с константой и переносом	2	2
SUBB A,Rn	Вычитание из аккумулятора регистра и заёма	1	1
SUBB A,direct	Вычитание из аккумулятора прямо адресуемого байта и заёма	2	2
SUBB A,@Ri	Вычитание из аккумулятора косвенно адресуемого байта ОЗУ и заёма	1	2
SUBB A,#data	Вычитание из аккумулятора константы и заёма	2	2
INC A	Инкремент аккумулятора	1	1
INC Rn	Инкремент регистра	1	1
INC direct	Инкремент прямо адресуемого байта	2	2
INC @Ri	Инкремент косвенно адресуемого байта ОЗУ	1	2
DEC A	Декремент аккумулятора	1	1
DEC Rn	Декремент регистра	1	1
DEC direct	Декремент прямо адресуемого байта	2	2
DEC @Ri	Декремент косвенно адресуемого байта ОЗУ	1	2
INC DPTR	Инкремент указателя данных	1	1
MUL AB	Умножение аккумулятора на регистр B	1	4
DIV AB	Деление аккумулятора на регистр B	1	8
DA A	Десятичная коррекция аккумулятора	1	1
ЛОГИЧЕСКИЕ ОПЕРАЦИИ			
ANL A,Rn	Логическое И аккумулятора и регистра	1	1
ANL A,direct	Логическое И аккумулятора и прямо адресуемого байта	2	2
ANL A,@Ri	Логическое И аккумулятора и косвенно адресуемого байта ОЗУ	1	2
ANL A,#data	Логическое И аккумулятора и константы	2	2
ANL direct,A	Логическое И прямо адресуемого байта и аккумулятора	2	2
ANL direct,#data	Логическое И прямо адресуемого байта и константы	3	3
ORL A,Rn	Логическое ИЛИ аккумулятора и регистра	1	1
ORL A,direct	Логическое ИЛИ аккумулятора и прямо адресуемого байта	2	2
ORL A,@Ri	Логическое ИЛИ аккумулятора и косвенно адресуемого байта ОЗУ	1	2
ORL A,#data	Логическое ИЛИ аккумулятора и константы	2	2
ORL direct,A	Логическое ИЛИ прямо адресуемого байта и аккумулятора	2	2
ORL direct,#data	Логическое ИЛИ прямо адресуемого байта и константы	3	3
XRL A,Rn	Исключающее ИЛИ аккумулятора и регистра	1	1
XRL A,direct	Исключающее ИЛИ аккумулятора и прямо адресуемого байта	2	2
XRL A,@Ri	Исключающее ИЛИ аккумулятора и косвенно адресуемого байта ОЗУ	1	2

1	2	3	4
XRL A,#data	Исключающее ИЛИ аккумулятора и константы	2	2
XRL direct,A	Исключающее ИЛИ прямо адресуемого байта и аккумулятора	2	2
XRL direct,#data	Исключающее ИЛИ прямо адресуемого байта и константы	3	3
CLR A	Сброс аккумулятора	1	1
CPL A	Инверсия аккумулятора	1	1
RL A	Сдвиг аккумулятора влево циклический	1	1
RLC A	Сдвиг аккумулятора влево через перенос	1	1
RR A	Сдвиг аккумулятора вправо циклический	1	1
RRC A	Сдвиг аккумулятора вправо через перенос	1	1
SWAP A	Обмен местами тетрад в аккумуляторе	1	1
КОМАНДЫ ПЕРЕДАЧИ ДАННЫХ			
MOV A,Rn	Пересылка в аккумулятор из регистра (n = 0…7)	1	1
MOV A,direct	Пересылка в аккумулятор прямо адресуемого байта	2	2
MOV A,@Ri	Пересылка в аккумулятор косвенно адресуемого байта ОЗУ	1	2
MOV A,#data	Загрузка в аккумулятор константы	2	2
MOV Rn,A	Пересылка в регистр из аккумулятора	1	1
MOV Rn,direct	Пересылка в регистр прямо адресуемого байта	2	2
MOV Rn,#data	Загрузка в регистр константы	2	2
MOV direct,A	Пересылка по прямому адресу аккумулятора	2	2
MOV direct,Rn	Пересылка по прямому адресу регистра	2	2
MOV direct,direct	Пересылка прямо адресуемого байта по прямому адресу	3	3
MOV direct,@Ri	Пересылка косвенно адресуемого байта ОЗУ по прямому адресу	2	2
MOV direct,#data	Пересылка по прямому адресу константы	3	3
MOV @Ri,A	Пересылка в косвенно адресуемую ячейку ОЗУ аккумулятора	1	2
MOV @Ri,direct	Пересылка в косвенно адресуемую ячейку ОЗУ прямо адресуемого байта	2	2
MOV @Ri,#data	Пересылка в косвенно адресуемую ячейку ОЗУ константы	2	2
MOV DPTR,#data16	Загрузка указателя данных	3	3
MOVC A,@A+DPTR	Пересылка в аккумулятор байта из памяти программ	1	3
MOVC A,@A+PC	Пересылка в аккумулятор байта из памяти программ	1	3
MOVX A,@Ri	Пересылка в аккумулятор байта из внешней памяти данных	1	3
MOVX @Ri,A	Пересылка байта из аккумулятора во внешнюю память данных	1	3
MOVX A,@DPTR	Пересылка в аккумулятор из расширенной внешней памяти данных	1	3
MOVX @DPTR,A	Пересылка из аккумулятора в расширенную внешнюю память данных	1	3
PUSH direct	Загрузка в стек	2	2
POP direct	Извлечение из стека	2	2
XCH A,Rn	Обмен аккумулятора с регистром	1	1
XCH A,direct	Обмен аккумулятора с прямо адресуемым байтом	2	2
XCH A,@Ri	Обмен аккумулятора с косвенно адресуемым байтом ОЗУ	1	2
XCHD A,@Ri	Обмен младшей тетрады аккумулятора с младшей тетрадой Косвенно адресуемого байта ОЗУ	1	2
ОПЕРАЦИИ С БИТАМИ			
CLR C	Сброс переноса	1	1
CLR bit	Сброс бита	2	2
SETB C	Установка переноса	1	1
SETB bit	Установка бита	2	2

1	2	3	4
CPL C	Инверсия переноса	1	1
CPL bit	Инверсия бита	2	2
ANL C,bit	Логическое И бита и переноса	2	2
ANL C,/bit	Логическое И инверсии бита и переноса	2	2
ORL C,bit	Логическое ИЛИ бита и переноса	2	2
ORL C,/bit	Логическое ИЛИ инверсии бита и переноса	2	2
MOV C,bit	Пересылка бита в перенос	2	2
MOV bit,C	Пересылка переноса в бит	2	2
JC rel	Переход, если перенос равен единице	2	2/3
JNC rel	Переход, если перенос равен нулю	2	2/3
JB bit,rel	Переход, если бит равен единице	3	3/4
JNB bit,rel	Переход, если бит равен нулю	3	3/4
JBC bit,rel	Переход, если бит установлен, с последующим сбросом бита	3	3/4
ПРОГРАММНЫЕ ПЕРЕХОДЫ			
ACALL addr11	Абсолютный вызов подпрограммы в пределах страницы в 2 Кбайта	2	3
LCALL addr16	Длинный вызов подпрограммы	3	4
RET	Возврат из подпрограммы	1	5
RETI	Возврат из подпрограммы обработки прерывания	1	5
AJMP addr11	Абсолютный переход внутри страницы в 2 Кбайта	2	3
LJMP addr16	Длинный переход в полном объеме памяти программ	3	4
SJMP rel	Короткий относительный переход внутри страницы в 256 байт	2	3
JMP @A+DPTR	Косвенный относительный переход	1	3
JZ rel	Переход, если аккумулятор равен нулю	2	2/3
JNZ rel	Переход, если аккумулятор не равен нулю	2	2/3
CJNE A,direct,rel	Сравнение аккумулятора с прямо адресуемым байтом и переход, если они не равны	3	3/4
CJNE A,#data,rel	Сравнение аккумулятора с константой и переход, если они не равны	3	3/4
CJNE Rn,#data,rel	Сравнение регистра с константой и переход, если они не равны	3	3/4
CJNE @Ri,#data,rel	Сравнение косвенно адресуемого байта ОЗУ с константой и переход, если они не равны	3	4/5
DJNZ Rn,rel	Декремент регистра и переход, если не равен нулю	2	2/3
DJNZ direct,rel	Декремент прямо адресуемого байта и переход, если не равен нулю	3	3/4
NOP	Холостая команда	1	1

Условные обозначения:

Rn – Регистр R0-R7 выбранного банка регистров.

@Ri – Ячейка ОЗУ данных, косвенно адресуемая через регистры R0-R1.

rel – 8-битное смещение со знаком (в дополнительном коде) относительно первого байта следующей команды. Используется командой SJMP и всеми командами условных переходов.

direct – 8-битный адрес ячейки внутреннего ОЗУ данных. Это может быть ячейка ОЗУ данных прямого доступа (0x00–0x7F) или регистр специального назначения SFR (0x80–0xFF).

#data – 8-битная константа.

#data 16 – 16-битная константа.

bit – Прямо адресуемый бит ячейки ОЗУ данных или регистра специального назначения SFR.

Св. план 2009, поз. 13

Учебное издание

Дик Сергей Константинович
Давыдов Максим Викторович
Терех Александр Сергеевич

ПРОЕКТИРОВАНИЕ НА ОСНОВЕ МИКРОКОНТРОЛЛЕРОВ

ЛАБОРАТОРНЫЙ ПРАКТИКУМ

для студентов специальности «Медицинская электроника»
дневной и заочной форм обучения

Редактор Т. П. Андрейченко
Корректор Е. Н. Батурчик
Компьютерная верстка Е. С. Чайковская

Подписано в печать 01.07.2009. Формат 60х84 1/16. Бумага офсетная.
Гарнитура «Таймс». Печать ризографическая. Усл. печ. л. 2,79.
Уч.-изд. л. 2,5. Тираж 100 экз. Заказ 33.

Издатель и полиграфическое исполнение: Учреждение образования
«Белорусский государственный университет информатики и радиоэлектроники»
ЛИ №02330/0494371 от 16.03.2009. ЛП №02330/0494175 от 03.04.2009.
220013, Минск, П. Бровки, 6