Федеральное агентство по образованию
Государственное образовательное учреждение высшего профессионального образования
Ульяновский государственный технический университет

П. П. Редькин, А. Б. Виноградов

Микроконверторы фирмы Analog Devices в микропроцессорных приборных комплексах

Учебное пособие

Рекомендовано Учебно-методическим объединением по образованию в области радиотехники, электроники, биомедицинской техники и автоматизации в качестве учебного пособия для студентов, обучающихся по направлению 210200 «Проектирование и технология электронных средств» и по специальностям 210201 «Проектирование и технология радиоэлектронных средств» и 210202 «Проектирование и технология электронно-вычислительных средств»

Ульяновск 2005

УДК 681.3.06(075)
ББК 32.973.26-04я7
Р33

Рецензенты:
ОАО «Ульяновское конструкторское бюро приборостроения»
(канд. техн. наук Д. Л. Федоров);
д-р техн. наук, профессор В. Н. Негода.

Редькин П. П.

Р33 Микроконверторы фирмы Analog Devices в микропроцессорных приборных комплексах: учебное пособие / П. П. Редькин, А. Б. Виноградов. – Ульяновск: УлГТУ, 2005. – 316 с.

ISBN 5-89146-814-X

В пособии изложены основные сведения о микроконверторах фирмы Analog Devices, принципах их использования в микропроцессорных приборных комплексах, приводятся сведения решениях типовых узлов приборных комплексов.

Учебное пособие предназначено для студентов, обучающихся по направлению 210200 «Проектирование и технология электронных средств» и специальностям 210201 «Проектирование и технология радиоэлектронных средств» и 210202 «Проектирование и технология электронно-вычислительных средств».

УДК 681.3.06 (075)
ББК 32.973.26-04я7

ISBN 5-89146-814-X

СОДЕРЖАНИЕ

Список таблиц

Список рисунков

Список листингов программ

Введение

Книга представляет собой практическое руководство и справочное пособие для изучения микроконвертора ADuC824 и создания на его основе реальных проектов. Для полноценного понимания и усвоения материала пособия необходимо знать основы языка ассемблера Asm51, цифровой и аналоговой схемотехники.

Первая часть книги содержит довольно полную справочную информацию по ADuC824 [1]. Сюда входят:

– технические характеристики ADuC824;

– описание архитектуры и программной модели ADuC824;

– описание аппаратного построения и программирования всей периферии ADuC824 (каждый периферийный модуль описан в отдельном пункте);

– рекомендации производителя по применению ADuC824.

Вторая часть книги содержит обширную справочную информацию по фирменному программному пакету разработки-отладки приложений на основе ADuC824 [2]. Программные продукты этого пакета размещены для бесплатного скачивания на сайте компании Analog Devices.

Третья часть книги представляет собой описание алгоритмов и исходных тестов управляющих программ, демонстрирующих работу основных периферийных узлов МК ADuC824. Все программы написаны и отлажены. Каждая из них является полностью законченным продуктом и содержит помимо демонстрационных компонент отдельные программные блоки-драйверы периферийных узлов ADuC824, оформленные в виде подпрограмм и подробно документированные, что дает возможность применять их в реальных приложениях, содержащих ADuC824 и другие 51-совместимые микроконтроллеры.

Все рассмотренные в учебном пособии примеры программ и документация на электронные компоненты размещены на сервере Ульяновского государственного технического университета по адресу ftp://ftp.ustu/pub/edu/aduc.

Принятые сокращения

АЦП – аналого-цифровой преобразователь
ИОН – источник опорного напряжения
МЗР – младший значащий разряд
МК – микроконвертор
ОЗУ – оперативное запоминающее устройство
ФАПЧ – фазовая автоматическая подстройка частоты
ТС – таймер-счетчик
ЦАП – цифро-аналоговый преобразователь
ПО – программное обеспечение

1. Справочная информация о микроконверторах

1.1. Общее описание микроконвертора

МК ADuC824 является функционально законченным контроллером интеллектуального датчика, включающим в себя: два аппаратных модуля сигма-дельта АЦП высокого разрешения (24-разрядное и 16-разрядное), 8-разрядное микропроцессорное устройство управления и встроенную Flash-память программ и данных. Кроме двух независимых модулей АЦП в составе устройства имеется датчик температуры и прецизионный программируемый усилитель, что позволяет МК выполнять прямые измерения малых уровней напряжения. АЦП с встроенным цифровым фильтром предназначены для измерения низкочастотных сигналов в широком динамическом диапазоне. Частота выдачи результатов измерений с выходов АЦП программируется. Упрощенная структурная схема приведена на рис. 1.1. Спецификация характеристик приведена в табл. 1.1.

Таблица 1.1

Спецификация параметров ADuC824[1]

(AVDD=2,7–3,6 В или 4,75–5,25 В, DVDD=2,7–3,6 В или 4,75–5,25 В; REFIN(+)=2,5 В; REFIN(–)=AGND; AGND=DGND=0 В; XTAL1/XTAL2 = 32 768 Гц. Все параметры приводятся для температурного диапазона от TMIN до TMAX, если это не оговорено особо.)

Параметр	**Значение**	**Условия измерения**	**Единицы измерения**
Параметры АЦП			
Скорость преобразования	5,4 (min) 105 (max)	По обоим каналам программируется с шагом 0,732 с	[Гц] [Гц]
Модуль основного АЦП			
Преобразование без пропуска кодов[2]	24 (min)	Частота преобразования 20 Гц	[разряды]
Разрешение	13 (P–P) 18 (P–P)	Частота преобразования 20 Гц, диапазон ±20 мВ Частота преобразования 20 Гц, диапазон ±2,56 В	[разряды] [разряды]
Интегральная нелинейность	±15 (max)		[ppm от FSR]
Ошибка смещения[3]	±3 (тип)		[мкВ]
Дрейф смещения	±10 (тип)		[нВ/°C]
Ошибка полной шкалы[4]	±10 (тип)		[мкВ]
Дрейф усиления[5]	±0,5 (тип)		[ppm/°C]
Согласование диапазонов АЦП	±2 (тип)	AIN=18 мВ	[мкВ]
Ослабление влияния напряжения питания (PSR)	113 (min) 80 (min)	AIN=7,8 мВ, диапазон ±20 мВ AIN=1 В, диапазон ±2,56 В	[дБ] [дБ]

Продолжение табл. 1.1

Параметр	Значение	Условия измерения	Единицы измерения
Ослабление синфазного сигнала			
на AIN	95 (min)	DC, AIN=7,8 мВ, диапазон ±20 мВ	[дБ]
на AIN	113 (min)	DC, AIN=1 В, диапазон ±2,56 В	[дБ]
на REFIN	125 (min)	DC, AIN=1 В, диапазон ±2,56 В	[дБ]
Ослабление синфазного сигнала на частоте 50 Гц/60 Гц[2]		Частота преобразования 20 Гц	
на AIN	95 (min)	50 Гц/60 Гц ±1 Гц, AIN=7,8 мВ, диапазон ±20 мВ	[дБ]
на AIN	90 (min)	50 Гц/60 Гц ±1 Гц, AIN=1 В, диапазон ±2,56 В	[дБ]
на REFIN	90 (min)	50 Гц/60 Гц ±1 Гц, AIN=1 В, диапазон ±2,56 В	[дБ]
Ослабление противофазного сигнала на частоте 50 Гц/60 Гц[2]			
на AIN	60 (min)	50 Гц/60 Гц ±1 Гц, частота преобразования 20 Гц	[дБ]
на REFIN	60 (min)	50 Гц/60 Гц ±1 Гц, частота преобразования 20 Гц	[дБ]
Модуль дополнительного АЦП			
Преобразование без пропуска кодов[2]	16 (min)		[разряды]
Разрешение	16 (P-P) (тип)	Диапазон ±2,5 В, частота преобразования 20 Гц	[разряды]
Интегральная нелинейность	±15 (max)		[ppm от FSR]
Ошибка смещения[3]	– 2 (тип)		[МЗР]
Дрейф смещения	1 (тип)		[мкВ/°С]
Ошибка полной шкалы[6]	– 2,5 (тип)		[МЗР]
Дрейф усиления[5]	±0,5 (тип)		[ppm/°С]
Ослабление влияния напряжения питания (PSR)	80 (min)	AIN=1 В, частота преобразования 20 Гц	[дБ]
Параметры ЦАП			
По постоянному току[7]:			
Разрешение	12		[разряды]
Относительная точность	±3 (тип)		[МЗР]
Дифференциальная нелинейность	–1 (max)	Гарантируется монотонность 12 разрядов	[МЗР]
Ошибка смещения	±50 (max)		[мВ]
Ошибка усиления	±1 (max)	Диапазон AV_{DD}	[%]
По переменному току[2,7]:	±1 (тип)	Диапазон V_{REF}	[%]

Продолжение табл. 1.1

Параметр	Значение	Условия измерения	Единицы измерения
Время установления вых. напряжения Импульсная энергия, передаваемая из цифровой части в аналоговую часть	15 (тип) 10 (тип)	Время установления до 1 МЗР от конечной величины Изменение на 1 МЗР с переносом в старший разряд	[мкс] [нВ×с]
Внутренний ИОН			
ИОН АЦП			
Опорное напряжение	1,25 ±1 % (min/max)	Начальный допуск при +25°С, V_{DD}=5 В	[В]
Ослабление влияния напряжения питания	45 (тип)		[дБ]
Температурный коэффициент	100 (тип)		[ppm/°С]
ИОН ЦАП			
Опорное напряжение	2,5 ±1 % (min/ max)	Начальный допуск при +25°С, V_{DD}=5 В	[В]
Ослабление влияния напряжения питания	50 (тип)		[дБ]
Температурный коэффициент	±100 (тип)		[ppm/°С]
Аналоговые входы/входы ИОН			
Основной АЦП			
Диапазон входных дифференциальных напряжений[9,10] Биполярный режим (ADC0CON3=0)	 ±20 ±40 ±80 ±160 ±320 ±640 ±1,28 ±2,56	Внешний опорный источник = 2,5 В RN2, RN1, RN0 в ADC0CON установлены как: 000 (однополярный режим 0 – 20 мВ) 001 (однополярный режим 0 – 40 мВ) 010 (однополярный режим 0 – 80 мВ) 011 (однополярный режим 0 – 160 мВ) 100 (однополярный режим 0 – 320 мВ) 101 (однополярный режим 0 – 640 мВ) 110 (однополярный режим 0 – 1,28 В) 111 (однополярный режим 0 – 2,56 В)	 [мВ] [мВ] [мВ] [мВ] [мВ] [мВ] [В] [В]
Ток аналогового входа[2]	±1 (max)		[нА]
Дрейф выходного тока	±5 (тип)		[пА/°С]
Абсолютные пределы входных напряжений	AGND+100 мВ (min) AV_{DD} –100 мВ (max)		[В] [В]
Дополнительный АЦП			
Униполярный режим, для биполярного режима см. [11]	[В]		
Диапазон входных сигналов[9,10]	0 – V_{REF}		

Продолжение табл. 1.1

Параметр	Значение	Условия измерения	Единицы измерения
Средний ток аналогового входа	125 (тип)	Входной ток небуферизованного входа дополнительного АЦП будет меняться с изменением входного напряжения	[нА/В]
Дрейф среднего входного тока[2]	±2 (тип)		[пА/В/°С]
Абсолютные пределы входных напряжений[11]	AGND-30 мВ (min) AVDD +30 мВ (max)		[В] [В]
Входы внешнего ИОН			
Диапазон от REFIN(+) до REFIN(–)[2]	1 (min) AVDD (max)		[В] [В]
Средний ток аналогового входа	1 (тип)	Оба модуля АЦП разрешены	[мкА/В]
Дрейф среднего входного тока	±0,1 (тип)		[нА/В/°С]
Напряжение срабатывания «отсутствие внешнего ИОН»	0,3 (min) 0,65 (max)	NOXREF бит активен, если VREF<0,3 В NOXREF бит пассивен, если VREF>0,65 В	[В] [В]
Системная калибровка АЦП			
Предел калибровки полной шкалы	+1,05 × FS (max)		[В]
Предел калибровки нуля	–1,05 × FS (min)		[В]
Входной диапазон	0,8 × FS (min) 2,1 × FS (max)		[В] [В]
Аналоговые выходы ЦАП			
Диапазон по напряжению	0–VREF (тип) 0–AVDD (тип)	DACRN=0 в регистре DACCON DACRN=1 в регистре DACCON	[В] [В]
Величина резистивной нагрузки	10 (тип)	Между выходом ЦАП и AGND	[кОм]
Величина емкостной нагрузки	100 (тип)	Между выходом ЦАП и AGND	[пФ]
Выходной импенданс	0,5		[Ом]
Втекающий ток (ISINK)	50 (тип)		[мкА]
Датчик температуры			
Точность	±2 (тип)		[°С]
Температурное сопротивление	90		[°С/Вт]
Источники контроля целостности преобразователя			
Ток AIN+	–100 (тип)	AIN+ выбранный положительный вход модуля основного АЦП	[нА]
Ток AIN–	+100 (тип)	AIN– выбранный отрицательный вход дополнительного АЦП	[нА]

Продолжение табл. 1.1

Параметр	**Значение**	**Условия измерения**	**Единицы измерения**
Начальный допуск при +25 °C	±10 (тип)		[%]
Дрейф	0,03 (тип)		[%/°C]
Источники тока возбуждения			
Выходной ток	–200 (тип)	От каждого из источников	[мкА]
Начальный допуск при +25 °C	±10 (тип)		[%]
Дрейф	200 (тип)		[ppm/°C]
Начальное согласование токов при +25 °C	±1 (тип)		[%]
Взаимный дрейф	20 (тип)		[ppm/°C]
Нестабильность по напряжению (AV_{DD})	1 (тип)	AV_{DD} =5 В + 5 %	[мкА/В]
Нестабильность по нагрузке	0,1 (тип)		[мкА/В]
Допустимое входное напряжение	AV_{DD}–0,6 В (max) AGND(min)		[В] [В]
Логические входы			
Все, кроме SCLOCK, RESET и XTAL1			
V_{INL}, низкий уровень напряжения	0,8 (max) 0,4 (max)	DV_{DD}=5 В DV_{DD}=3 В	[В] [В]
V_{INH}, высокий уровень напряжения	2,0 (min)		[В]
Только SCLOCK и RESET		(На входе триггер Шмидта) [2]	
V_{T+}	1,3/3 (min/max) 0,95/2,5 (min/max)	DV_{DD} =5 В DV_{DD} =3 В	[В] [В]
$V_{T–}$	0,8/1,4 (min/max) 0,4/1,1 (min/max)	DV_{DD} =5 В DV_{DD} =3 В	[В] [В]
V_{T+} - $V_{T–}$	0,3/0,85 (min/max) 0,3/0,85 (min/max)	DV_{DD} =5 В DV_{DD} =3 В	[В] [В]
Входные токи			
Порт 0, P1.2 – P1.7, EA/	±10 (max)	V_{IN}=0 В или V_{DD}	[мкА]
SCLOCK, SDATA/ MOSI, MISO, SS/ [12]	–10/–40 (min/max) ±10 (max)	V_{IN}=0 В, DV_{DD}=5 В, внутр. нагрузка V_{IN}=V_{DD}, V_{DD} = 5 В	[мкА] [мкА]
RESET	±10 (max) 35/105 (min/max)	V_{IN}=0 В, V_{DD} = 5 В V_{IN}=V_{DD}, DV_{DD}=5 В, внутр. нагрузка	[мкА] [мкА]

Продолжение табл. 1.1

Параметр	**Значение**	**Условия измерения**	**Единицы измерения**
P1.0, P1.1, порты 2 и 3	±10 (max) –180 (min) –660 (max) –20 (min) –75 (max)	$V_{IN}=V_{DD}$, V_{DD} = 5 В V_{IN}=2 В, V_{DD} = 5 В V_{IN}=450 мВ, V_{DD} = 5 В	[мкА] [мкА] [мкА] [мкА] [мкА]
Входная емкость	5 (min)	Все цифровые входы	[пФ]
Резонатор (XTAL1 и XTAL2)			
Логический вход XTAL1			
V_{INL}, низкий уровень напряжения	0,8 (max) 0,4 (max)	DV_{DD}=5 В DV_{DD} =3 В	[В] [В]
V_{INH}, высокий уровень напряжения	3,5 (min) 2,5 (min)	DV_{DD} =5 В DV_{DD} =3 В	[В] [В]
Входная емкость XTAL1	18 (тип)		[пФ]
Выходная емкость XTAL2	18 (тип)		[пФ]
Логические выходы (кроме XTAL2) [2]			
V_{OH}, высокий выходной уровень	2,4 (min) 2,4 (min)	VDD= 5 В, I_{SOURSE}=80 мкА VDD= 3 В, I_{SOURSE}=20 мкА	[В] [В]
V_{OL}, низкий выходной уровень [13]	0,4 (max) 0,4 (max) 0,4 (max)	I_{SINK}=8 мА, SCLOCK, SDATA/MOSI I_{SINK}=10 мА, P1.0 и P1.1 I_{SINK}=1,6 мА, все другие	[В] [В] [В]
Ток утечки в Z-состоянии	±10 (max)		[мкА]
Емкость выхода в Z-состоянии	5 (тип)		[пФ]
Монитор источника питания			
Диапазон порога срабатывания по AV_{DD}	2,63 (min) 4,63 (max)	Программируется 4 значения через TPA1–TPA0 регистра PSMCON	[В] [В]
Точность установки порога по AV_{DD}	±3,5 (max)		[%]
Диапазон порога срабатывания по DV_{DD}	2,63 (min) 4,63 (max)	Программируется 4 значения через TPD1–TPD0 регистра PSMCON	[В] [В]
Точность установки порога по DV_{DD}	±3,5 (max)		[%]
Сторожевой таймер			
Величина периода	0 2 000	Программируется 9 временных интервалов через PRE3-PRE0 регистра WDCON	[мс] [мс]
Тактовая частота ядра МПУ			
Тактовая частота МПУ[2]	98,3 (min) 12,58 (max)	Тактовая частота генерируется встроенной системой ФАПЧ, программируется через CD2-CD0 регистра PLLCON	[кГц] [МГц]
Задержка запуска			
По включению питания	300 (тип)		[мс]

Продолжение табл. 1.1

Параметр	**Значение**	**Условия измерения**	**Единицы измерения**
По выходу из холостого режима	1 (тип)		[мс]
По выходу из режима «питание снято»: Тактовый генератор включен		(Бит OSC_PD=0 в регистре PLLCON)	
По сигналу прерывания INT0/	1 (тип)		[мс]
По сигналу прерывания SPI/I2C	1 (тип)		[мс]
По сигналу прерывания TIC	1 (тип)		[мс]
По внешнему сигналу RESET	3,4 (тип)		[мс]
Тактовый генератор выключен		(Бит OSC_PD=1 в регистре PLLCON)	
По внешнему сигналу RESET	0,9 (тип)		[с]
По внешнему сигналу RESET в нормальном режиме	3,3 (тип)		[мс]
По сбросу от WDT в нормальном режиме	3,3 (тип)	Управляется через регистр WDCON	[мс]
Спецификация Flash/EE-памяти[14]			
Надежность[15]	100 000(min)		[циклов]
Сохранность данных[16]	100 (min)		[лет]
Требования к источникам питания			
Напряжения источников питания		Источники DV_{DD} и AV_{DD} можно устанавливать независимо	
AV_{DD}, 3В номинально	2,7–3,6		[В]
AV_{DD}, 5В номинально	4,75–5,25		[В]
DV_{DD}, 3В номинально	2,7–3,6		[В]
DV_{DD}, 5В номинально	4,75–5,25		[В]
Токи потребления от источников питания в нормальном режиме[17,18]			
Ток от DV_{DD}	4 (max) 2,1 (max)	DV_{DD} =4,75–5,25 В, CLK=1,57 МГц DV_{DD} =2,7–3,6 В, CLK=1,57 МГц	[мА] [мА]
Ток от AV_{DD}	170 (max)	DV_{DD} =5,25 В, CLK=1,57 МГц	[мкА]
Ток от DV_{DD}	15 (max) 8 (max)	DV_{DD} =4,75–5,25 В, CLK=12,58 МГц DV_{DD} =2,7–3,6 В, CLK=12,58 МГц	[мА] [мА]
Ток от AV_{DD}	170 (max)	DV_{DD} =5,25 В, CLK=12,58 МГц	[мкА]
Токи потребления от источников питания в холостом режиме[17,18]			
Ток от DV_{DD}	1,2 (max) 750 (max)	DV_{DD} =4,75–5,25 В, CLK=1,57 МГц DV_{DD} =2,7–3,6 В, CLK=1,57 МГц	[мА] [мкА]
Ток от AV_{DD}	140 (max)	DV_{DD} =5,25 В, CLK=1,57 МГц	[мкА]
Ток от DV_{DD}	2 (max) 1 (max)	DV_{DD} =4,75–5,25 В, CLK=12,58 МГц DV_{DD} =2,7–3,6 В, CLK=12,58 МГц	[мА] [мА]
Ток от AV_{DD}	140 (max)	DVDD =5,25 В, CLK=12,58 МГц	[мкА]

Окончание табл. 1.1

Параметр	Значение	Условия измерения	Единицы измерения
Токи потребления от источников питания в режиме «питание снято»[17,18] CLK=1,57 МГц – 12,58 МГц			
Ток от DV_{DD}	50 (max)	DV_{DD} =4,75–5,25 В, Тактовый генератор включен, TIC включен	[мкА]
Ток от DV_{DD}	20 (max)	DV_{DD} =2,7–3,6 В, Тактовый генератор включен, TIC включен	[мкА]
Ток от AV_{DD}	1 (max)	AV_{DD} =5,25 В, Тактовый генератор включен или выключен	[мкА]
Ток от DV_{DD}	20 (max)	DV_{DD} =4,75–5,25 В, Тактовый генератор выключен	[мкА]
Ток от DV_{DD}	5 (max)	DV_{DD} =2,7–3,6 В, Тактовый генератор выключен	[мкА]
Типичные токи потребления модулей от источников питания (AI_{DD} и DI_{DD}) CLK=1,57 МГц, AV_{DD} =DV_{DD} =5 В			
Периферия PSM	50 (тип)		[мкА]
Основной АЦП	1 (тип)		[мА]
Дополнительный АЦП	500 (тип)		[мкА]
ЦАП	150 (тип)		[мкА]
Сдвоенный источник тока	400 (тип)		[мкА]

[1] Температурный диапазон –40 ... +85 °С.

[2] Эти данные не являются результатом испытаний, но гарантируются самой конструкцией и/или характеристикой при выпуске устройства.

[3] Указанная ошибка может быть скомпенсирована системной калибровкой «нуля».

[4] Основной АЦП калибруется при изготовлении в условиях +25 °С, AV_{DD}=DV_{DD}=5В, что обеспечивает ошибку верхнего предела (полной шкалы) 10 мкВ. Если условия эксплуатации по питанию или температуре существенно отличаются от приведенных, то внутренняя калибровка полной шкалы восстановит указанное значение (10 мкВ).

Системная калибровка «нуля» и верхнего предела устранит указанную ошибку.

[5] Дрейф усиления является дрейфом диапазона устройства. Для получения дрейфа полной шкалы следует добавить к дрейфу усиления дрейф смещения.

[6] Дополнительный АЦП калибруется при изготовлении в условиях +25 °С, AV_{DD}=DV_{DD}=5В, что обеспечивает ошибку верхнего предела (полной шкалы) 2,5 МЗР. Системная калибровка нуля и верхнего предела устранит указанную ошибку.

[7] Линейность ЦАП и параметры спецификации по переменному току рассчитываются при использовании:

– уменьшенного диапазона от 48 до 4 095 при 0 до V_{REF},

– уменьшенного диапазона от 48 до 3 995 при 0 до V_{DD}.

[8] Ошибка усиления является ошибкой диапазона ЦАП.

[9] В общем виде, диапазон биполярного входного напряжения основного АЦП:

$$Range_{ADC} = \pm(V_{REF}\, 2^{RN})/125,$$

где V_{REF} = напряжение REFIN(+) относительно REFIN(–),

V_{REF} = 1,25 В, если выбран внутренний V_{REF},

RN = десятичный эквивалент битов RN2, RN1, RN0,

т. е. если V_{REF} = 2,5В и RN2, RN1, RN0 = 1, 1, 0, то диапазон составит ±1,28 В.

В однополярном режиме в этом случае эффективный диапазон составит 0 В – 1,28 В.

[10] Когда для АЦП выбран внутренний ИОН битами XREF0 и XREF1 в регистрах ADC0CON и ADC1CON, соответственно, то в качестве опорного напряжения используется напряжение 1,25 В.

[11] В биполярном режиме на дополнительный АЦП можно подавать напряжение не меньше, чем (AGND – 30 мВ), как указано в предельно допустимых параметрах. Несмотря на то, что биполяр-

ный режим заявляется для диапазона от –VREF до +VREF, отрицательное напряжение ограничено величиной – 30 мВ.

[12] Выводы модулей интерфейсов I^2C или SPI для данного испытания конфигурируются как цифровые входы.

[13] Выводы сконфигурированы только для режима интерфейса I^2C.

[14] Характеристики сохранности данных Flash/EE-памяти справедливы как для памяти программ, так и для памяти данных.

[15] Надежность определяется как 100 000 циклов стирания/записи в соответствии с документом «JEDEC Std. 22 Method A117» и измеряется в температурном диапазоне – 40 °С...+25 °С и +85 °С. Типовая надежность при температуре +25 °С составляет 700 000 циклов стирания/записи.

[16] Эквивалентное время сохранности при температуре перехода (TJ) = +55 °С в соответствии с документом «JEDEC Std. 22 Method. A117». Интервал сохранности, основанный на значении энергии активации 0,6 эВ, уменьшается с ростом температуры.

[17] Потребление тока от источника измеряется для трех режимов: нормального, холостого, «питание снято» при следующих условиях:

1) нормальный режим: RESET = 0,4 В, цифровые порты ввода/вывода отключены от нагрузки, тактовая частота ядра меняется битами CD в регистре PLLCON, ядро выполняет программный цикл во внутренней памяти;

2) холостой режим: RESET= 0,4 В, цифровые порты ввода/вывода отключены от нагрузки, тактовая частота ядра меняется битами CD в регистре PLLCON, PCON.0 = 1, выполнение программы приостановлено;

3) режим «питание снято»: RESET = 0,4 В, все контакты P0 и контакты P1.2 – P1.7 = 0,4 В, все прочие цифровые порты ввода/вывода отключены от нагрузки, тактовая частота ядра меняется битами CD в регистре PLLCON, PCON.1 = 1, выполнение программы приостановлено, тактовый генератор либо включен, либо выключен в соответствие с состоянием бита OSC_PD (PLLCON.7) в регистре PLLCON.

[18] Ток, потребляемый от источника DVDD во время выполнения цикла программирования или стирания Flash/EE-памяти, увеличится приблизительно на 3 мА (при питании 3 В) и на 10 мА (при питании 5 В).

Микроконвертор спроектирован для работы с внешним кварцевым резонатором на частоту 32 768 Гц, из которой встроенная система ФАПЧ вырабатывает внутреннюю рабочую частоту 12,582 912 МГц. Эта частота поступает на программируемый делитель, с выхода которого снимается рабочая тактовая частота вычислительного ядра микропроцессорного устройства. Такая схема организации тактирования позволяет ослабить вредное влияние паразитных высокочастотных токов, протекающих по общей шине и шине питания устройства, на точность аналого-цифровых преобразований. Процессорное ядро представляет собой микроконтроллер с системой команд, совместимой с набором инструкций семейства 8051. Машинный цикл ядра состоит из двенадцати циклов выбранной рабочей тактовой частоты.

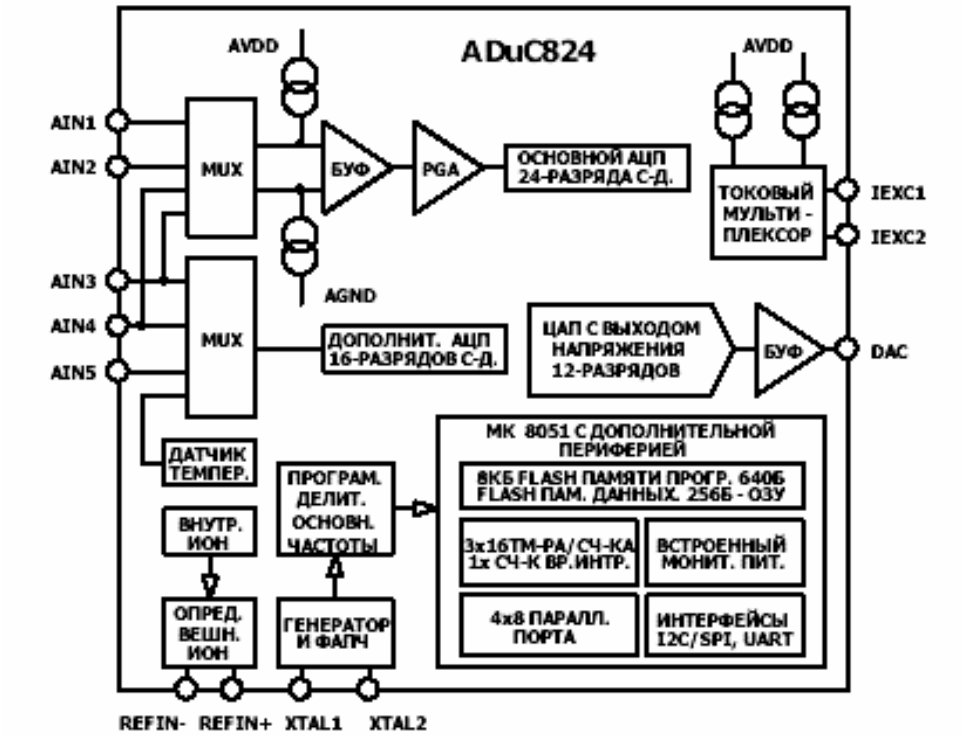


Рис. 1.1. Структура МК

МК имеет 8 кбайт Flash-памяти программ, 640 байт Flash-памяти данных и 256 байт оперативной памяти данных с произвольным доступом. В состав МК включены также 12-разрядный ЦАП с выходом напряжения, два источника тока, монитор источника питания. Встроенная цифровая периферия МК включает в себя сторожевой таймер, счетчик временных интервалов (реального времени), три таймера-счетчика и модули последовательных портов UART и I2C/SPI.

МК поддерживает режимы последовательной загрузки и отладки через UART, а также режим эмуляции через одну линию (ножку EA/). Устройство питается от однополярного источника с напряжением +3…+5 В. При напряжении источника +3 В потребляемая микроконвертором мощность составляет менее 10 мВт. Конструктивно МК выпускается в 52-контактном корпусе типоразмера MQFP (прил. 1). Блок-схема микроконвертора приведена на рис. 1.2. Описание выводов микроконвертора приведено в табл. 1.2.

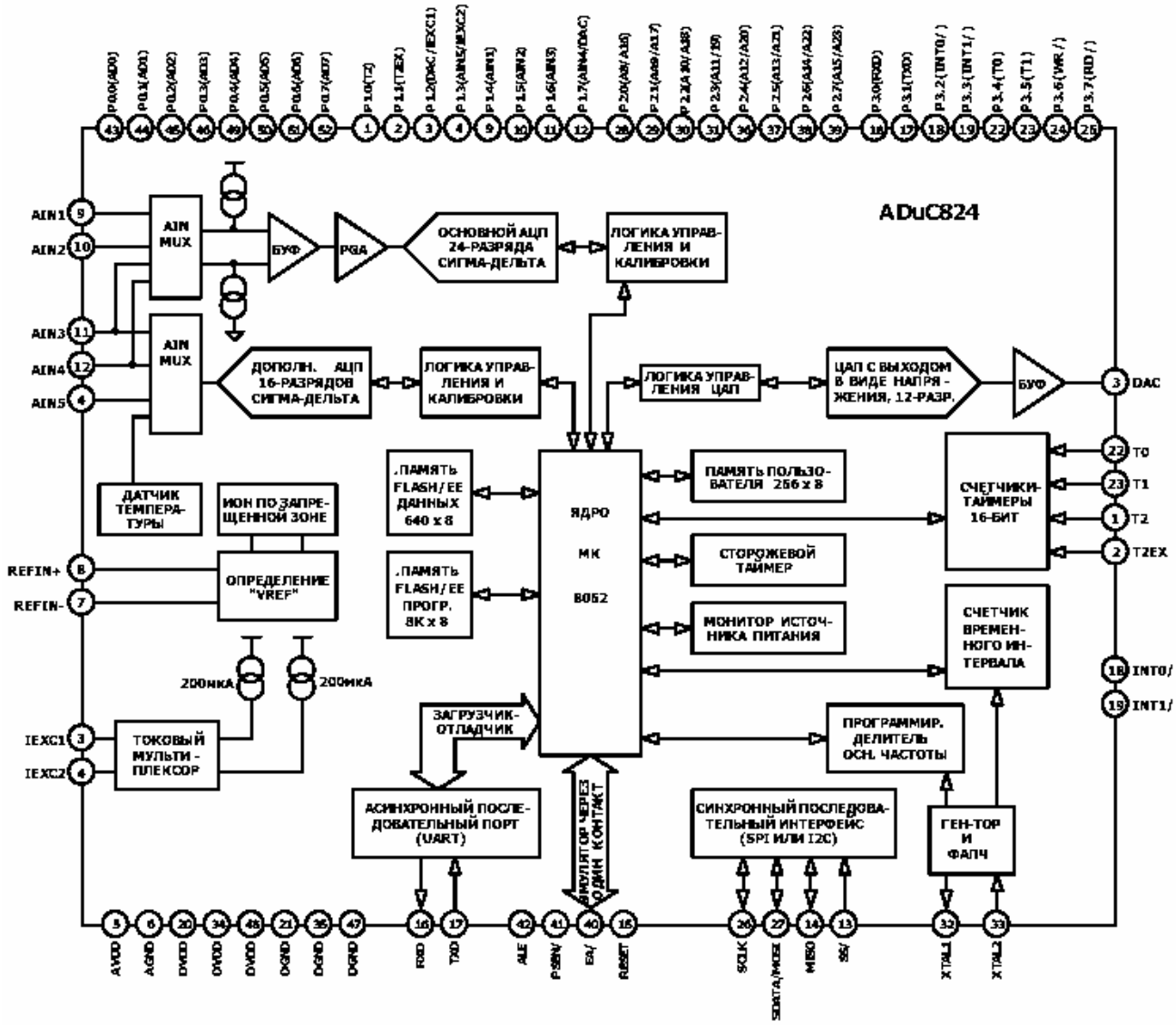


Рис. 1.2. Блок-схема микроконвертора

Таблица 1.2

Описание выводов ADuC824

№	Мнемоническое обозначение вывода	Вх./ вых./ питающ. (I/O/S)	Функция вывода
1	P1.0/T2	I/O	Может быть цифровым входом или выходом и имеет внутренний подтягивающий к «плюсу» питания резистор. P1.0 имеет также усиленный выходной буфер, допускающий значение втекающего тока до 10 мА. P1.0 совмещен со входом приема синхроимпульсов ТС 2.
2	P1.1/T2EX	I/O	Может быть цифровым входом или выходом и имеет внутренний подтягивающий к «плюсу» питания резистор. P1.1 имеет также усиленный выходной буфер, допускающий втекающий ток до 10 мА. P1.0 совмещен со входом приема сигнала управления ТС 2.

Продолжение табл. 1.2

№	Мнемоническое обозначение вывода	Вх./ вых./ питающ. (I/O/S)	Функция вывода
3	P1.2/DAC/IEXC1	I/O	Не имеет цифрового выходного каскада, он работает только как цифровой вход, для чего необходимо записать «0» в бит порта. Как цифровой вход порт непосредственно считывает внешний низкий или высокий уровень. P1.2 может быть также сконфигурирован как выход аналогового напряжения модуля ЦАП или для вывода одного из двух токов возбуждения (200 мкА или 2×200 мкА).
4	P1.3/AIN5/IEXC2	I	Не имеет цифрового выходного каскада, он работает только как цифровой вход, для чего необходимо записать «0» в бит порта. Как цифровой вход порт непосредственно считывает внешний низкий или высокий уровень. P1.3 может быть также сконфигурирован как аналоговый вход (AIN5) модуля дополнительного АЦП или для вывода одного из двух токов возбуждения (200 мкА или 2×200 мкА).
5	Avdd	S	Вывод подключения источника аналогового питания 3 В или 5 В.
6	AGND	S	Аналоговая «земля». Общий вывод для аналоговой части устройства.
7	REFIN(–)	I	Вывод подключения отрицательного полюса опорного напряжения.
8	REFIN(+)	I	Вывод подключения положительного полюса опорного напряжения.
9 – 11	P1.4 – P1.6	I	P1.4–P1.6 не имеют цифровых выходных каскадов, они работают только как цифровые входы, для чего необходимо записать «0» в соответствующие биты портов. Как цифровые входы порты непосредственно считывают внешние низкие или высокие уровни. Эти порты имеют следующие альтернативные аналоговые функции:
	P1.4/AIN1	I	• аналоговый вход положительного напряжения модуля основного АЦП;
	P1.5/AIN2	I	• аналоговый вход отрицательного напряжения модуля основного АЦП;
	P1.6/AIN3/IEXC	I	• аналоговый вход дополнительного АЦП или мультиплексированный вход положительного напряжения основного АЦП.
12	P1.7/AIN4/DAC	I/O	Не имеет цифрового выходного каскада, он работает только как цифровой вход, для чего необходимо записать «0» в бит порта. Как цифровой вход порт непосредственно считывает внешний низкий или высокий уровень. Может быть также сконфигурирован как аналоговый вход (AIN4) модуля дополнительного АЦП или как вход отрицательного напряжения модуля основного АЦП. Может быть также сконфигурирован как выход аналогового напряжения модуля ЦАП.
13	SS/	I	Вход выбора ведомого устройства модуля интерфейса SPI. Этот вывод «слабо подтянут» к «плюсу» питания.
14	MISO	I/O	Вход ведущего/выход ведомого устройства модуля интерфейса SPI. Этот вывод «слабо подтянут» к «плюсу» питания.

Продолжение табл. 1.2

№	Мнемоническое обозначение вывода	Вх./ вых./ питающ. (I/O/S)	Функция вывода
15	RESET	I	Вход сброса. Высокий уровень на этом выводе в течение 24 циклов тактовой частоты при работающем тактовом генераторе осуществляет сброс устройства. На этом выводе установлен триггер Шмидта и он «слабо подтянут» к «плюсу» питания.
16 – 19	P3.0–P3.3	I/O	P3.0–P3.3 – выводы двунаправленных портов, имеющие внутренние подтягивающие к «плюсу» питания резисторы. Выводы, в триггеры которых записаны «1», с помощью подтягивающих резисторов устанавливаются в высокое состояние и в таком виде могут использоваться как входы. Из-за наличия подтягивающих резисторов при низком внешнем уровне на выводе порта во внешнюю цепь будет втекать ток. При работе в качестве выходов и генерации на выходе положительного перепада вывод порта в течение двух периодов тактовой частоты находится в фазе активного (сильноточного) разряда переходных емкостей. Эти порты имеют следующие альтернативные функции:
	P3.0/RXD	I/O	• вход данных асинхронного приемника UART или вход/выход данных синхронного последовательного порта обмена;
	P3.1/TXD	I/O	• выход данных асинхронного передатчика UART или выход синхросигналов синхронного последовательного порта обмена;
	P3.2/INT0/	I/O	• вход внешнего прерывания 0. Этот вывод можно использовать также как вход управления разрешением ТС 0;
	P3.3/INT1/	I/O	• вход внешнего прерывания 1. Этот вывод можно использовать также как вход управления разрешением ТС 1.
20, 34, 48	DVdd	S	Вывод подключения источника цифрового питания 3 В или 5 В.
21, 35, 47	DGND	S	Цифровая «земля». Общий вывод для цифровой части устройства.
22 – 25	P3.4 – P3.7	I/O	P3.4–P3.7 – выводы двунаправленных портов, имеющие внутренние подтягивающие к «плюсу» питания резисторы. Выводы, в триггеры которых записаны «1», с помощью подтягивающих резисторов устанавливаются в высокое состояние и в таком виде могут использоваться как входы. Из-за наличия подтягивающих резисторов при низком внешнем уровне на выводе порта во внешнюю цепь будет втекать ток. При работе в качестве выходов и генерации на выходе положительного перепада вывод порта в течение двух периодов тактовой частоты находится в фазе активного (сильноточного) разряда переходных емкостей. Эти порты имеют следующие альтернативные функции:
	P3.4/T0	I/O	Вход ТС 0.
	P3.5/T1	I/O	Вход ТС1.
	P3.6/WR/	I/O	Выход управления записью. Защелкивает байт данных из порта 0 во внешнюю память данных.
	P3.7/RD/	I/O	Выход управления чтением. Разрешает передачу данных из внешней памяти в порт 0.
26	SCLK	I/O	Синхросигнал модуля последовательного интерфейса I2C или SPI. В режиме входа к этому выводу подключен внутренний триггер Шмидта и он «слабо подтянут» к «плюсу» питания, если только не генерирует низкий уровень.

Продолжение табл. 1.2

№	Мнемоническое обозначение вывода	Вх./ вых./ питающ. (I/O/S)	Функция вывода
27	SDATA/MOSI	I/O	Вывод последовательного ввода/ввода данных для модуля интерфейса I^2C. Для модуля интерфейса SPI это выход ведущего/вход ведомого устройства. Этот вывод «слабо подтянут» к «плюсу» питания, если только не генерирует низкий уровень.
28 – 31	P2.0 – P2.3 (A8 – A11) (A16 – A19)	I/O	Выводы двунаправленных портов, имеющие внутренние подтягивающие к «плюсу» питания резисторы. Выводы, в триггеры которых записаны «1», с помощью подтягивающих резисторов устанавливаются в высокое состояние и в таком виде могут использоваться как входы. Из-за наличия подтягивающих резисторов при низком внешнем уровне на выводе порта во внешнюю цепь будет втекать ток. Порт 2 содержит старший байт адреса при обращении к внешней памяти программ и средний и старший байты адреса при обращении к памяти данных с 24-разрядным адресным пространством.
32	XTAL1	I	Вход инвертора внутреннего тактового генератора.
33	XTAL2	O	Выход инвертора внутреннего тактового генератора.
36, 37, 38, 39	P2.4 – P2.7 (A12 – A15) (A20 – A23)	I/O	Выводы двунаправленных портов, имеющие внутренние подтягивающие к «плюсу» питания резисторы. Выводы, в триггеры которых записаны «1», с помощью подтягивающих резисторов устанавливаются в высокое состояние и в таком виде могут использоваться как входы. Из-за наличия подтягивающих резисторов при низком внешнем уровне на выводе порта во внешнюю цепь будет втекать ток. Порт 2 содержит старший байт адреса при обращении к внешней памяти программ и средний и старший байты адреса при обращении к памяти данных с 24-разрядным адресным пространством.
40	EA/	I/O	Вход разрешения доступа к внешней памяти. Высокий уровень на этом входе разрешает выборку кода из внутренней памяти программ с адресами от 0000h до 1FFFh. Низкий уровень на этом входе разрешает выборку кода из внешней памяти программ. Для определения режима доступа к памяти (из внутренней или из внешней) состояние входа EA/ опрашивается в момент окончания активного уровня сигнала на входе RESET или при начальной выдаче питания. EA/ можно использовать как линию ввода/вывода для внешней эмуляции, поэтому уровень на этом входе не должен изменяться во время обычной работы, так как это изменение может вызвать эмулирующее прерывание, которое остановит выполнение программы.
41	PSEN/	O	Выход разрешения (выбора) внешней памяти. Сигнал с этого выхода разрешает доступ внешней памяти программ к шине во время операций выборки. Сигнал активен в каждом цикле из шести периодов тактового генератора, кроме случая, когда осуществляется доступ к внешней памяти данных. Выход PSEN/ остается в высоком состоянии при обращении к внутренней памяти программ. Выход PSEN/ можно использовать как вход разрешения режима последовательной загрузки, если его удерживать в низком уровне в момент окончания активного уровня сигнала на входе RESET или при начальной выдаче питания.

Окончание табл. 1.2

№	Мнемоническое обозначение вывода	Вх./ вых./ питающ. (I/O/S)	Функция вывода
42	ALE	O	Выход управления защелкиванием (фиксацией) адреса, установленного на шине. Выход ALE используется для фиксации младшего байта адреса (а также байта страницы при доступе к внешней памяти данных с 24-разрядным адресным пространством) внешней памяти во время циклов доступа к памяти программ или данных. Сигнал ALE активен в каждом цикле из шести периодов тактового генератора, кроме случая, когда осуществляется доступ к внешней памяти данных. Этот сигнал можно запретить путем установки бита PCON.4 в регистре PCON
43 – 46	P0.0 – P0.3 (AD0 – AD3)	I/O	Являются выводами части 8-разрядного двунаправленного порта 0 с открытым стоком. Если в порт 0 записаны «1», то соответствующие выводы будут «плавать» (иметь неопределенное состояние) и в этом состоянии их можно использовать как входы с высоким входным сопротивлением. Для правильной передачи портом 0 высокого уровня необходимо наличие внешних подтягивающих к «плюсу» питания резисторов. Порт 0 также мультиплексирован с младшим байтом адреса и шиной данных при доступе к внешней памяти программ и данных. В этом случае порт для передачи высокого уровня использует подтягивание к «плюсу» питания через резистор с небольшим сопротивлением (сильноточное).
49 – 52	P0.4 – P0.7 (AD4 – AD7)	I/O	Являются выводами части 8-разрядного двунаправленного порта 0 с открытым стоком. Если в порт 0 записаны «1», то соответствующие выводы будут «плавать» (иметь неопределенное состояние) и в этом состоянии их можно использовать как входы с высоким входным сопротивлением. Для правильной передачи портом 0 высокого уровня необходимо наличие внешних подтягивающих к «плюсу» питания резисторов. Порт 0 также мультиплексирован с младшим байтом адреса и шиной данных при доступе к внешней памяти программ и данных. В этом случае порт для передачи высокого уровня использует подтягивание к «плюсу» питания через резистор с небольшим сопротивлением (сильноточное).

1.2. Организация памяти и программная модель

Как и все 8051-совместимые микроконтроллеры, МК ADuC824 имеет разделенное пространство памяти программ и данных, как показано на рис. 1.3 и 1.4. При включении питания или при осуществлении «горячего» сброса устройства в случае, если на ножке EA установлен низкий логический уровень, микроконвертор будет выполнять код из области внешней программной памяти. В противном случае выполняется код из внутренней Flash-памяти программ. Внутренняя программная память может программироваться через UART даже в составе целевого устройства (внутрисхемно).

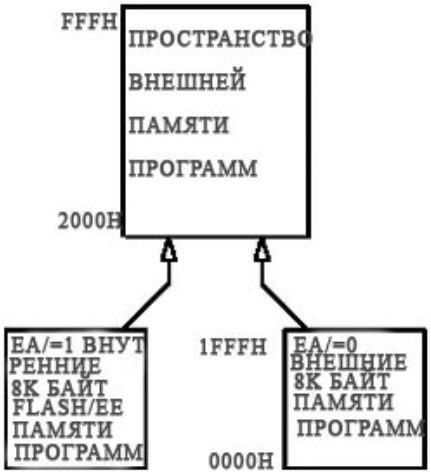


Рис. 1.3. Распределение памяти МК

Рис. 1.4. Дополнительные 640 байт Flash-памяти данных

Адресное пространство памяти данных состоит из пространства внутренней и внешней областей. Область внутренней памяти подразделяется на четыре отдельных области: младшие 128 байт ОЗУ, старшие 128 байт ОЗУ, 128 байт регистров специальных функций (SFR) и 640 байт Flash-памяти данных. Хотя область старших 128 байт ОЗУ и область специальных регистров разделяют одно и то же адресное пространство, программный доступ к ним осуществляется через различные режимы адресации. К младшим 128 байтам памяти данных можно получить доступ с помощью прямой или косвенной адресации, к старшим 128 байтам ОЗУ – только с помощью косвенной, а к области специальных регистров – только с помощью прямой адресации. На рис. 1.4 показана организация дополнительных 640 байт Flash-памяти данных, доступной для записи и чтения в целевой программе. Доступ осуществляется через группу регистров управления, расположенных в области регистров специальных функций. Внешняя память данных МК может иметь размер до 16 Мбайт, в то время как объем адресуемой памяти стандартного ядра, совместимого с семейством 8051, не превышает 64 кбайт. Для получения более подробной информации о подключении и адресации внешней памяти следует обратиться к [3].

Младшие 128 байт внутренней памяти данных организованы, как показано на рис. 1.5. Тридцать два младших байта сгруппированы в четыре банка по восемь регистров в каждом. Регистры в каждом банке имеют мнемонические имена с R0 по R7. Следующие шестнадцать байт с адресами от 20h до 2Fh, образуют блок памяти с битовой адресацией (128 бит) с адресами битов от 00h до 7Fh. Область стека можно программно располагать в любом месте внутренней памяти, а глубина стека может достигать двухсот пятидесяти шести байт. По умолчанию после сброса указатель стека устанавливается на адрес 07h, а заполнение стека начинается с адреса 08h, который является также первым регистром регистрового банка 1 (R0). Таким образом, если в целевой программе необходимо иметь стек большей глубины, чем размер одного банка регистров, следует программно установить указатель стека в ту область ОЗУ, которая не используется программой для хранения данных. Область специальных регистров располагается в старших 128 байтах пространства внутренней памяти данных, и она доступна программе только через прямую адресацию. Эта область обеспечивает связь между центральным процессором и всей внутренней периферией МК. Блок-схема программной модели МК показана на рис. 1.6.

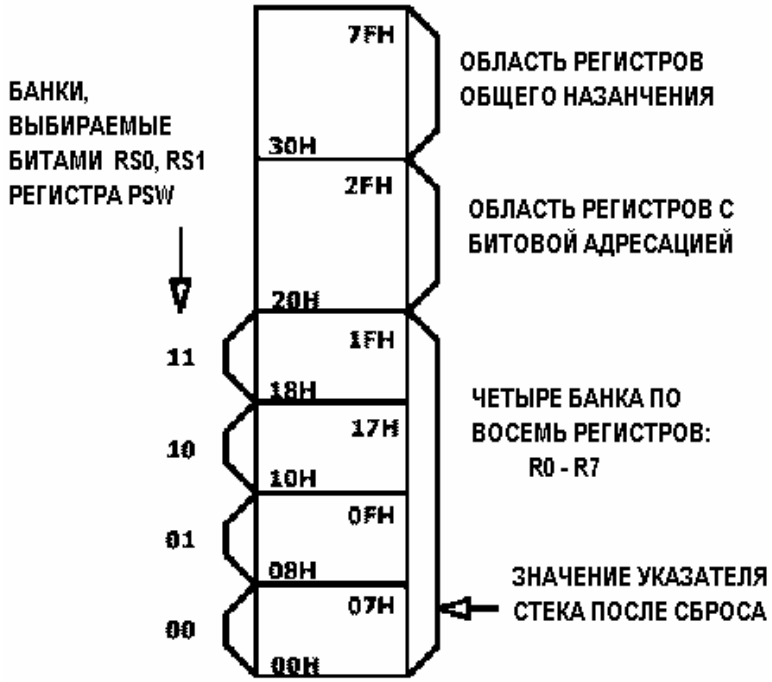


Рис. 1.5. Младшие 128 байт внутренней памяти данных

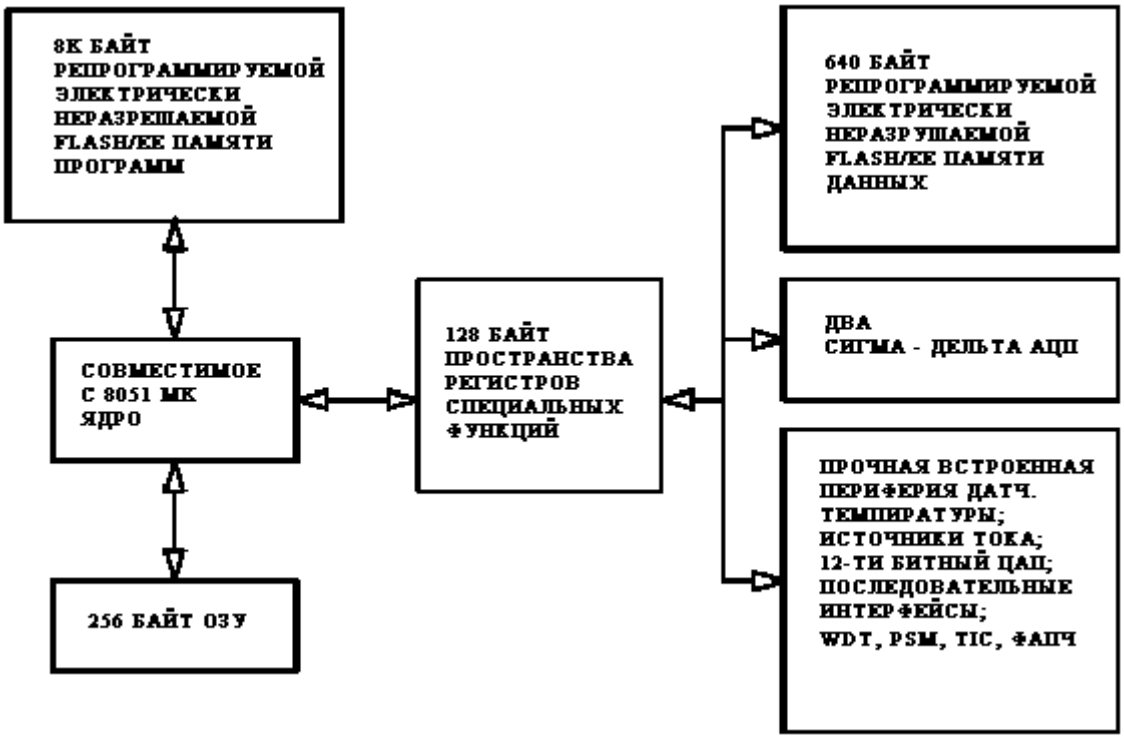


Рис. 1.6. Блок-схема программной модели МК

SPICON F8H 04H	РЕЗЕРВ	РЕЗЕРВ	**DACL** FBH 00H	**DACH** FCH 00H	**DACCON** FDH 00H	РЕЗЕРВ	РЕЗЕРВ
B F0H 00H	РЕЗЕРВ	РЕЗЕРВ	НЕ ИСП.	РЕЗЕРВ	РЕЗЕРВ	РЕЗЕРВ	**SPIDAT** F7H 00H
I2CCON E8H 00H	**GN01*** E9H 55H	**GN0M*** EAH 55H	**GN0H*** EBH 53H	**GN1L*** ECH 9AH	**GN1H*** EDH 59H	РЕЗЕРВ	РЕЗЕРВ
ACC E0H 00H	**OF0L*** E1H 00H	**OF0M*** E2H 00H	**OF0H*** E3H 80H	**OF1L*** E4H 00H	**OF1H*** E5H 80H	РЕЗЕРВ	РЕЗЕРВ
ADCSTAT D8H 00H	**ADC0L** D9H 00H	**ADC0M** DAH 00H	**ADC0H** DBH 00H	**ADC1L** DCH 00H	**ADC1H** DDH 00H	РЕЗЕРВ	**PSMCON** DFH DEH
PSW D0H 00H	**ADCMODE** D1H 00H	**ADC0CON** D2H 07H	**ADC1CON** D3H 00H	**SF** D4H 45H	**ICON** D5H 00H	РЕЗЕРВ	**PLLCON** D7H 03H
T2CON C8H 00H	РЕЗЕРВ	**RCAP2L** CAH 00H	**RCAP2H** CBH 00H	**TL2** CCH 00H	**TH2** CDH 00H	РЕЗЕРВ	РЕЗЕРВ
WDCON C0H 10H	РЕЗЕРВ	**CHIPID** C2H 0xH	РЕЗЕРВ	РЕЗЕРВ	РЕЗЕРВ	**EADRL** C6H 00H	РЕЗЕРВ
IP B8H 00H	**ECON** B9H 00H	РЕЗЕРВ	РЕЗЕРВ	**EDATA1** BCH 00H	**EDATA2** BDH 00H	**EDATA3** BEH 00H	**EDATA4** BFH 00H
P3 B0H FFH	НЕИСП.	НЕИСП.	НЕИСП.	НЕИСП.	РЕЗЕРВ	РЕЗЕРВ.	НЕ ИСП.
IE A8H 00H	**IEIP2** A9H A0H	РЕЗЕРВ	РЕЗЕРВ	РЕЗЕРВ	РЕЗЕРВ	РЕЗЕРВ	РЕЗЕРВ
P2 A0H FFH	**TIMECON** A1H 00H	**HTHSEC** A2H 00H	**SEC** A3H 00H	**MIN** A4H 00H	**HOUR** A5H 00H	**INTVAL** A6H 00H	НЕ ИСП.
SCON 98H 00H	**SBUF** 99H 00H	**I2CDAT** 9AH 00H	**I2CADD** 9BH 55H	НЕ ИСП.	НЕ ИСП.	НЕ ИСП.	НЕ ИСП.
P1 90H FFH	НЕ ИСП.	НЕ ИСП.	НЕ ИСП.	НЕ ИСП.	НЕ ИСП.	НЕ ИСП.	НЕ ИСП.
TCON 88H 00H	**TMOD** 89H 00H	**TL0** 8AH 00H	**TL1** 8BH 00H	**TH0** 8CH 00H	**TH1** 8DH 00H	РЕЗЕРВ	РЕЗЕРВ
P0 80H FFH	**SP** 81H 07H	**DPL** 82H 00H	**DPH** 83H 00H	**DPP** 84H 00H	РЕЗЕРВ	РЕЗЕРВ	**PCON** 87H 00H

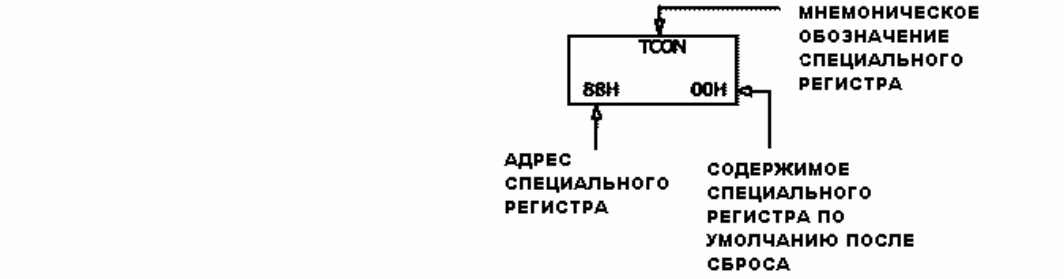


Рис. 1.7. Программная модель МК
(при включении питания в эти регистры записываются калибровочные коэффициенты, значения которых установлены на заводе-изготовителе)

Набор специальных регистров МК включает в себя регистры управления, конфигурации и регистры данных, через которые осуществляется связь между ядром и внутренней периферией устройства. Полная карта специальных регистров с указанием их состояний после сброса приведена на рис. 1.7. Надпись «не исп. (не используется)» указывает на незанятые адреса в области SFR. Их не следует использовать в программе, поскольку физически никаких регистров в памяти по этим адресам не существует. Чтение байта по незанятому адресу SFR возвращает неопределенную величину. Резервные элементы SFR, предназначенные для использования в перспективных изделиях, обозначены на рисунке как «резерв» и их также не следует использовать в целевой программе. На рис. 1.8 приведена карта битов специальных регистров, имеющих битовую адресацию. Адреса бит-адресуемых регистров оканчиваются на 0h или 8h.

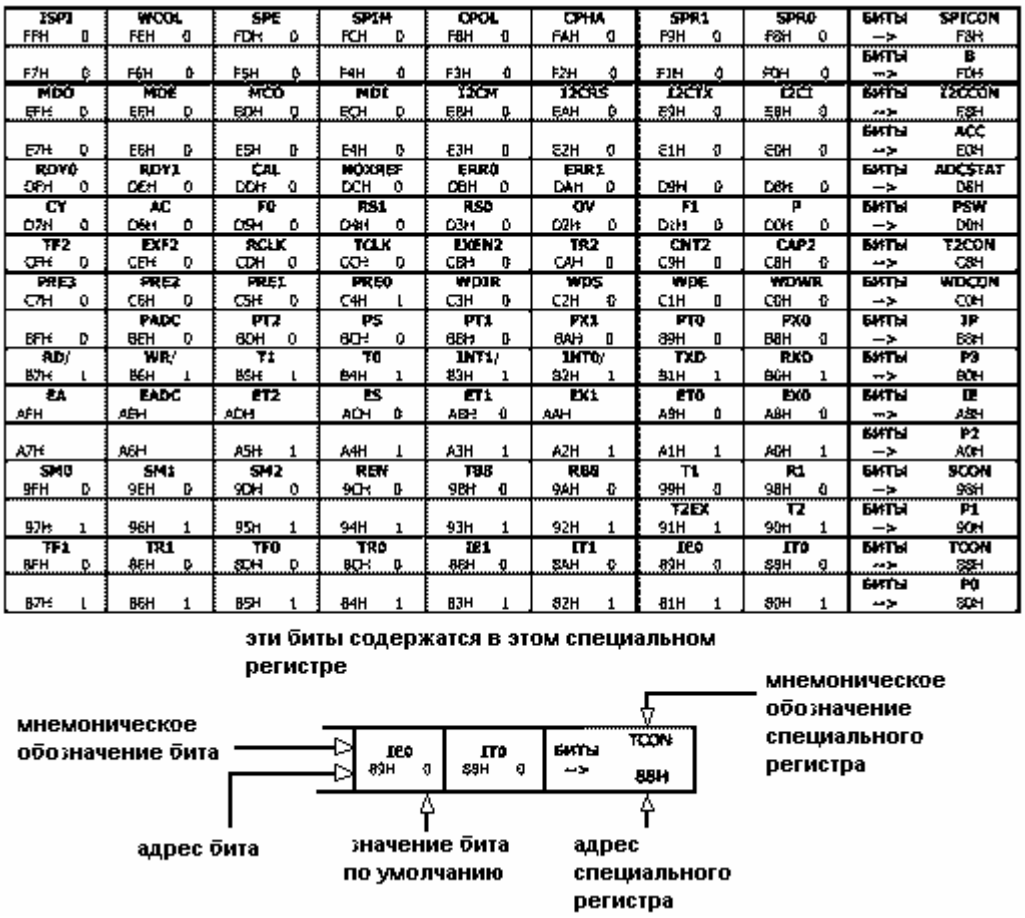


Рис. 1.8. Карта битов специальных регистров с битовой адресацией

1.3. Описание регистров специальных функций

A (аккумулятор)

Адрес E0h, значение после сброса 00h, битовая адресация имеется.

Аккумулятор используется для выполнения математических операций, включая сложение, вычитание, целочисленное умножение и деление, а также для операций с битами. В мнемониках инструкций обращение к аккумулятору выполняется через символьные имена A или ACC.

B (регистр второго операнда)

Адрес F0h, значение после сброса 00h, битовая адресация имеется.

Регистр B используется совместно с аккумулятором для операций умножения и деления. В других инструкциях его можно использовать как регистр ОЗУ общего назначения.

SP (регистр-указатель стека)

Адрес 81h, значение после сброса 07h, битовая адресация отсутствует.

Регистр-указатель стека SP используется для запоминания адреса внутреннего ОЗУ, который называется «вершиной стека». При выполнении инструкций PUSH и CALL регистр SP инкрементируется до записи данных в стек. По умолчанию после сброса в SP записывается значение 07h, поэтому стек начинается с адреса 08h.

DPTR (регистр-указатель данных DPP/DPH/DPL)

Адреса 84h/83h/82h, значения после сброса 00h/00h/00h, битовая адресация отсутствует.

Регистр-указатель данных DPTR составлен из трех 8-битных регистров, называемых DPP (регистр номера страницы), DPH (регистр старшего байта) и DPL (регистр младшего байта). Они используются для обеспечения доступа программы к данным из внешней памяти. С регистром можно работать как с 24-битным (DPTR) либо как с тремя независимыми 8-битными регистрами (DPP, DPH, DPL).

PSW (регистр слова состояния программы)

CY	AC	F0	RS1	RS0	OV	F1	P

Адрес D0h, значение после сброса 00h, битовая адресация имеется.

Регистр PSW содержит биты (флаги), отражающие текущее состояние процессора, как показано в табл. 1.3.

Таблица 1.3

Назначение битов специального регистра PSW

Бит	Имя	Описание		
7	CY	Флаг переноса		
6	AC	Дополнительный флаг переноса		
5	F0	Флаг общего назначения		
4	RS1	Биты выбора банка регистров		
3	RS0	RS1	RS0	Выбранный банк
		0	0	0
		0	1	1
		1	0	2
		1	1	3
2	OV	Флаг переполнения		
1	F1	Флаг общего назначения		
0	P	Бит паритета		

PCON (регистр управления питанием)

CMOD	SERIPD	INT0PD	ALEOFF	GF1	GF0	PD	IDL

Адрес 87h, значение после сброса 00h, битовая адресация отсутствует.

Регистр управления питанием PCON содержит биты выбора вариантов управления питанием с различным уровнем потребляемой мощности, а так же флаги состояния общего назначения, как показано в табл. 1.4.

Таблица 1.4

Назначение битов специального регистра PCON

Бит	Имя	Описание
7	SMOD	Удвоение скорости обмена UART
6	SERIPD	Разрешение прерывания от I2C/SPI в режиме «питание снято»
5	INT0PD	Разрешение прерывания INT0/ в режиме «питание снято»
4	ALEOFF	Запрет выхода ALE
3	GF1	Бит флага общего назначения
2	GF0	Бит флага общего назначения
1	PD	Разрешение режима «питание снято»
0	IDL	Разрешение холостого режима

Регистры обслуживания основного и дополнительного АЦП

Основной и дополнительный модули АЦП управляются и конфигурируются с помощью следующих специальных регистров:

ADCSTAT – статусный регистр АЦП, содержит биты состояния (статуса) основного и дополнительного модулей АЦП.

ADCMODE – регистр режима АЦП, управляющий режимами работы основного и дополнительного модулей АЦП.

ADC0CON – регистр управления основным АЦП, содержит текущую конфигурацию основного модуля АЦП.

ADC1CON – регистр управления дополнительным АЦП, содержит текущую конфигурацию дополнительного модуля АЦП.

SF – регистр цифрового фильтра АЦП. Позволяет установить скорость потока данных на выходе основного и дополнительного АЦП.

ICON – регистр управления источниками тока, позволяет устанавливать различные режимы включения встроенных источников тока.

ADC0L/ADC0M/ADC0H – три 8-битных регистра, содержащие 24-битный результат преобразования основного АЦП.

ADC1L/ADC1H – два 8-битных регистра, содержащих 16-битный результат преобразования дополнительного АЦП.

OF0L/OF0M/OF0H – три 8-битных регистра, содержащих 24-битный калибровочный коэффициент смещения основного АЦП.

OF1L/OF1H – два 8-битных регистра, содержащих 16-битный калибровочный коэффициент смещения дополнительного АЦП.

GN0L/GN0M/GN0H – три 8-битных регистра, содержащих 24-битный калибровочный коэффициент усиления основного АЦП.

GN1L/GN1H – два 8-битных регистра, содержащих 16-битный калибровочный коэффициент усиления дополнительного АЦП.

ADCSTAT (статусный регистр АЦП)

RDY0	RDY1	CAL	NOXREF	ERR0	ERR1	–	–

Адрес D8h, значение после сброса 00h, битовая адресация имеется.

Этот регистр отражает состояние обоих каналов АЦП, включая готовность данных, калибровку и различные относящиеся к АЦП ошибки и предупреждения, включая ошибку определения наличия ИОН и флаг переполнения. Назначение битов регистра ADCSTAT указано в табл. 1.5.

Таблица 1.5

Назначение битов специального регистра ADCSTAT

Бит	Имя	Описание
7	RDY0	Бит готовности основного АЦП. Устанавливается по завершению преобразования АЦП или по завершению цикла калибровки. Сбрасывается непосредственно пользователем или косвенно путем записи битов запуска следующего цикла преобразования или калибровки основного АЦП. Основному АЦП запрещается запись результатов в регистры данных или калибровки до тех пор, пока бит RDY0 не сброшен
6	RDY1	Бит готовности дополнительного АЦП. Все определения для RDY0 (модуля основного АЦП) справедливы и для RDY1 (модуля дополнительного АЦП).
5	CAL	Бит состояния калибровки. Устанавливается аппаратно по завершению цикла калибровки. Сбрасывается косвенно путем записи битов запуска следующего цикла преобразования или калибровки
4	NOXREF	Бит отсутствия внешнего ИОН. Активен в случае, если активны модули основного или дополнительного АЦП. Устанавливается для индикации того, что один или оба вывода REFIN никуда не подключены или, что приложенное между ними напряжение ниже нормированного порога. Если при использовании внешнего ИОН этот бит устанавливается, то результатом преобразования будут «единицы» во всех разрядах регистров данных. Бит сбрасывается для индикации того, что величина V_{REF} требуемого номинала
3	ERR0	Бит ошибки основного АЦП. Устанавливается аппаратно в случае, если результаты преобразования, записанные в регистры данных АЦП, фиксируются на уровне либо все «нули», либо все «единицы». После калибровки установка этого бита сигнализирует об ошибке, вызванной тем, что в регистре калибровки не было произведено записи. Сбрасывается путем записи битов начала преобразования или калибровки
2	ERR1	Бит ошибки дополнительного АЦП. Все определения для ERR0 (модуля основного АЦП) справедливы и для ERR1 (модуля дополнительного АЦП)
1	–	Зарезервирован для дальнейшего использования
0	–	Зарезервирован для дальнейшего использования

ADCMODE (регистр режима АЦП)

–	–	ADC0EN	ADC1EN	–	MD2	MD1	MD0

Адрес D1h, значение после сброса 07h, битовая адресация отсутствует.

Этот регистр используется для управления режимами работы основного и дополнительного модулей АЦП. Назначение битов регистра ADCMODE указано в табл. 1.6.

Таблица 1.6

Назначение битов специального регистра ADCMODE

Бит	Имя	Описание			
7	–	Зарезервирован для дальнейшего использования.			
6	–	Зарезервирован для дальнейшего использования.			
5	ADC0EN	Разрешение основного АЦП. Устанавливается пользователем для того, чтобы разрешить работу модуля основного АЦП и определить его режим в соответствии с выбранными битами режима MD2-MD0. Сбрасывается пользователем для установки модуля основного АЦП в режим «снятое питание».			
4	ADC1EN	Разрешение дополнительного АЦП. Устанавливается пользователем для того, чтобы разрешить работу модуля дополнительного АЦП и определить его режим в соответствии с выбранными битами режима MD2-MD0. Сбрасывается пользователем для установки модуля дополнительного АЦП в режим «снятое питание».			
3	–	Зарезервирован для дальнейшего использования.			
2 1 0	MD2 MD1 MD0	Биты режима основного и дополнительного АЦП. Эти биты определяют режим работы разрешенного модуля АЦП следующим образом:			
		MD2	MD1	MD0	
		0	0	0	Режим «питание снято». (По умолчанию питание включено)
		0	0	1	Холостой режим. В холостом режиме фильтр АЦП и его модулятор поддерживаются в сброшенном состоянии, хотя на модулятор подаются импульсы тактовой частоты
		0	1	0	Режим однократного преобразования. В режиме однократного преобразования выполняется однократное преобразование в разрешенных каналах АЦП. По завершению цикла преобразования регистры данных АЦП (AD0L/AD0M/AD0H и/или AD1L/AD1H) модифицируются, устанавливаются соответствующие флаги регистра ADCSTAT, и происходит возврат в режим «снятое питание», причем биты MD2-MD0 сбрасываются в 000
		0	1	1	Режим циклического преобразования. В режиме циклического преобразования регистры данных АЦП постоянно модифицируются с выбранной частотой обновления выхода (см. описание регистра SF)
		1	0	1	Внутренняя калибровка верхнего предела (полной шкалы). При выполнении этой калибровки внутренний или внешний ИОН (в соответствии с состоянием битов XREF0 /XREF1 в регистрах ADC0CON/ADC1CON) автоматически подключается к входу АЦП

Окончание табл. 1.6

Бит	Имя	Описание			
		MD2	MD1	MD0	
		1	1	0	Системная калибровка «нуля». Пользователь должен подать напряжение системного «нуля» на входы каналов в соответствии с состоянием битов CH1/CH0 и ACH1/ACH0 регистров ADC0CON/ADC1CON
		1	1	1	Системная калибровка верхнего предела (полной шкалы). Пользователь должен подать напряжение системной полной шкалы на входы каналов в соответствии с состоянием битов CH1/CH0 и ACH1/ACH0 регистров ADC0CON/ADC1CON

1. Любое изменение битов MD2-MD0 немедленно сбросит оба модуля АЦП. Запись в биты MD2-MD0 без изменения их содержимого также рассматривается как сброс.
2. Если команда заносится в ADC0CON, когда ADC0EN=1, или если ADC0EN изменяется с «0» на «1», тогда оба АЦП также немедленно сбрасываются. Другими словами, основному модулю АЦП дается приоритет над дополнительным и любая команда, адресованная основному модулю АЦП немедленно отражается на дополнительном.
3. Однако, если в ADC1CON заносится команда или ADC1EN изменяется с «0» на «1», тогда сбрасывается только один модуль дополнительного АЦП. Например, если основной АЦП выполняет циклическое преобразование, когда дополнительный АЦП запускается или разрешается, то основной АЦП продолжает беспрепятственно работать, а работа дополнительного АЦП будет автоматически согласована по фазе с работой основного АЦП. В результате, время выполнения первого цикла преобразования дополнительного АЦП будет растянуто на три такта (задержано), пока выходной сигнал с дополнительного АЦП не синхронизируется с выходным сигналом основного.
4. Как только в ADCMODE записываются биты режима калибровки, биты RDY0/1 (ADCSTAT) немедленно сбрасываются и запускается цикл калибровки. По завершению цикла данные записываются в соответствующие регистры калибровки. В регистре ADCSTAT записываются соответствующие биты, а биты MD2-MD0 сбрасываются в 000, тем самым указывая, что модуль АЦП возвратился в состояние «питание снято».
5. Любая попытка калибровки модуля дополнительного АЦП игнорируется в случае, если в качестве источника входного сигнала выбран внутренний температурный датчик.
6. Калибровки выполняются при максимально возможном значении содержимого регистра SF, обеспечивая тем самым максимальную точность преобразования при выполнении калибровки.

ADC0CON (регистр управления основным АЦП)

–	XREF0	CH1	CH0	UNI0	RN2	RN1	RN0

Адрес D2h, значение после сброса 07h, битовая адресация отсутствует.

Используется для конфигурации модуля основного АЦП по диапазону измерений, выбору канала, разрешению внешнего ИОН и установке режима униполярного или биполярного преобразования. Назначение битов регистра ADC0CON указано в табл. 1.7.

Таблица 1.7

Назначение битов специального регистра ADC0CON

Бит	Имя	Описание				
7	–	Зарезервирован для дальнейшего использования				
6	XREF0	Бит выбора внешнего ИОН для модуля основного АЦП. Устанавливается пользователем для того, чтобы разрешить использование основным АЦП внешнего ИОН, подключенного к выводам REFIN(+), REFIN(–). Сбрасывается пользователем для использования основным АЦП внутреннего ИОН (VREF=1,25В)				
5 4	CH1 CH0	Биты выбора канала для модуля основного АЦП. Записываются пользователем для выбора пары дифференциальных входов, используемых основным АЦП:				
		CH1	CH0	Вход (+)	Вход (–)	
		0 0 1 1	0 1 0 1	AIN1 AIN3 AIN2 AIN3	AIN2 AIIN4 AIN2 AIN2	(Внутреннее замыкание)
3	UNI0	Бит режима униполярного преобразования модуля основного АЦП. Устанавливается пользователем для разрешения униполярного кодирования, т. е. ноль на дифференциальном входе даст код 000000h на выходе АЦП. Сбрасывается пользователем для разрешения биполярного кодирования, т. е. ноль на дифференциальном входе даст код 800000h на выходе АЦП				
2 1 0	RN2 RN1 RN0	Биты диапазона модуля основного АЦП. Устанавливаются пользователем для выбора входного диапазона АЦП:				
		RN2	RN1	RN0	Входной диапазон модуля основного АЦП (VREF=2,5В)	
		0 0 0 0 1 1 1 1	0 0 1 1 0 0 1 1	0 1 0 1 0 1 0 1	±20 мВ ±40 мВ ±80 мВ ±160 мВ ±320 мВ ±640 мВ ±1,28 В ±2,56 В	

ADC1CON (регистр управления дополнительным АЦП)

–	XREF1	ACH1	ACH0	UNI1	–	–	–

Адрес D3h, значение после сброса 00h, битовая адресация отсутствует.

Используется для конфигурации дополнительного АЦП по выбору канала, разрешению внешнего ИОН и установке режима униполярного или биполярного преобразования. Следует отметить, что дополнительный АЦП работает только при фиксированном значении опорного напряжения ±Vref. Назначение битов регистра ADC1CON указано в табл. 1.8.

Таблица 1.8

Назначение битов специального регистра ADC1CON

Бит	Имя	Описание				
7	–	Зарезервирован для дальнейшего использования				
6	XREF 1	Бит выбора внешнего ИОН для модуля дополнительного АЦП. Устанавливается пользователем для того, чтобы разрешить использование дополнительным АЦП внешнего ИОН, подключенного к выводам REFIN(+), REFIN(–). Сбрасывается пользователем для использования основным АЦП внутреннего ИОН				
5 4	ACH1 ACH0	Биты выбора канала для модуля дополнительного АЦП. Записываются пользователем для выбора однополярного входа, используемого дополнительным АЦП:				
		ACH1	ACH0	Вход (+)	Вход (–)	
		0	0	AIN3	AGND	
		0	1	AIN4	AGND	
		1	0	Температурный датчик	AGND	Внутренний датчик температуры подключается к входу АЦП
		1	1	AIN5	AGND	
3	UNI1	Бит режима униполярного преобразования модуля дополнительного АЦП. Устанавливается пользователем для разрешения униполярного кодирования, т. е. ноль на входе даст код 0000h на выходе АЦП. Сбрасывается пользователем для разрешения биполярного кодирования, т. е. ноль на входе даст код 8000h на выходе АЦП				
2	–	Зарезервирован для дальнейшего использования				
1	–	Зарезервирован для дальнейшего использования				
0	–	Зарезервирован для дальнейшего использования				

1. Когда в качестве входного сигнала выбран температурный датчик, код пользователя должен выбрать внутренний ИОН битом XREF1 и сбросить бит UNI1 для выбора биполярного кодирования.
2. Температурный датчик откалиброван на предприятии-изготовителе так, чтобы на выходе модуля дополнительного АЦП при подключенном датчике был код 8000h при 0 °С.
3. При подключенном ко входу модуля дополнительного АЦП внутреннем температурном датчике изменение температуры на 1 °С приведет к изменению значения в регистре результата преобразования АЦП ADC1H на 1 МЗР.

SF (регистр цифрового фильтра с «SINC»-характеристикой)

Адрес D4h, значение после сброса 45h, битовая адресация отсутствует.

Значение, записываемое в этот регистр, используется для установки коэффициента деления (децимации) основной частоты при установке частоты обновления выходных данных модулей основного и дополнительного АЦП. Этот регистр не может быть установлен целевой программой во время осуществления преобразования любым из модулей АЦП. Частота потока выходных данных одинакова для основного и дополнительного АЦП и рассчитывается следующим образом:

$$f\,adc = 1/3 \times 1/(8,SF) \times f\,mod,$$

где f adc – поток данных на выходе АЦП (частота модификации выхода);

f mod – опорная (тактовая) частота модулятора = 32768 кГц;

SF – десятичное значение содержимого регистра SF.

Допустимый диапазон значений SF составляет от 0Dh до FFh. Примеры значений SF и соответствующих им частот (fadc) и времен (tadc) преобразования АЦП показаны в табл. 1.9. После сброса SF принимает значение по умолчанию, равное 45h, что обеспечивает частоту модификации выходных данных около 20 Гц. При осуществлении преобразований оба модуля АЦП для минимизации ошибок смещения, используют цикл стабилизирующего прерывания.

Это означает, что время первого однократного преобразования для режима однократных преобразований АЦП или время первого цикла преобразований для циклического режима работы АЦП увеличивается в два раза и составляет 2×tadc.

Таблица 1.9

Соответствие между значениями SF и частотами преобразования АЦП

SF, dec	SF, hex	f ADC, Гц	t ADC, мс
13	0Dh	103,3	9,52
69	45h	19,79	50,34
255	FFh	5,35	186,77

Вне зависимости от текущего значения SF все циклы калибровки будут выполняться с максимальной величиной значения SF, т. е. 0FFh для достижения максимальной точности при калибровке. Как только цикл калибровки закончится, в регистре SF будет восстановлена величина, записанная туда ранее пользователем или установленная ранее по умолчанию.

ICON (регистр управления источниками тока)

–	BO	ADC1IC	ADC0IC	I2PIN	I1PIN	I2EN	I1EN

Адрес D5h, значение после сброса 00h, битовая адресация отсутствует.

Используется для управления и конфигурации включения встроенных источников тока возбуждения и контроля целостности внешней цепи датчика. Назначение битов регистра ICON указано в табл. 1.10.

Таблица 1.10

Назначение битов специального регистра ICON

Бит	Имя	Описание
7	–	Зарезервирован для дальнейшего использования.
6	BO	Бит разрешения источников тока контроля целостности внешней цепи. Устанавливается пользователем для включения обоих источников тока контроля целостности датчика во входную цепь модуля основного АЦП. Сбрасывается пользователем для выключения обоих источников тока контроля.
5	ADC1IC	Бит токовой коррекции модуля дополнительного АЦП. Устанавливается пользователем для масштабирования дополнительного АЦП с помощью слова калибровки внутреннего источника тока.
4	ADC0IC	Бит токовой коррекции модуля основного АЦП. Устанавливается пользователем для масштабирования основного АЦП с помощью слова калибровки внутреннего источника тока.
3	I2PIN	Бит выбора направления источника тока 2. Устанавливается пользователем для подачи тока от источника тока 2 (200 мкА) на внешний вывод 3 (P1.2/DAC/IEXC1). Сбрасывается пользователем для подачи тока от источника тока 2 (200 мкА) на внешний вывод 4 (P1.3/AIN5/IEXC2).
2	I1PIN	Бит выбора направления источника тока 1. Устанавливается пользователем для подачи тока от источника тока 1 (200 мкА) на внешний вывод 4 (P1.3/AIN5/IEXC2). Сбрасывается пользователем для подачи тока от источника тока 1 (200 мкА) на внешний вывод 3 (P1.2/DAC/IEXC1).
1	I2EN	Бит разрешения источника тока 2 (200 мкА). Устанавливается пользователем для включения источника тока 2 (200 мкА). Сбрасывается пользователем для выключения источника тока 2 (200 мкА).
0	I1EN	Бит разрешения источника тока 1 (200 мкА). Устанавливается пользователем для включения источника тока 1 (200 мкА). Сбрасывается пользователем для выключения источника тока 1 (200 мкА).

Ток от обоих источников можно подать на один и тот же внешний вывод, что даст величину выходного тока 400 мкА.

ADC0H/ADC0M/ADC0L (регистры данных основного АЦП)

Адреса DBh/DAh/D9h, значения после сброса 00h/00h/00h, битовая адресация отсутствует.

Эти три 8-битных регистра содержат результат 24-битного преобразования основного АЦП.

ADC1H/ADC1L(регистры данных дополнительного АЦП)

Адреса DDh/DCh, значения после сброса 00h/00h, битовая адресация отсутствует.

Эти два 8-битных регистра содержат результат 16-битного преобразования дополнительного АЦП.

OF0H/OF0M/OF0L (регистры калибровки смещения основного АЦП)

Адреса E3h/E2h/E1h, значения после сброса 80h/00h/00h, битовая адресация отсутствует.

Эти три 8-битных регистра содержат 24-битный калибровочный коэффициент смещения для модуля основного АЦП. Регистры выполнены таким образом, что при включения питания в них заносится заводской калибровочный коэффициент, по умолчанию равный 800000h. Однако, это значение будет автоматически перезаписано, если пользователем производится внутренняя или системная калибровка нуля шкалы с использованием бит MD2-MD0 специального регистра ADCMODE.

OF1H/OF1L (регистры калибровки смещения дополнительного АЦП)

Адреса E4h/E5h, значения после сброса 80h/00h, битовая адресация отсутствует.

Эти два 8-битных регистра содержат 16-битный калибровочный коэффициент смещения для модуля дополнительного АЦП. Регистры выполнены таким образом, что при включении питания в них заносится заводской калибровочный коэффициент, по умолчанию равный 8000h. Однако, это значение будут автоматически перезаписано, если пользователем проводится внутренняя или системная калибровка нуля шкалы с использованием бит MD2-MD0 специального регистра ADCMODE.

GN0H/GN0M/GNOL (регистры калибровки усиления основного АЦП)

Адреса EBh/EAh/E9h, значения после сброса индивидуальны для каждого экземпляра МК, битовая адресация отсутствует.

Эти три 8-битных регистра содержат 24-битный калибровочный коэффициент усиления для модуля основного АЦП. Регистры выполнены таким образом, что при включения питания в них заносится заводской калибровочный коэффициент полной шкалы, индивидуальный для каждого экземпляра устройства. Однако, это значение будут автоматически перезаписано, если пользователем проводится внутренняя или системная калибровка верхнего предела шкалы с использованием бит MD2-MD0 специального регистра ADCMODE.

GN1H/GN1L (регистры калибровки усиления дополнительного АЦП)

Адреса EDh/ECh, значения после сброса индивидуальны для каждого экземпляра МК, битовая адресация отсутствует.

Эти два 8-битных регистра содержат 16-битный калибровочный коэффициент усиления для модуля дополнительного АЦП. Регистры выполнены таким образом, что при включении питания в них заносится заводской калибровочный коэффициент полной шкалы, индивидуальный для каждого экземпляра устройства. Однако, это значение будут автоматически перезаписано, если пользователем проводится внутренняя или системная калибровка верхнего предела шкалы с использованием бит MD2-MD0 специального регистра ADCMODE. Регистры OF0H, OF0M, OF0L, OF1H, OF1L, GN0H, GN0M, GN0L, GN1H, GN1L программно доступны для записи данными пользователя, если биты MD0-MD2 регистра ADCMODE сброшены, т. е. при нахождении АЦП в режиме «питание снято».

1.4. Модули основного и дополнительного АЦП

МК ADuC824 имеет два независимых модуля сигма-дельта АЦП, предназначенных для измерения напряжения в области низких частот в широком динамическом диапазоне. Поскольку общая идеология построения МК такова, что модули АЦП с их аналоговой периферией являются основными компонентами устройства, а процессорное ядро и цифровая периферия – вспомогательными, то в данном изложении основное внимание будет уделяться описанию АЦП.

Модуль основного АЦП предназначен для преобразования сигналов от первичных датчиков. Входы модуля буферизованы и могут быть программно настроены на один из восьми диапазонов входных сигналов от ±20 мВ до ±2,56 В, причем измеряемые сигналы могут подаваться на одну из трех дифференциальных пар входов AIN1-AIN2, AIN3-AIN4 или AIN3-AIN2. Поскольку входы снабжены буферами, к ним можно подключать источники сигналов с большим выходным сопротивлением, а также, если потребуется, устанавливать непосредственно на входах RC-фильтры для уменьшения внешних шумов и наводок. Имеется возможность до выполнения рабочего измерения задействовать генераторы токов контроля целостности цепи внешнего первичного датчика, чтобы программно убедиться в ее исправности. Для достижения 24-битного разрешения без пропуска кодов в АЦП используется принцип сигма-дельта преобразования. Сигма-дельта модулятор преобразует входной аналоговый сигнал в последовательность цифровых импульсов, скважность которых содержит в себе информацию о величине измеряемого сигнала. В качестве встроенного узла математической обработки результатов преобразования используется программируемый низкочастотный аппаратный фильтр с характеристикой вида $(\sin(x)/x)^3$. Фильтр позволяет программно устанавливать частоту обновления данных на выходе от 5,35 Гц до 105,03 Гц. Модуль основного АЦП является 24-

разрядным, однако, одиночное преобразование не обеспечивает реального разрешения в 24 двоичных разряда, так как на входной сигнал при измерениях накладываются шумы, поэтому младшие 5–6 разрядов в результате каждого последующего преобразования будут меняться («дрожать»). Для увеличения реального разрешения производитель рекомендует использовать программную обработку (усреднение по выборке, программные фильтры со скользящим окном и т. п.) результатов преобразований. Блок-схема модуля основного АЦП приведена на рис. 1.9.

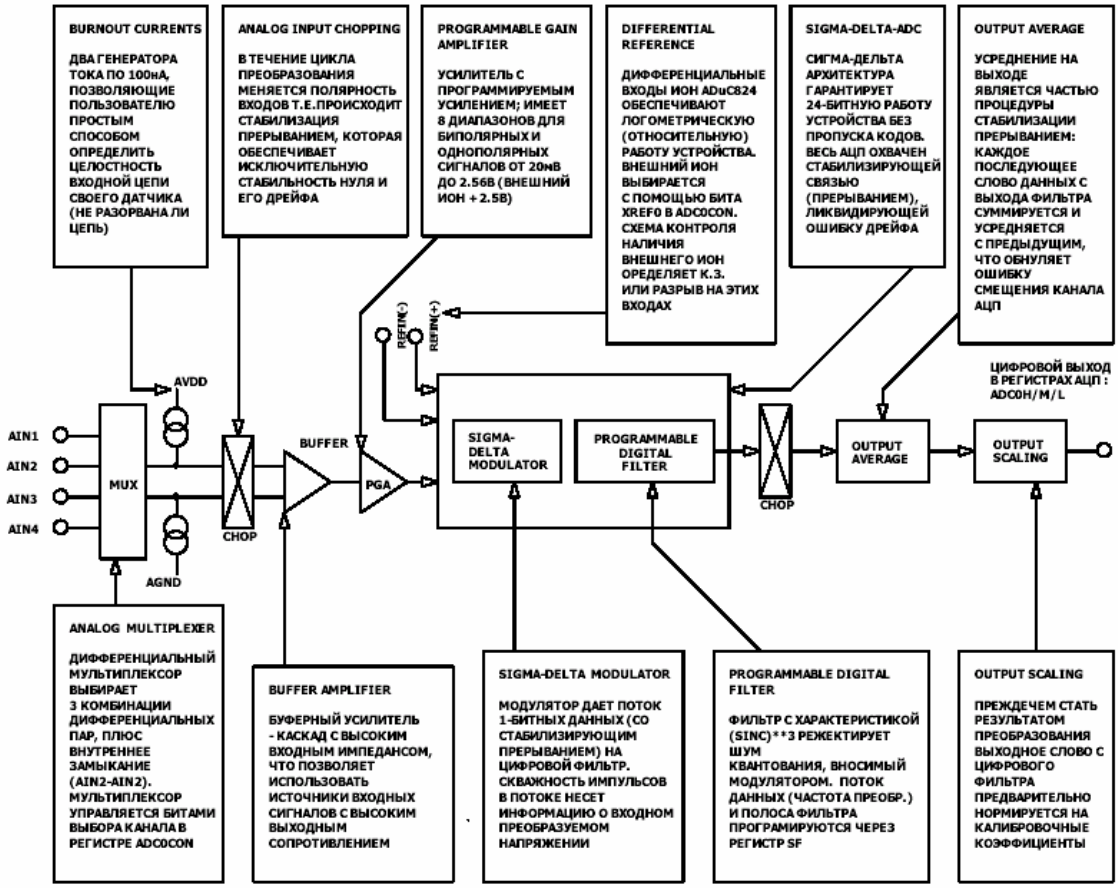


Рис. 1.9. Схема основного АЦП

Модуль дополнительного АЦП предназначен для преобразования вспомогательных аналоговых сигналов, например, напряжения источника питания или напряжения встроенного температурного датчика. В отличие от модуля основного АЦП модуль дополнительного АЦП не имеет внутреннего буфера. Значение входного напряжения должно лежать в пределах от 0 до 2,5 В. Однополярные входные сигналы могут подаваться относительно ножки AGND на один из трех входов AIN3, AIN4, AIN5 или внутри кристалла непосредственно с выхода встроенного датчика температуры. Следует заметить, что по причине отсутствия буфера эти входы дают заметную динамическую нагрузку на источник сигнала, поэтому использование источников с большим внутренним сопротивлением и

включение на входах RC-цепочек нежелательно, так как приведет к значительным ошибкам измерений. Модули основного и дополнительного АЦП являются независимыми, что дает возможность, например, запускать преобразование в одном модуле, не дожидаясь завершения длящегося преобразования в другом. Блок-схема дополнительного АЦП приведена на рис. 1.10. В табл. 1.11–1.12 приведены значения соответственно напряжения среднеквадратичного выходного шума в мкВ и разрешения «от пика до пика» на выходе АЦП в битах (с округлением до 0,5 младшего значащего разряда) для некоторых значений частоты обновления данных с выхода модуля основного АЦП.

Таблица 1.11

Типовые значения напряжения среднеквадратического шума на выходе модуля основного АЦП (мкВ)

Значение SF, dec	**Частота обновления выхода, Гц**	**Диапазон входных сигналов**							
		±20 мВ	**±40 мВ**	**±80 мВ**	**±160 мВ**	**±320 мВ**	**±640 мВ**	**±1,24 В**	**±2,56 В**
13	105,3	1,50	1,50	1,60	1,75	3,50	4,50	6,70	11,75
69	19,79	0,60	0,65	0,65	0,65	0,65	0,95	1,40	2,30
255	5,35	0,35	0,35	0,37	0,37	0,37	0,51	0,82	1,25

Таблица 1.12

Разрешение «от пика до пика» модуля основного АЦП (бит)

Значение SF, dec	**Частота обновления выхода, Гц**	**Диапазон входных сигналов**							
		±20 мВ	**±40 мВ**	**±80 мВ**	**±160 мВ**	**±320 мВ**	**±640 мВ**	**±1,24 В**	**±2,56 В**
13	105,3	12	13	14	15	15	15,5	16	16
69	19,79	13	14	15	16	17	17,5	18	18,5
255	5,35	14	15	16	17	18	18,5	18,8	19,2

В табл. 1.13–1.14 указаны аналогичные данные для модуля дополнительного АЦП. Приведенные значения являются типовыми и получены при нулевом напряжении на входе АЦП. Модуль дополнительного АЦП работает в биполярном режиме. В униполярном режиме работы дополнительного АЦП разрешение «от пика до пика» для частоты обновления выхода 105 Гц составляет 15 бит. Величина разрешения «от пика до пика» представляют собой такое разрешение, для которого в интервале «6 × СИГМА» будет отсутствовать «дрожание» кода.

Таблица 1.13

Типовые значения напряжения среднеквадратического шума на выходе модуля дополнительного АЦП (мкВ)

Значение SF, dec	**Частота обновления выхода, Гц**	**Входной диапазон 2,5 В**
13	105,3	10,75
69	19,79	2,00
255	5,35	1,15

АЦП работает в биполярном режиме.

Таблица 1.14

Разрешение «от пика до пика» модуля дополнительного АЦП (бит)

Значение SF (dec)	Частота обновления выхода, Гц	Входной диапазон 2,5 В
13	105,3	16
69	19,79	18,5
255	5,35	19,5

1. АЦП работает в биполярном режиме.
2. В униполярном режиме разрешение «от пика до пика» на частоте 105,3 Гц составляет 15 бит.

Под интервалом «6×СИГМА» здесь понимается интервал времени, количественно характеризующий работу сигма-дельта модулятора АЦП. С модулем основного АЦП связаны четыре входных линии (AIN1 – AIN4), которые можно программно сконфигурировать как два независимых дифференциальных канала. Биты выбора канала находятся в специальном регистре ADC0CON и показаны в табл. 1.7. Путем установки различных комбинаций этих битов можно выбрать три варианта конфигурации дифференциального канала, а также дополнительный вариант короткозамкнутой петли внутри устройства (AIN2-AIN2). Выбранная дифференциальная пара мультиплексируется на вход внутреннего буферного усилителя. Модуль дополнительного АЦП имеет три входных линии (AIN3 – AIN5), а также внутреннее соединение с встроенным датчиком температуры. Все входы дополнительного АЦП являются однополярными относительно аналогового общего провода (AGND) устройства. Биты выбора канала модуля дополнительного АЦП находятся в регистре ADC1CON и показаны в табл. 1.8. Выбранная входная линия мультиплексируется непосредственно на вход сигма-дельта модулятора. Для осуществления корректного преобразования с гарантией получения достоверного результата следует после переключения входных линий произвести программную задержку на время установления напряжения на входе АЦП. Для модуля основного АЦП значения входных напряжений могут находиться в пределах от уровня AGND+100 мВ до уровня AVDD –100 мВ. В случае выхода входного измеряемого напряжения за эти значения достоверность результата преобразования не гарантируется. В частности, не следует заземлять одну из линий в дифференциальной входе модуля. Для модуля дополнительного АЦП значения входных напряжений могут находиться в пределах от AGND –30 мВ до AVDD +30 мВ. Небольшая отрицательная величина нижнего предела входного напряжения дает возможность измерять значения малых биполярных сигналов.

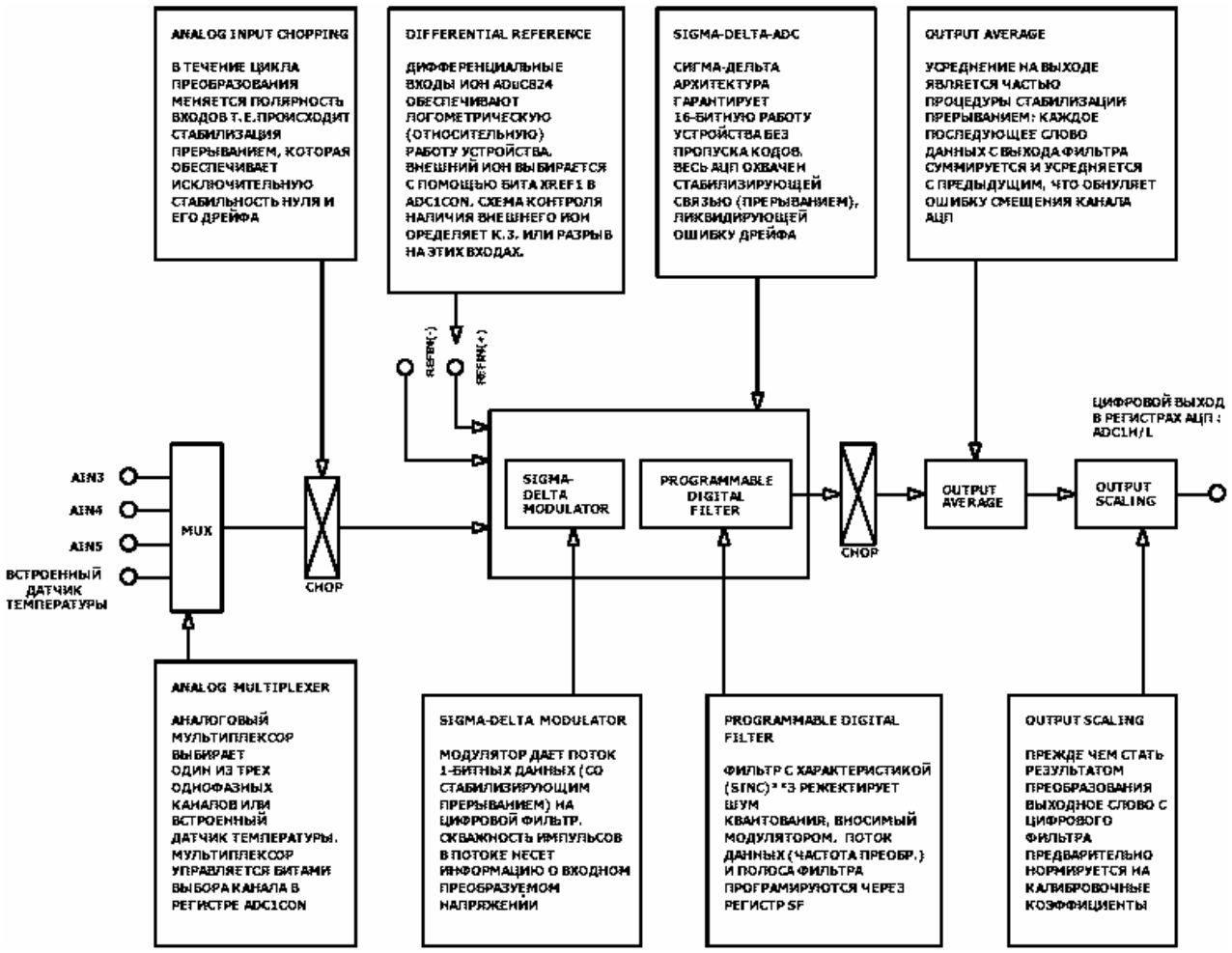


Рис. 1.10. Схема дополнительного АЦП

Выход буфера основного АЦП подключен к входу встроенного усилителя с программируемым коэффициентом усиления (PGA). Коэффициент усиления PGA может быть программно выбран из восьми возможных значений. Выбор осуществляется путем установки комбинаций бит выбора диапазона в специальном регистре ADC0CON. При установленном бите разрешения внешнего ИОН и величине его напряжения 2,5 В для входного однополярного напряжения можно получить следующие диапазоны измерений: +0...20 мВ, +0...40 мВ, +0...80 мВ, +0...160 мВ, +0...320 мВ, +0...640 мВ, +0...1,28 В, +0...2,56 В. Для входного биполярного напряжения имеются следующие диапазоны измерений: ±20 мВ, ±40 мВ, ±80 мВ, ±160 мВ, ±320 мВ, ±640 мВ, ±1,28 В, ±2,56 В. В составе модуля дополнительного АЦП нет PGA, и он работает в единственном диапазоне входных напряжений 0 В – VREF, где VREF – напряжение ИОН. Утверждение, что напряжение на входе АЦП может быть биполярным, применительно к модулю основного АЦП означает, что данное напряжение может принимать положительное или отрицательное значение относительно напряжения на другом проводе дифференциального входа, но не относительно ножки аналоговой «земли» AGND. Например, если на вход AIN(–) подано напряжение 2,5 В, а модуль основного АЦП сконфигурирован для работы с однополярным

напряжением, в диапазоне 0 мВ – +20 мВ, то допустимое напряжение на входе AIN(+) будет лежать в диапазоне 2,5–2,52 В. Если на вход AIN(–) подано напряжение 2,5 В, а модуль основного АЦП сконфигурирован для работы с биполярным напряжением в диапазоне ±1,28 В, то допустимое напряжение на входе AIN(+) будет лежать в диапазоне 1,22–3,78 В (т. е. 2,5 В ±1,28 В). Для модуля дополнительного АЦП, как уже было указано, напряжение на входе может быть отрицательным относительно AGND, но при этом разность не должна превышать 30 мВ. Режимы биполярной или однополярной работы модулей основного и дополнительного АЦП выбираются с помощью соответствующих битов в специальных регистрах ADC0CON и ADC1CON. Задание однополярного или биполярного режима работы АЦП никак не меняет требований к входным сигналам, а только меняет способ их внутренней кодировки, т. е. вид передаточной функции, по которой выполняется кодировка. Когда АЦП настроен на однополярную работу, кодировка имеет естественный двоичный характер: при нулевом дифференциальном входном напряжении выходной код будет равен 000…000, при напряжении, равном половине диапазона, код будет равен 100…000, а при напряжении, равном верхней границе диапазона, – 111…111. Когда АЦП настроен на работу с биполярными сигналами, кодировка становится двоичной смещенной: для входного отрицательного напряжения, равного нижней границе диапазона, выходной код будет равен 000…000, нулевое дифференциальное напряжение на входе даст код 100…000, а положительное напряжение, равное верхней границе диапазона, – код 111…111.

Если после осуществления «внутренней» калибровки «нуля» в целевой программе производилась смена диапазона измерений, то нет необходимости заново производить калибровку, так как получившееся рассогласование в соответствии со спецификацией фирмы-производителя не превысит 2 мкВ. Проиллюстрировать это утверждение можно сигнальной диаграммой, изображенной на рис. 1.11. Кривая на рисунке представляет собой результат измерений модулем основного АЦП входного постоянного напряжения, примерно равного 19 мВ. Ломаный вид кривой обусловлен периодичностью обновления данных на выходе АЦП. В данном примере модуль основного АЦП работает в биполярном режиме, имеется внешний ИОН с напряжением 2,5 В АЦП непрерывно преобразует входное напряжение с частотой обновления данных на выходе 5,35 Гц. Первые 100 отсчетов выдаются при установленном диапазоне ±20 мВ. Затем диапазон переключается на ±40 мВ и выдаются еще 100 отсчетов, и так далее до последнего диапазона ±2,56 В. Калибровка после переключений не производится. Как можно видеть из рисунка, разница между математическими ожиданиями (средними значениями по выборкам) в любых двух диапазонах не превышает 2 мкВ.

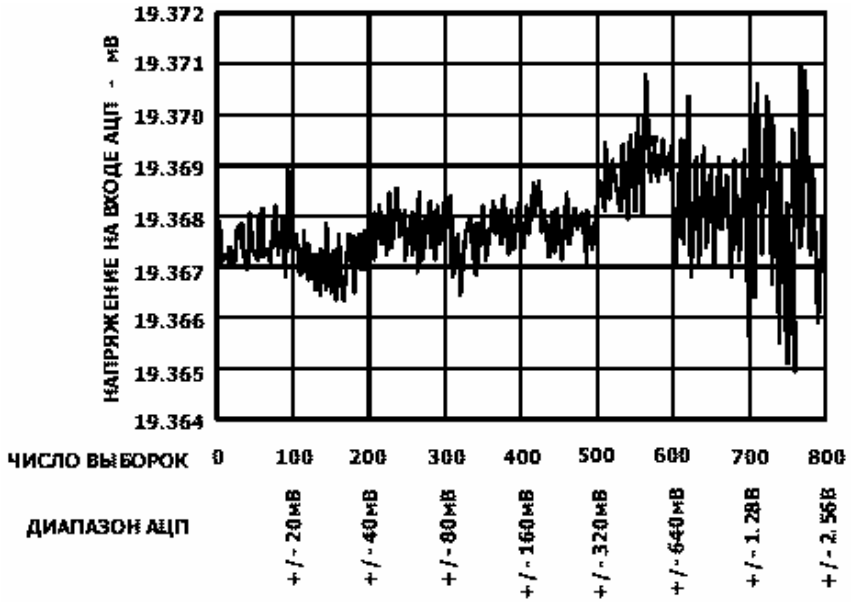


Рис. 1.11. Результаты измерения калиброванным АЦП

Для «системной» калибровки «нуля» ситуация с рассогласованием по усилению при смене диапазонов будет несколько иная. Пусть, например, при установленном диапазоне 20 мВ и имеющемся внешнем смещении 10 мкВ была произведена «системная» калибровка «нуля», в результате которой в регистры калибровки смещения OF0H, OF0M, OF0L записалось число 802100h, «учитывающее» внешнее смещение 10 мкВ. Затем диапазон был сменен на значение 40 мВ без проведения повторной «системной» калибровки «нуля». Теперь число 802100h, оставшееся неизменным в регистрах OF0H, OF0M, OF0L, будет соответствовать значению внешнего смещения 20 мкВ, так как одному биту кода теперь соответствует вдвое большее значение напряжения. Для диапазона 2,56 В число 802100h будет соответствовать значению смещения 1,28 мВ.

Производитель МК предлагает два способа устранения описанного рассогласования. Первый заключается в том, чтобы предварительно произвести «системную» калибровку «нуля» во всех диапазонах, использование которых предполагается в данном приложении, и последовательно скопировать полученные значения из регистров OF0H, OF0M, OF0L во Flash-память данных МК. При переключении на очередной диапазон соответствующее ему сохраненное число следует программно переписывать из Flash-памяти в регистры OF0H, OF0M, OF0L.

Второй способ заключается в следующем. Предварительно производится «системная» калибровка «нуля» в диапазоне ±20 мВ. После этого при каждом изменении диапазона необходимо программно корректировать содержимое регистров OF0H, OF0M, OF0L таким образом, чтобы разность между предыдущим содержимым OF0H, OF0M, OF0L и числом 800000h уменьшалась вдвое при увеличении диапазона на одну ступень. Сказанное поясняется следующей формулой:

$$(OF0H, OF0M, OF0L)\ new\_range = 800000h + ((OF0H, OF0M, OF0L)20mV\_range - 800000h) / 2^{RN},$$

где (OF0H, OF0M, OF0L) new_range – содержимое регистров OF0H, OF0M, OF0L, соответствующее новому диапазону;

(OF0H, OF0M,OF0L)20mV_range – содержимое регистров OF0H, OF0M, OF0L, соответствующее диапазону 20 мВ;

RN – показатель степени, равный порядковому номеру диапазона (0 для 20 мВ, 1 для 40 мВ, …).

Модуль основного АЦП содержит два встроенных генератора постоянного тока 100 нА, один из которых обеспечивает протекание тока в направлении от AVDD к AIN(+), а другой – в направлении от AIN(–) к AGND внутри устройства. Источники программно подключаются к выбранной паре аналоговых входов. Оба источника можно одновременно либо включить, либо выключить, соответственно установив или сбросив бит разрешения тока контроля ВО в специальном регистре ICON (табл. 1.10). Токи, вырабатываемые этими источниками, можно использовать для проверки целостности цепи внешнего датчика до проведения рабочего измерения в данном канале. При включении источников токи от них начинают протекать по цепи внешнего датчика, обеспечивая некоторое падение напряжения на ней. Если это напряжение, будучи измеренным, окажется равным максимуму шкалы в данном диапазоне, то такой результат укажет на разрыв в цепи датчика. Если измеренное напряжение будет близко к нулю, значит цепь датчика короткозамкнута. Зная, каким должно быть сопротивление исправной цепи датчика, пользователь может, таким образом, программно организовать анализ и диагностику ее состояния. При проведении рабочих измерений источники следует отключить. Применение источников не накладывает дополнительных ограничений на значение входных напряжений модуля АЦП.

В составе МК имеются также два одинаковых встроенных источника стабильного постоянного тока номиналом по 200 мкА. Они обеспечивают протекание тока от AVDD на ножку IEXC1 или ножку IEXC2 внутри устройства. Источники управляются битами специального регистра ICON, как показано в табл. 1.10. Их можно сконфигурировать как отдельные с токами по 200 мкА, протекающими на две соответствующих ножки микросхемы, либо объединить в один источник с током 400 мкА, протекающим на любую из названных ножек. Токи, генерируемые этими источниками, можно использовать для возбуждения внешних резистивных датчиков.

Для подключения внешнего источника опорного напряжения МК имеет соответствующие ножки REFIN(+) и REFIN(–). Входы REFIN(+) и REFIN(–) являются дифференциальными. Диапазон синфазных сигналов на этих входах лежит в пределах от AGND до AVDD. Номинальное рабочее напряжение на входах REFIN(+) и REFIN(–) составляет 2,5 В. Именно для этого значения нормируются параметры модулей АЦП устройства. Минимальное рекомендуемое

опорное напряжение на входах REFIN(+) и REFIN(–) составляет 1 В, максимальное – AVDD. В случае использования несимметричного источника опорного напряжения ножка REFIN(–) может быть подключена к аналоговой «земле» AGND.

Для разрешения внешнего ИОН следует установить биты XREF0, XREF1 в специальных регистрах ADC0CON и ADC1CON модулей основного и дополнительного АЦП соответственно.

В случае, если внешний ИОН отсутствует, МК может использовать внутренний ИОН, для чего следует сбросить биты XREF0, XREF1. Поскольку значения границ диапазонов входных напряжений нормированы для опорного напряжения 2,5 В, а у внутреннего ИОН оно составляет 1,25 В, то границы всех диапазонов уменьшаются ровно в два раза. В результате, при использовании внутреннего ИОН произойдет заметное ухудшение разрешения АЦП «от пика до пика», которое в этом случае будет составлять всего 13–14 двоичных разрядов. По этой причине для получения наилучшего разрешения производитель настоятельно рекомендует использовать внешний ИОН. Внутренний ИОН часто применяется с модулем дополнительного АЦП, кроме того, он необходим для работы встроенного температурного датчика.

В приложениях, где для создания внешнего опорного напряжения МК и возбуждения внешнего резистивного датчика используется один и тот же источник (логометрический принцип измерения), влияние на точность измерений низкочастотного шума источника будет автоматически скомпенсировано, поскольку в этом случае форма шумовой составляющей совпадает во входном и опорном напряжениях. Если же МК применяется не в логометрическом приложении, следует использовать ИОН с низким уровнем шумов. Производитель рекомендует применение в качестве ИОН микросхем AD780, REF43 и REF192.

Следует отметить, что входы опорного напряжения REFIN(+) и REFIN(–) являются для внешнего ИОН высокоимпендансной динамической нагрузкой. Поскольку входное сопротивление каждого опорного входа имеет динамический характер, подключение к такому входу RC-цепочки может привести к ошибке в измерениях, которая будет тем заметнее, чем больше внутреннее сопротивление ИОН. Рекомендованные ИОН имеют малое внутреннее сопротивление, поэтому в случае их использования включение между REFIN(+) и REFIN(–) фильтрующих конденсаторов является допустимым. Если же внешнее опорное напряжение подается на МК с резистивного делителя со значительным выходным импедансом, установка фильтрующих конденсаторов между выводами REFIN(+) и REFIN(–) не рекомендуется.

В составе МК имеется встроенная схема, позволяющая программно определить, имеется ли на выводах подключения внешнего ИОН напряжение необходимой величины для осуществления преобразований и калибровки. Если напряжение между ножками REFIN(+) и REFIN(–) меньше 0,3 В или эти ножки ни к чему не подключены, схема определяет факт отсутствия в системе опорного напряжения и производит аппаратную установку бита NOXREF в специаль-

ном регистре ADCSTAT. В случае, если диагностирование факта отсутствия опорного напряжения и, соответственно, установка бита NOXREF происходит во время измерительного (рабочего) преобразования, результат преобразования возвращается в виде слова, состоящего из одних единиц. Таким образом, при выполнении преобразования нет необходимости постоянно программно следить за текущим состоянием бита NOXREF. Целевая программа должна проверить состояние этого бита только в том случае, если результат преобразования в регистре данных АЦП будет состоять из одних единиц.

Если установка бита NOXREF происходит во время цикла калибровки нуля или усиления основного или дополнительного модулей АЦП, то запись в соответствующие регистры калибровки автоматически запрещается, т.е., калибровка в этом случае не производится. Одновременно устанавливаются биты ошибки модуля АЦП ERR0 или ERR1 соответственно в специальном регистре ADCSTAT. Для обеспечения гарантии корректной калибровки целевая программа должна проверять биты ERR0 и (или) ERR1 всякий раз по окончании цикла калибровки.

Сигма–дельта АЦП МК состоит из аналогового модулятора и цифрового фильтра. Аналоговый модулятор включает в себя разностный (дифференциальный) усилитель, интегратор, компаратор и ЦАП сигнала обратной связи, как показано на рис. 1.12. При работе аналогового модулятора выборки аналогового сигнала и сигнала с выхода ЦАП поступают на входы дифференциального усилителя. Разность этих двух сигналов интегрируется и подается на вход компаратора. Сигнал с выхода компаратора поступает на вход ЦАП и, таким образом, замыкается петля отрицательной обратной связи. Информация о величине входного аналогового сигнала содержится в скважности импульсной последовательности, формируемой на выходе компаратора. Преобразование скважности импульсов в цифровой код осуществляется в цифровом фильтре. Частота выборок модулятора (частота модуляции) во много раз больше частотной полосы входного сигнала, а цифровой фильтр ограничивает полосу выходного сигнала до величины, существенно ниже половинной частоты модуляции. Таким образом, подавляется генерируемый интегратором модулятора шум квантования, спектр которого лежит в области, большей половинной частоты модуляции.

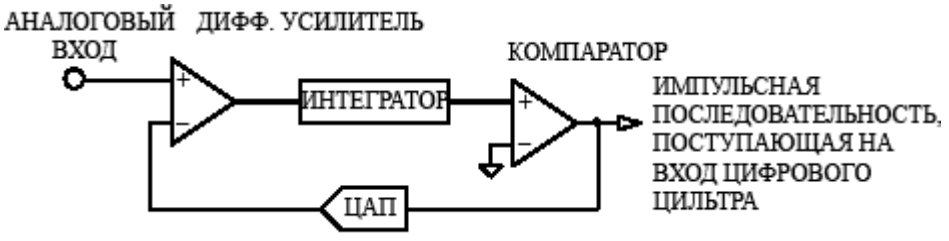


Рис. 1.12. Структура АЦП

Фильтр МК имеет функцию передачи вида $(\sin(x)/x)^3$. Частота среза фильтра и частота поступления данных с его выхода устанавливаются программно с помощью регистра SF, как показано в табл. 1.9.

На рис. 1.13 показан АЧХ фильтра при значении SF по умолчанию, равном 45h. Частота обновления выходных данных для этого значения составляет чуть меньше 20 Гц. Из рисунка можно видеть, что подавление частотных составляющих входного сигнала, совпадающих с частотами переменного напряжения промышленных сетей 50 и 60 Гц, обеспечивается фильтром на уровне >65 и >100 дБ соответственно. Значение этого ослабления принято называть ослаблением помехи нормального вида. На рис. 1.14 показан вид АЧХ фильтра при максимальном значении SF, равном 0FFh. В этом случае ослабление помехи нормального вида для частот 50 и 60 Гц составляет соответственно > 90 и >70 дБ. На рис. 1.15 показана графическая зависимость ослабления помехи нормального вида для частоты входного сигнала 50 Гц от десятичного содержимого регистра SF, а на рис. 1.16 – аналогичная зависимость для частоты 60 Гц. Пользуясь графиком на рис. 1.15, можно, например, определить, что наименьшее влияние на точность измерений сетевых наводок с частотой 50 Гц будет обеспечиваться при десятичных значениях SF, равных 71, 166 и 245.

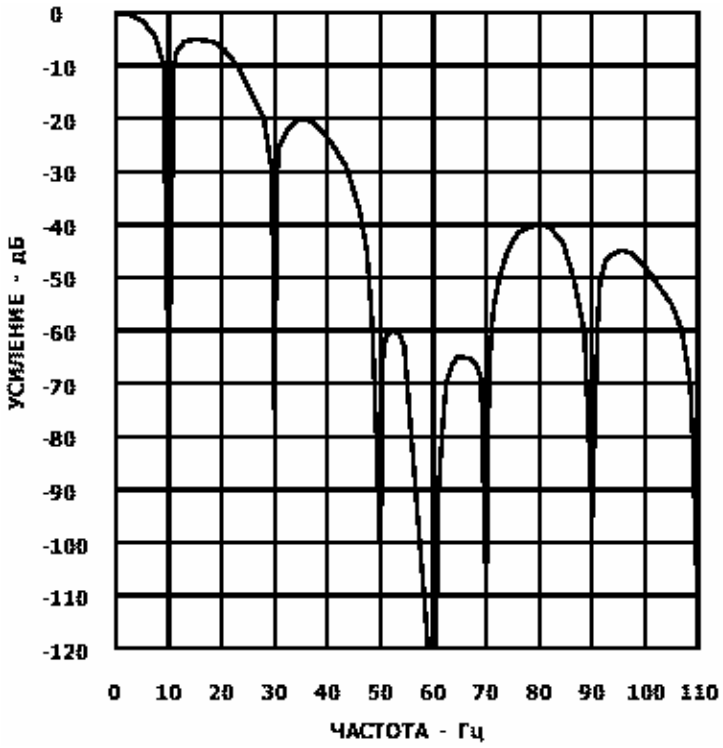


Рис. 1.13. АЧХ фильтра при $SF=(69)_{10}$

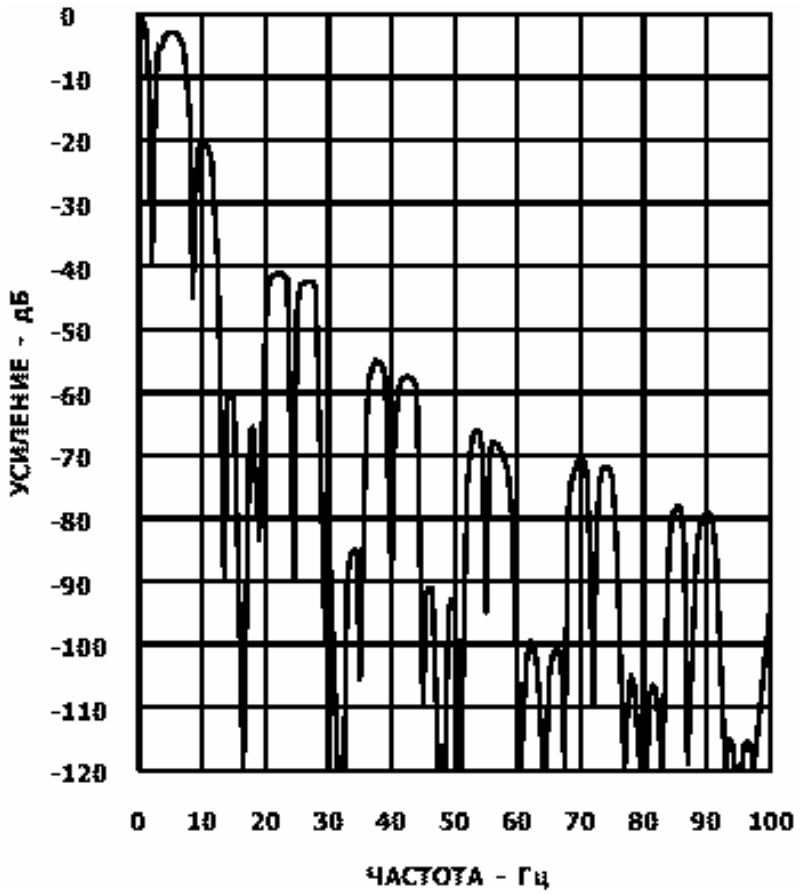


Рис. 1.14. АЧХ фильтра при $SF=(255)_{10}$

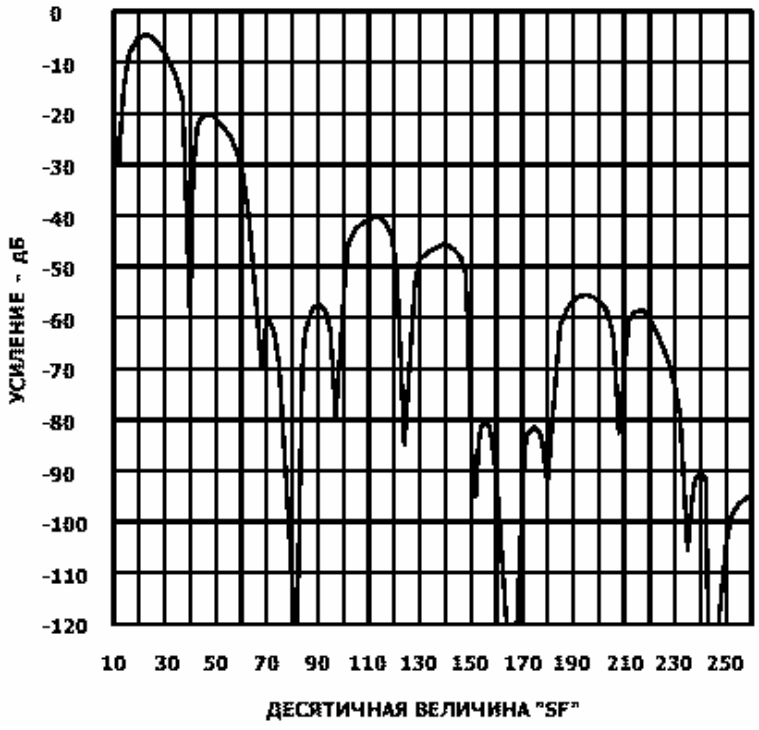


Рис. 1.15. АЧХ фильтра при F=50 Гц

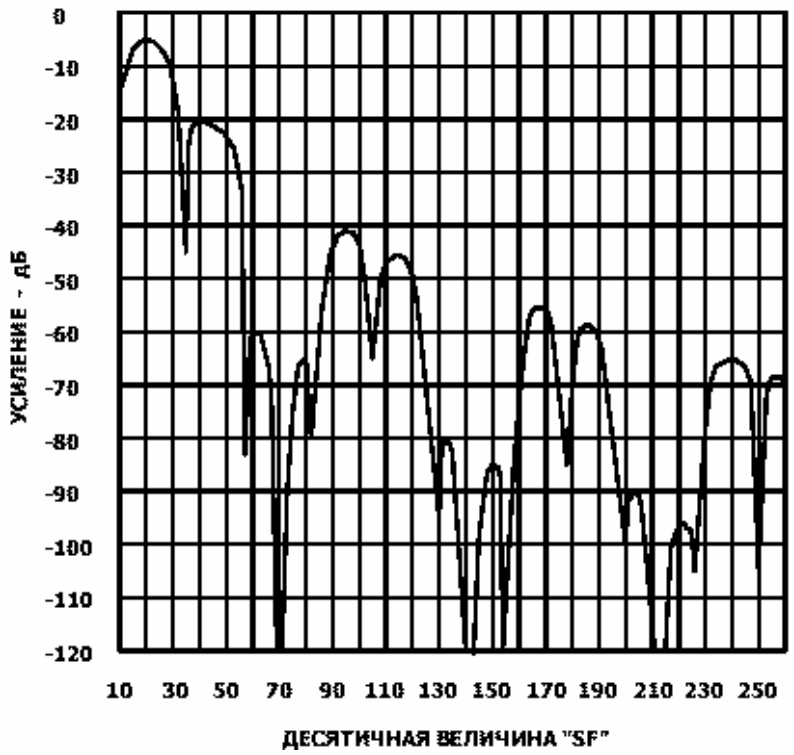


Рис. 1.16. АЧХ фильтра при F=60 Гц

Оба модуля АЦП при измерениях используют способ стабилизации прерыванием, суть которого заключается в периодической внутренней автоматической переполюсовке (смене местами) линий в дифференциальной паре. По этой причине цифровые слова данных на выходе цифрового фильтра помимо составляющей входного сигнала содержат паразитные составляющие положительного и отрицательного смещения. Чтобы исключить из окончательного результата эти составляющие, в состав каждого модуля АЦП входит оконечный суммирующий каскад, производящий суммирование и усреднение каждого последующего выходного слова с предыдущим. Результат этого усреднения записывается в регистры данных модуля АЦП и, таким образом, является окончательным. При описанном режиме работы время полного цикла преобразования (интервал времени, необходимый для получения первого результата), как уже отмечалось выше, составит 2 × tadc.

Модули основного и дополнительного АЦП обеспечивают проведение четырех видов калибровки, запустить которые можно с помощью битов MD0 – MD2 специального регистра ADCMODE, как показано в табл. 1.6. Вообще говоря, каждый конкретный экземпляр МК уже полностью откалиброван на предприятии-изготовителе. Заводская калибровка производится при напряжении питания +5 В и окружающей температуре +25 °С. Полученные в результате калибровочные коэффициенты смещения и усиления для модулей основного и дополнительного АЦП размещаются в специальной внутренней технологической Flash-памяти. При включении питания эти заводские коэффициенты автоматически загружаются в регистры калибровки МК, которые были описаны

выше. В случае, если целевая программа не производит собственную калибровку, то при проведении рабочих преобразований используются заводские калибровочные коэффициенты. Такой способ организации измерений может быть рекомендован, когда условия работы устройства по напряжению питания и температуре не отличаются от условий проведения заводской калибровки. В случае, если предполагается производить измерения при других условиях, рекомендуется осуществить калибровку в целевой программе. Если программа производит для данного модуля АЦП один из четырех видов калибровки, содержимое калибровочных регистров будет перезаписано полученными в результате пользовательскими калибровочными коэффициентами.

Для проведения полной калибровки выбранного модуля АЦП внутренней логике требуется зафиксировать значения выходного сигнала модулятора для двух уровней входного сигнала: «ноль» и «верхний предел диапазона». Результат калибровочного преобразования уровня «нуля» записывается в регистры калибровки смещения. Результат калибровочного преобразования уровня «верхнего предела диапазона» записывается в регистры калибровки усиления. Только при наличии этих величин логика калибровки может рассчитать смещение и передаточный коэффициент функции передачи конвертера со входа на выход.

МК обеспечивает возможность проведения «внутренней» и «системной» калибровок. Во время «внутренней» калибровки «нуля» или «верхнего предела диапазона» соответствующие напряжения «нуля» или «верхнего предела» автоматически подаются на вход внутри устройства. При «системной» калибровке предполагается, что напряжения «системного нуля» и «системного верхнего предела» будут поданы пользователем на внешние выводы АЦП до того, как соответствующая калибровка будет запущена. «Системная» калибровка позволяет учесть и минимизировать ошибки измерения, вносимые внешними цепями измерительного тракта устройства. Для увеличения точности преобразований калибровка во всех режимах автоматически выполняется с минимальным значением частоты обновления данных на выходе цифрового фильтра. Таким образом, каждый полный цикл калибровки с учетом удвоенного времени преобразования занимает 0,374 с.

Хранящийся в регистрах калибровки смещения калибровочный коэффициент смещения вычитается из результата преобразования до его умножения на значение передаточного коэффициента, полученное из калибровочного коэффициента усиления. Фактические значения характеристик АЦП будут соответствовать значениям, заявленным производителем (табл. 1.1), только после проведения «внутренней» калибровки «нуля» и «верхнего предела диапазона» при текущих рабочих условиях (напряжении питания и температуре).

С точки зрения функционирования модулей АЦП калибровку следует рассматривать как обычное преобразование. Калибровку «нуля», если таковая требуется, следует всегда проводить до калибровки «верхнего предела диапазона».

Для определения момента окончания калибровки или окончания рабочего преобразования Целевая программа должна следить за состоянием битов RDY0 и RDY1 специального регистра ADCSTAT путем их программного опроса или путем обработки прерываний по завершению преобразования.

1.5. Встроенная Flash/EE-память

МК имеет встроенную многократно программируемую энергонезависимую Flash-память данных и программ. Архитектура Flash-памяти имеет в основе архитектуру однотранзисторной ячейки. Подобно EEPROM, Flash-память можно программировать в составе системы побайтно, хотя предварительно ее необходимо стереть; причем процесс стирания выполняется поблочно. Исходя из этих соображений, производитель в спецификациях на изделие именует Flash-память МК Flash/EE-памятью (электрически стираемой Flash-памятью). Для целевых приложений МК имеет два массива Flash/EE-памяти: область памяти программ и область памяти данных. Массив Flash/EE-памяти программ имеет размер 8 кбайт. Его можно программировать в параллельном режиме, используя стандартные программаторы сторонних производителей, а также в составе системы, используя встроенный режим последовательной загрузки. Массив Flash/EE-памяти данных имеет размер 640 байт. Эта область памяти доступна для программных записи и чтения и может быть задействована в целевой программе как память данных общего применения. Доступ пользователя к области Flash/EE-памяти данных осуществляется через группу из шести специальных регистров, которые будут рассмотрены ниже. Область Flash/EE-памяти данных можно программно модифицировать побайтно, хотя предварительно ее следует стереть постранично (одна страница содержит четыре байта).

При эксплуатации МК в индустриальном диапазоне температур –40... +85 °С производитель гарантирует надежность Flash/EE-памяти устройства, которая количественно определяется как минимальное число циклов безошибочной записи-стирания, – 100 000 циклов, а при эксплуатации в типовых условиях (при температуре +25 °С) – 700 000 циклов. Под циклом записи-стирания в данном случае понимается следующая последовательность операций с памятью: стирание страницы, чтение-верификация страницы, программирование (запись) байта в странице, чтение-верификация страницы.

Сохранность данных в памяти МК количественно характеризуется числом лет, в течение которых записанные в память данные сохраняются в ней неизменными. Производитель гарантирует сохранность данных в памяти МК при температуре перехода (очевидно, имеется ввиду температура перехода транзистора ячейки памяти) Т=+55 °С в течение 100 лет.

Указанное значение допустимого числа циклов стирания-записи было получено производителем при тестировании устройства в соответствии с доку-

ментом «JEDEC Specification A117», а характеристики надежности – при тестировании в соответствии с документом «JEDEC Retention Life-Time Specification A117».

Flash/EE-память программ занимает младшие 8 кбайт из 64 кбайт адресуемой программной памяти. Этот массив можно программировать в одном из двух режимов:

1. Режим последовательной загрузки с помощью встроенной в устройство программы-загрузчика через стандартный последовательный порт (UART). Режим последовательной загрузки включается автоматически при подаче питания, если ножка PSEN/ подключена через внешний резистор сопротивлением 1 кОм к общему проводу, как показано на рис. 1.17. Переведя МК в этот режим, можно загрузить код в массив памяти программ внутрисхемно, без извлечения МК из устройства пользователя. Исполняемая программа WSD, обеспечивающая последовательную загрузку памяти МК с персонального компьютера, поставляется как часть системы разработки QuickStart ADuC824 и подробно описана в следующей части этого учебного пособия. Описание протокола последовательной загрузки приводится в [4].

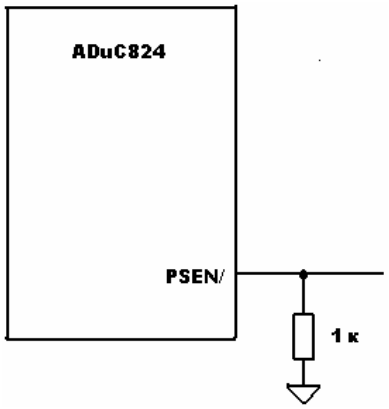


Рис. 1.17. Схема для перевода МК в режим последовательной загрузки

2. Режим параллельного программирования, поддерживаемый стандартными программаторами EEPROM/ Flash-памяти сторонних производителей. Функциональная схема внешних соединений для режима параллельного программирования МК ADuC824 показана на рис. 1.18. В этом режиме порты P0, P1 и P2 используются в качестве внешней интерфейсной магистрали адреса и данных, сигнал на линии ALE является стробом разрешения записи, а порт P3 используется как порт общего назначения, сигналы на линиях которого устанавливает устройство в различные подрежимы стирания и программирования в процессе выполнения процедуры параллельного программирования.

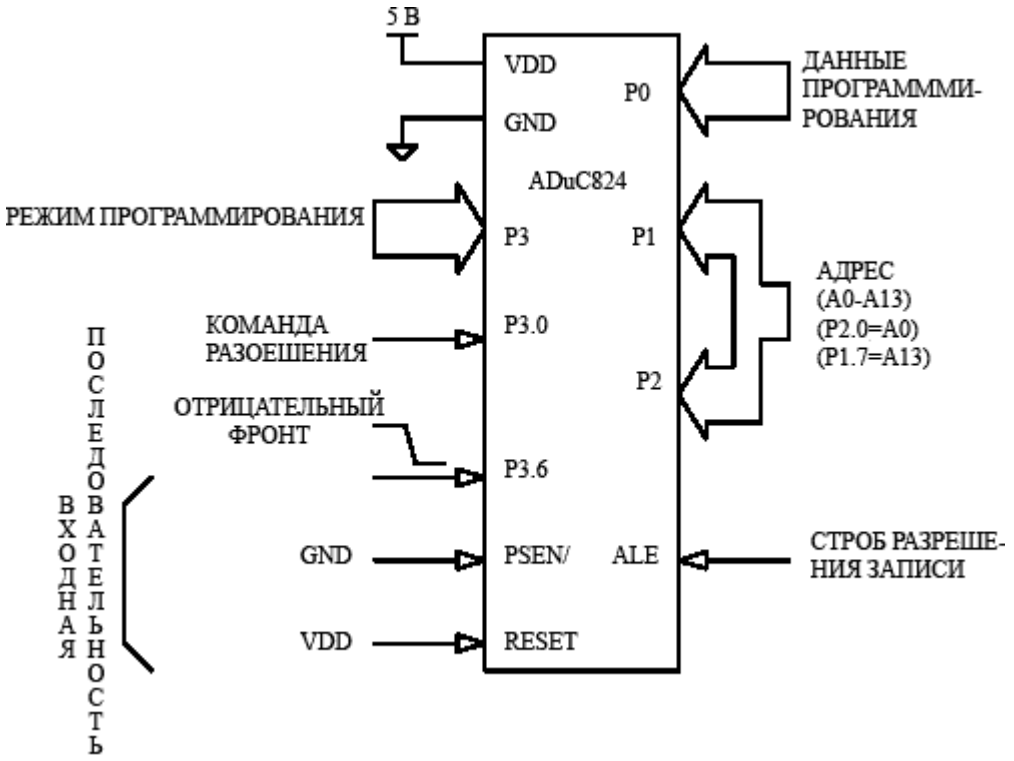


Рис. 1.18. Параллельное программирование

Источник напряжения +12В, необходимый для программирования Flash/EE-памяти, встроен в устройство.

Режимы параллельного программирования, которые доступны путем установки комбинаций битов порта P3, показаны в табл. 1.15.

Таблица 1.15

Режимы параллельного программирования Flash/EE-памяти

Линии порта P3							**Режим программирования**
0.7	0.6	0.5	0.4	0.3	0.2	0.1	
X	X	X	X	0	0	0	Стирание Flash/EE-памяти программ, данных и битов защиты
X	X	X	X	0	0	1	Чтение сигнатуры/идентификатора устройства
X	X	X	0	0	1	0	Программирование байта данных
X	X	X	1	0	1	0	Программирование байта кода
X	X	X	0	0	1	1	Чтение байта данных
X	X	X	1	0	1	1	Чтение байта кода
X	X	X	X	1	0	0	Программирование битов защиты
X	X	X	X	1	0	1	Чтение/проверка битов защиты
Все оставшиеся коды							Резервные

МК имеет три возможных режима защиты Flash/EE памяти программ. Любой из них можно активировать независимо от других, в различной степени ограничив тем самым доступ к содержимому встроенной памяти программ. Установка режимов защиты производится при программировании МК в рамках пользовательского интерфейса программы последовательного загрузчика WSD.

Режим запирания (Lock Mode)

Этот режим «запирает» код в программной памяти, запрещая выполнение ее параллельного программирования, и разрешает ее параллельное считывание. Режим запирания снимается в момент выполнения команды «стереть код» в режимах последовательной загрузки или параллельного программирования.

Режим защиты (Secure Mode)

Этот режим «запирает» код в программной памяти, запрещая параллельное программирование, в частности, выполнение команд верификации-чтения, а также выполнение инструкции MOVC, если она считывается из внешней программной памяти. Последнее является попыткой считать исполняемый код из внутренней программной памяти. Как и предыдущий, этот режим также снимается в момент выполнения команды «стереть код» в режимах последовательной загрузки или параллельного программирования.

Режим защиты от последовательной загрузки (Serial Safe Mode)

Этот режим запрещает выполнение последовательной загрузки устройства. Если при установленном режиме защиты от последовательной загрузки производится попытка перевести устройство в режим последовательной загрузки путем установки сигнала RESET при низком уровне на ножке PSEN/, МК будет интерпретировать это как обычный сброс. Устройство не перейдет в режим последовательной загрузки, а лишь выполнит последовательность сброса. Режим защиты от последовательной защиты может быть снят только в момент выполнения команды «стереть код» в режиме параллельного программирования.

1.6. Операции с Flash/EE-памятью данных

Массив Flash/EE-памяти данных на кристалле имеет размер 640 байт и организован в виде ста шестидесяти 4-битных страниц с адресами от 00h до 9Fh (рис. 1.19). Доступ из целевой программы к Flash/EE-памяти данных осуществляется через обслуживающие ее специальные регистры. Для хранения содержимого 4-х байтовой страницы памяти, к которой производилось последнее по времени обращение, используется четыре специальных регистра данных EDATA1 – EDATA4. Для хранения 8-битного адреса страницы, к которой производится обращение, служит специальный регистр EADRL. Для управления доступом используется специальный регистр управления ECON, в который можно записать одну из пяти имеющихся команд, реализующих следующие операции с Flash/EE-памятью данных: чтение, запись, стирание и верификация. Блок-схема регистрового интерфейса массива Flash/EE-памяти данных показана на рис. 1.20.

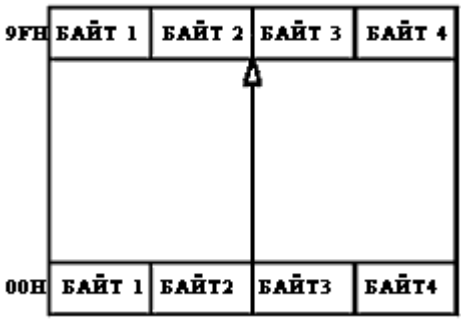


Рис. 1.19. Организация Flash/EE-памяти данных

ECON (регистр управления Flash/EE-памятью данных)

Адрес B9h, значение после сброса 00h, битовая адресация отсутствует.

Этот регистр управляет доступом к пространству Flash/EE-памяти данных, являясь интерпретатором команд. В него можно записать одну из пяти имеющихся команд запуска циклов чтения, программирования или стирания, управляющие слова (коды) и описания которых приведены в табл. 1.16.

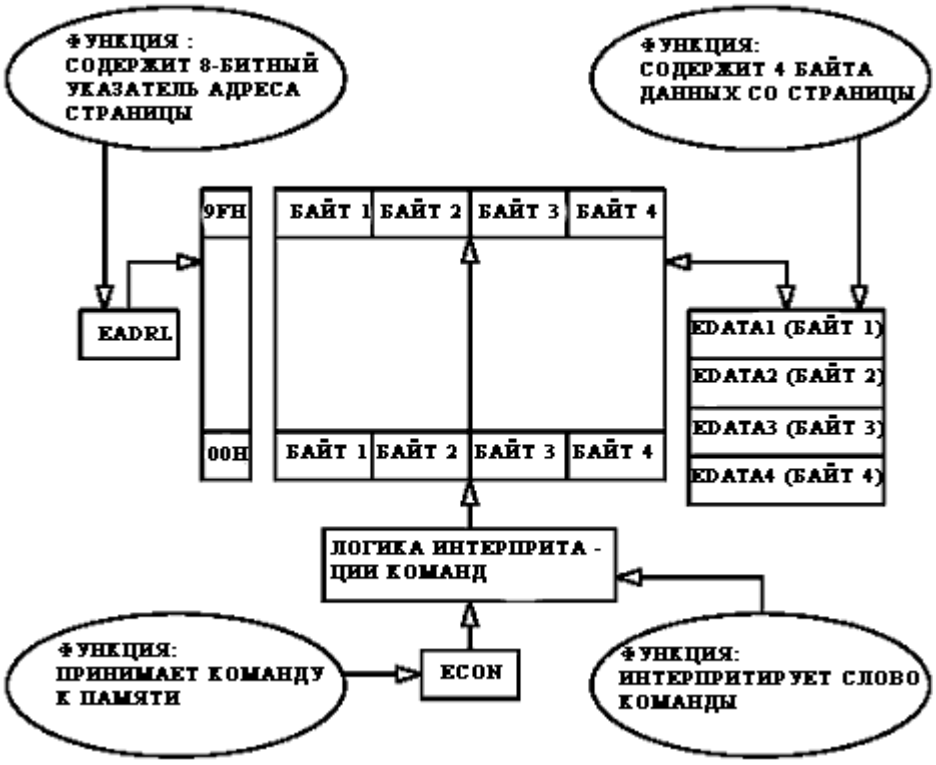


Рис. 1.20. Схема регистрового интерфейса массива Flash/EE-памяти данных

Таблица 1.16

Перечень команд управления Flash/EE-памятью данных, записываемых в специальный регистр ECON

Команда	**Режим**
01h	«ЧТЕНИЕ СТРАНИЦЫ» Возвращает 4 байта данных, считываемых в регистры EDATA1-EDATA4 со страницы, адрес которой находится в регистре EADRL
02h	«ЗАПИСЬ СТРАНИЦЫ» В результате выполнения этой команды 4 байта данных, находящихся в регистрах EDATA1-EDATA4, записываются в страницу, адрес которой находится в регистре EADRL. При подаче этой команды предполагается, что предназначенная для записи страница была предварительно стерта
03h	КОМАНДА, ЗАРЕЗЕРВИРОВАННАЯ ДЛЯ ВНУТРЕННИХ ЦЕЛЕЙ Код 03h не следует записывать в регистр ECON
04h	«ВЕРИФИКАЦИЯ СТРАНИЦЫ» Позволяет пользователю выполнить верификацию, если данные, находящиеся в регистрах EDATA1-EDATA4, уже содержатся в странице памяти с адресом, находящимся в регистре EADRL. Последовательное чтение регистра ECON даст «0», если верификация верна, и ненулевую величину, если верификация не верна
05h	«СТИРАНИЕ СТРАНИЦЫ» В результате выполнения этой команды происходит стирание данных 4-байтовой страницы с адресом, находящимся в регистре EADRL
06h	«СТИРАНИЕ ВСЕГО МАССИВА» Выполнение этой команды приводит к стиранию всех ста шестидесяти страниц (640 байт) Flash/EE-памяти данных
07h-FFh	РЕЗЕРВНЫЕ КОМАНДЫ Команды зарезервированы для использования в перспективных изделиях

EADRL (регистр адреса страницы Flash/EE-памятью данных)

Адрес C6h, значение после сброса 00h, битовая адресация отсутствует. Содержит адрес страницы Flash/EE-памяти данных (всего имеется 160 адресов).

EDATA1/EDATA2/EDATA3/EDATA4
(регистры данных страницы Flash/EE-памяти данных)

Адреса BCh/BDh/BEh/BFh, значения после сброса 00h/00h/00h/00h, битовая адресация отсутствует.

Содержат данные со страницы Flash/EE-памяти данных для записи или после чтения.

Типовые значения длительностей выполнения приведенных в табл. 1.16 команд следующие:

– стирание всего массива (640 байт) – 2 мс;
– стирание 1 страницы (4 байта) – 2 мс;
– программирование страницы (4 байта) – 250 мкс;
– чтение страницы (4 байта) – 1 командный цикл ядра.

Доступ к массиву Flash/EE-памяти данных из целевой программы реко-

мендуется осуществлять в соответствии со следующим типовым алгоритмом.

Сначала производится запись в регистр EADRL значения адреса страницы, к которой будет производиться обращение, затем в регистры EDATA1-EDATA заносятся данные, которые будут записываться в память, если предполагается выполнить цикл записи. При выполнении цикла чтения в эти регистры заносить ничего не нужно. В заключение производится запись управляющего слова в регистр ECON, которая инициирует запуск выполнения одной из пяти команд (табл. 1.16). С момента записи в ECON ядро микроконтроллера ADuC824 начинает работать в холостом режиме до тех пор, пока запущенный цикл программирования или стирания не закончится. Это означает, что после выполнения инструкции MOV, производящей запись константы в регистр ECON, что по времени занимает два машинных цикла, следующая инструкция не будет выполняться до тех пор, пока не завершится операция записи во Flash/EE-память (в течение 250 мкс или 2 мс). Пока длится запись, ядро ADuC824 не будет реагировать на запросы прерывания, хотя некоторые периферийные устройства, например, таймеры-счетчики будут продолжать работать.

Хотя МК поставляется со стертой Flash/EE-памятью данных (во все ее ячейки записан код 0FFh), тем не менее, в составе целевого программного обеспечения рекомендуется иметь подпрограмму стирания всего массива памяти. Команда «стереть все» заключается в записи кода 06h в регистр ECON, что инициирует стирание всех 640 байт массива Flash/EE-памяти данных. Эта операция в мнемониках ассемблера 8051, выглядит следующим образом:

```
MOV ECON, #06h ;команда «стереть все»
;аппаратная задержка 2 мс
```

Значение произвольного байта в массиве Flash/EE-памяти данных можно модифицировать только в том случае, если он был предварительно стерт, т. е. если туда был записан код 0FFh. Специфика структурной организации Flash/EE-памяти МК такова, что стирание можно производить только постранично, поэтому при запуске команды стирания байта физически произойдет стирание всей включающей его страницы, состоящей из четырех байт.

Рассмотрим следующий пример. Пусть требуется записать код F3h во второй байт страницы с адресом 03h массива памяти данных. Три оставшихся байта этой страницы уже содержат какие-то данные, потерять которые нежелательно. Для избежания потерь данных содержимое страницы перед ее стиранием сохраняется в специальных регистрах данных EDATA1 – EDATA4. Модификация требуемого байта производится не в самой странице, а в соответствующем регистре данных. После этого страница перезаписывается целиком содержимым EDATA1 – EDATA4. В мнемониках ассемблера 8051 описанная последовательность действий выглядит следующим образом:

```
MOV EADRL, #03h  ; установить указатель адреса страницы
MOV ECON, #01h   ; прочитать страницу
MOV EDATA2,#0F3h ; записать новый байт
MOV ECON, #05h   ; стереть страницу
MOV ECON, #02h   ; записать страницу во Flash/EE память
```

1.7. Модуль ЦАП

В составе МК имеется встроенный модуль 12-битного ЦАП, преобразующий цифровой код на своем входе в выходное аналоговое напряжение. Цифровой вход ЦАП программно доступен в виде пары специальных регистров. ЦАП имеет на выходе буферный усилитель напряжения, обеспечивающий размах выходного сигнала во всем диапазоне питающего напряжения (rail-to-rail). На выход усилителя можно подключать нагрузку сопротивлением не менее 10 кОм и емкостью не более 100 пФ. ЦАП имеет два программно выбираемых диапазона выходного напряжения: от 0 до VREF (VREF = 2,5 В, это напряжение внутреннего ИОН модуля ЦАП) и от 0 до AVDD. Модуль ЦАП может работать в 12- или в 8-битном режимах. Программный доступ к ЦАП осуществляется через специальный регистр управления DACCON и два специальных регистра данных DACL и DACH. Программно также можно выбрать, на какую ножку микроконвертора будет подаваться выходное напряжение ЦАП: вывод 3 или вывод 12. Следует отметить, что в 12-битном режиме, напряжение на выходе ЦАП будет модифицировано в соответствии с входным кодом только тогда, когда произойдет запись в регистр DACL. По этой причине для обеспечения корректного функционирования модуля регистры данных ЦАП должны модифицироваться программой в следующем порядке: сначала DACH, а потом DACL. 12-битное слово данных следует записывать в регистры данных ЦАП таким образом, чтобы DACL содержал восемь младших разрядов, а младшая тетрада регистра DACH – четыре старших.

DACCON (регистр управления модулем ЦАП)

–	–	–	**DACPIN**	**DAC8**	**DACRN**	**DACCLR/**	**DACEN**

Адрес FDh, значение после сброса 00h, битовая адресация отсутствует. Назначение битов регистра DACCON описано в табл. 1.17.

Таблица 1.17

Назначение битов специального регистра DACCON

Бит	Имя	Описание
7	–	Зарезервирован для дальнейшего использования
6	–	Зарезервирован для дальнейшего использования
5	–	Зарезервирован для дальнейшего использования
4	DACPIN	Выбор ножки устройства для подачи на него выхода ЦАП. Устанавливается пользователем для того, чтобы подать выход ЦАП на ножку 12 (P1.7/AIN4/DAC). Сбрасывается пользователем для того, чтобы подать выход ЦАП на ножку 3 (P1.2/DAC/IEXC1)
3	DAC8	Бит выбора 8-битного режима ЦАП. Устанавливается пользователем для разрешения 8-битной работы ЦАП. В этом режиме 8 бит регистра данных ЦАП DACL загружаются в 8 старших разрядов ЦАП, а 4 младших разряда ЦАП устанавливаются равными нулю. Сбрасывается пользователем для разрешения обычной 12-битной работы ЦАП
2	DACRN	Бит выбора выходного диапазона ЦАП. Устанавливается пользователем для установки диапазона ЦАП: 0–AVDD. Сбрасывается пользователем для установки диапазона ЦАП: 0–2,5 В
1	DACCLR	Бит очистки ЦАП. Устанавливается пользователем для разрешения обычной работы ЦАП. Сбрасывается пользователем для обнуления регистров данных ЦАП: DACL и DACH
0	DACEN	Бит разрешения ЦАП. Устанавливается пользователем для разрешения обычной работы ЦАП. Сбрасывается пользователем для снятия питания с ЦАП

DACL/DACH (регистры данных ЦАП)

Адреса FBh/FCh, значения после сброса 00h/00h, битовая адресация отсутствует.

В эти регистры записывается цифровой код, в соответствии с которым устанавливается выходное аналоговое напряжение ЦАП.

1.8. Внутренняя система ФАПЧ

В составе целевых приложений МК используется совместно с часовым кварцевым резонатором на частоту 32 768 Гц. Для получения стабильной рабочей тактовой частоты 12,582 912 МГц внутренняя система фазовой автоподстройки частоты умножает эту величину на 384. Эта частота является максимальной тактовой частотой микропроцессорного ядра ADuC824. Ядро может работать и на частотах, меньших 12,582 912 МГц, что позволяет понизить потребляемую устройством мощность в тех приложениях, где не требуется максимальная производительность. Тактовая частота ядра после включения питания по умолчанию равна тактовой частоте ФАПЧ, деленной на 8

(1,572 864 МГц). Частота тактирования АЦП также вырабатывается из частоты синхронизации ФАПЧ, причем, частота тактирования модуляторов АЦП совпадает по величине с частотой кварцевого резонатора. Такой выбор схемы синхронизации гарантирует, что модуляторы и ядро всегда будут работать синхронно вне зависимости от значения тактовой частоты ядра.

PLLCON (регистр управления встроенной системой ФАПЧ)

OSC_PD	LOCK	–	LTEA/	FINT	CD2	CD1	CD0

Адрес D7h, значение после сброса 03h, битовая адресация отсутствует.

Через регистр PLLCON осуществляется управление встроенной системой ФАПЧ. Назначение битов специального регистра PLLCON описано в табл. 1.18.

Таблица 1.18

Назначение битов специального регистра PLLCON

Бит	Имя	Описание
7	OSC__PD	Бит снятия питания с тактового генератора. Устанавливается пользователем для остановки тактового генератора 32 кГц в режиме «снятое питание». Сбрасывается пользователем для разрешения работы тактового генератора 32 кГц в режиме «снятое питание». Эта возможность позволяет счетчику временного интервала (TIC) продолжать работать даже в режиме «снятое питание»
6	LOCK	Бит прерывания при блокировке ФАПЧ. Этот бит доступен только для чтения. Устанавливается автоматически при подаче питания для индикации того, что система ФАПЧ правильно отслеживает тактовую частоту внешнего кварцевого резонатора. Если цепь внешнего резонатора разорвется, то система ФАПЧ зависнет и ядро остановится. Сбрасывается автоматически при подаче питания для индикации того, что система ФАПЧ не отслеживает тактовую частоту внешнего резонатора. Это может происходить из-за отсутствия тактовой частоты резонатора или самого внешнего резонатора при подаче питания. В этом режиме на выходе ФАПЧ может генерироваться частота 12,58 МГц ±20 %
5	–	Зарезервирован для дальнейшего использования; должен быть сброшен
4	LTEA/	Чтение этого бита возвращает значение внешнего логического уровня на ножке EA/, зафиксированное в момент подачи питания или сброса системы
3	FINT	Бит быстрой реакции на прерывание. Устанавливается пользователем для разрешения ускоренной реакции на любое прерывание. Ускорение состоит в том, что прерывание будет обрабатываться на более высокой тактовой частоте ядра вне зависимости от состояния битов CD2-CD0 (см. ниже). Как только целевая программа возвращается из блока обработки прерывания, выполнение кода продолжается на прежней частоте ядра, выбранной битами CD2-CD0. Сбрасывается пользователем, чтобы запретить быструю реакцию на прерывание

Окончание табл. 1.18

Бит	Имя	Описание				
2–0	CD2–CD0	Биты настойки делителя рабочей частоты ядра. Состояние этих битов определяет частоту, на которой будет работать вычислительное ядро микроконтроллера.				
		CD2	CD1	CD0	Тактовая частота	Частота ядра, МГц
		0	0	0	12,582912	Частота ядра по умолчанию.
		0	0	1	6,291456	
		0	1	0	3,145728	
		0	1	1	1,572864	
		1	0	0	0,786432	
		1	0	1	0,393216	
		1	1	0	0,196608	
		1	1	1	0,098304	

1.9. Счетчик временных интервалов TIC

В состав МК входит аппаратный модуль TIC (Time interval counter), представляющий собою программируемый составной счетчик со схемой сравнения. Он предназначен для отсчета временных интервалов, превышающих по длительности максимальное время, отсчитываемое стандартным таймером 8051-совместимого контроллера. Этот счетчик способен отсчитывать временные интервалы в диапазоне от 1/128 с до 255 ч. Ввиду того, что TIC тактируется сигналом, поступающим непосредственно от генератора с внешним кварцевым резонатором, а не от системы ФАПЧ, он обладает способностью продолжать счет времени в режимах работы МК с отключением ядра и периферии и пониженным энергопотреблением («холостой» режим и режим «снятое питание»). Наличие такого узла в составе устройства позволяет программно организовать периодические сбор или выдачу данных в целевых приложениях, представляющих собой удаленные датчики или устройства сбора данных с батарейным питанием.

Упрощенная блок-схема модуля TIC приведена на рис. 1.21. Специальный регистр TIMECON является его управляющим регистром. Путем записи в биты ITS0 и ITS1 регистра TIMECON различных комбинаций можно произвести выбор счетчика базовых интервалов времени, выход переполнения которого будет подключен к входу счетчика, отсчитывающего значение временного интервала в базовых интервалах.

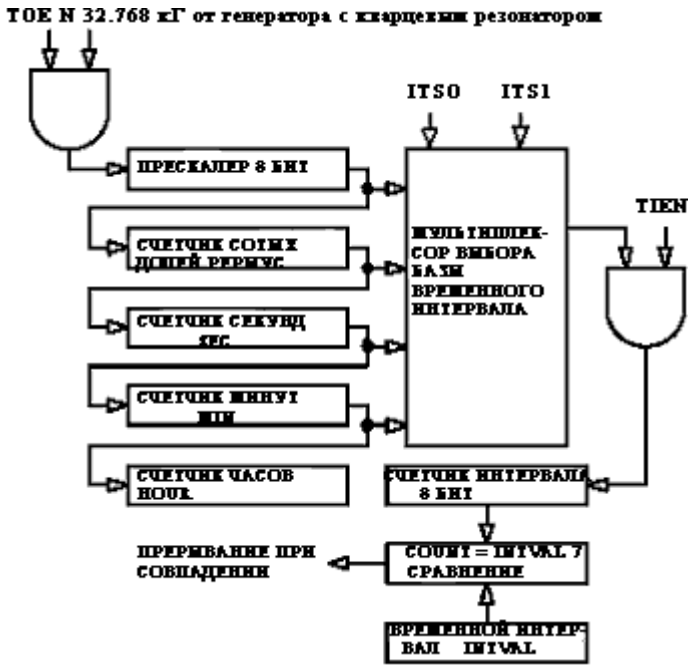


Рис. 1.21. Схема модуля TIC

Когда содержимое этого счетчика становится равным значению, загруженному в регистр INTVAL, бит TII регистра TIMECON устанавливается и вызывается прерывание, если оно разрешено. Когда МК находится в режиме «снятое питание», установка бита TII при разрешенном прерывании от TIC повлечет за собой выход устройства из этого режима. Выполнение программы при этом будет продолжено переходом по вектору прерывания от TIC (адрес программной памяти – 0053h). Если в счетчики базовых интервалов программно записать текущее время, то можно использовать модуль TIC в качестве часов реального времени.

TIMECON (регистр управления счетчиком временных интервалов TIC)

–	–	ITS1	ITS0	STI	TII	TIEN	TCEN

Адрес A1h, значение после сброса 00h, битовая адресация отсутствует.
Назначение битов специального регистра TIMECON описано в табл. 1.19.

Таблица 1.19

Назначение битов специального регистра TIMECON

Бит	Имя	Описание
7	–	Зарезервирован для дальнейшего использования.
6	–	Зарезервирован для дальнейшего использования. Для совместимости с кодом будущих изделий этот бит следует всегда устанавливать.
5	ITS1	Биты выбора базового интервала времени.

Окончание табл. 1.19

<table>
<tr><th>Бит</th><th>Имя</th><th>Описание</th></tr>
<tr><td>4</td><td>ITS0</td><td>Записываются пользователем для выбора входной частоты счетчика временного интервала.
<table>
<tr><th>ITS1</th><th>ITS0</th><th>Базовый интервал времени</th></tr>
<tr><td>0</td><td>0</td><td>1/128 секунды</td></tr>
<tr><td>0</td><td>1</td><td>секунда</td></tr>
<tr><td>1</td><td>0</td><td>минута</td></tr>
<tr><td>1</td><td>1</td><td>час</td></tr>
</table></td></tr>
<tr><td>3</td><td>STI</td><td>Бит выбора режима однократного временного интервала.
Устанавливается пользователем для выполнения однократной генерации интервального тайм-аута. Если бит STI установлен, тайм-аут сбросит бит TIEN. Сбрасывается пользователем для разрешения автоматической перезагрузки счетчика интервала и последующего запуска счета по каждому интервальному тайм-ауту.</td></tr>
<tr><td>2</td><td>TII</td><td>Бит (флаг) прерывания от счетчика временного интервала.
Является битом только для чтения, обычно находится в нулевом состоянии. Устанавливается, когда содержимое 8-разрядного счетчика временного интервала становится равным значению, записанному в регистр INTVAL.
Сбрасывается целевой программой.</td></tr>
<tr><td>1</td><td>TIEN</td><td>Бит разрешения счетчика временного интервала.
Устанавливается пользователем для разрешения 8-разрядного счетчика временного интервала.
Сбрасывается пользователем для запрещения и обнуления счетчика временного интервала.</td></tr>
<tr><td>0</td><td>TCEN</td><td>Бит разрешения счета времени.
Устанавливается пользователем для разрешения подачи счетных импульсов на счетчики времени (базовых интервалов).
Сбрасывается пользователем для запрещения подачи счетных импульсов на счетчики времени (базовых интервалов) и обнуления этих счетчиков.
В регистры счетчиков времени (базовых интервалов) HTHSEC, SEC, MIN, HOUR можно производить запись только тогда, когда бит TCEN сброшен.</td></tr>
</table>

INTVAL (регистр выбора значения интервала, отсчитываемого модулем TIC)

Адрес A6h, значение после сброса 00h, битовая адресация отсутствует, диапазон величин 0 ... 255.

Содержимое этого регистра, доступное для модификации в целевой программе, используется в качестве второго операнда при аппаратном сравнении с содержимым одного предварительно выбранного 8-разрядного счетчика временного интервала. При достижении равенства отсчет временного интервала завершается и генерируется прерывание от TIC.

HTHSEC (регистр счета сотых долей секунд модуля TIC)

Адрес A2h, значение после сброса 00h, битовая адресация отсутствует, диапазон величин 0 ... 127.

Этот регистр инкрементируется по истечении интервала времени 1/128

секунды, если бит TCEN в регистре TIMECON установлен. Регистр HTHSEC считает от 0 до 127, а в момент его сброса в 0 происходит инкрементирование регистра SEC.

SEC (регистр счета секунд модуля TIC)

Адрес A3h, значение после сброса 00h, битовая адресация отсутствует, диапазон величин 0 ... 59.

Этот регистр инкрементируется каждую секунду, если бит TCEN в регистре TIMECON установлен. Регистр SEC считает от 0 до 59, а в момент его сброса в 0 происходит инкрементирование регистра MIN.

MIN (регистр счета минут модуля TIC)

Адрес A4h, значение после сброса 00h, битовая адресация отсутствует, диапазон величин 0 ... 59.

Этот регистр инкрементируется каждую минуту, если бит TCEN в регистре TIMECON установлен. Регистр MIN считает от 0 до 59, а в момент его сброса в 0 происходит инкрементирование регистра HOUR.

HOUR (регистр счета часов модуля TIC)

Адрес A5h, значение после сброса 00h, битовая адресация отсутствует, диапазон величин 0 ... 23.

Этот регистр инкрементируется каждый час, если бит TCEN в регистре TIMECON установлен. Регистр HOUR считает от 0 до 23.

1.10. Сторожевой таймер WDT

Назначение встроенного в МК сторожевого таймера WDT состоит в том, чтобы путем вызова процедуры сброса микроконвертора или генерации прерывания вывести устройство из ошибочного, не предусмотренного алгоритмом состояния зацикливания или останова в каком-либо блоке целевой программы. Критерием ошибочного выполнения программы с точки зрения сторожевого таймера является наступление тайм-аута WDT, т. е. достижение некоторой заданной величины временного интервала, в течение которого управление в программе не передается за пределы блока, где отсутствуют инструкции сброса WDT. Причинами такого программного сбоя могут стать некорректное построение самой программы, а также влияние электрической или радиочастотной помехи.

Работа сторожевого таймера может быть запрещена путем сброса бита WDE в регистре управления таймером WDCON. В случае разрешения работы схема таймера вырабатывает системный сброс или прерывание (в зависимости от значения бита WDIR регистра WDCON), если целевая программа не переустановила бит WDE регистра WDCON в течение временного периода сторожевого таймера, величина которого определяется значениями битов PRE3 – PRE0 специального регистра WDCON.

Аппаратно сторожевой таймер представляет собой 16-битный счетчик, который тактируется импульсами с частотой 32768 Гц. Регистром управления WDT является специальный регистр WDCON. Назначение его битов описано в табл. 1.20. Следует отметить, что регистр WDCON доступен для записи в программе только в том случае, если запись производится в виде последовательности из двух инструкций, как показано в таблице.

WDCON (регистр управления сторожевым таймером)

PRE3	PRE2	PRE1	PRE0	WDIR	WDS	WDE	WDWR

Адрес C0h, значение после сброса 10h, битовая адресация имеется.

Таблица 1.20

Назначение битов специального регистра WDCON

<table>
<tr><th>Биты</th><th>Имя</th><th colspan="7">Описание</th></tr>
<tr><td>7
6
5
4</td><td>PRE3
PRE2
PRE1
PRE0</td><td colspan="7">Биты выбора коэффициента деления предварительного делителя сторожевого таймера. Временной период сторожевого таймера определяется выражением:
$t_{WD} = (2^{PRE} \times (2^9/f_{PLL}))$,
где $0 \le PRE \le 7$, $f_{PLL} = 32{,}768$ кГц</td></tr>
<tr><td></td><td></td><td>PRE3</td><td>PRE2</td><td>PRE1</td><td>PRE0</td><td>Период тайм-аута (мс)</td><td>Действие</td></tr>
<tr><td></td><td></td><td>0</td><td>0</td><td>0</td><td>0</td><td>15,6</td><td>Сброс или прерывание</td></tr>
<tr><td></td><td></td><td>0</td><td>0</td><td>0</td><td>1</td><td>31,2</td><td>Сброс или прерывание</td></tr>
<tr><td></td><td></td><td>0</td><td>0</td><td>1</td><td>0</td><td>62,5</td><td>Сброс или прерывание</td></tr>
<tr><td></td><td></td><td>0</td><td>0</td><td>1</td><td>1</td><td>125</td><td>Сброс или прерывание</td></tr>
<tr><td></td><td></td><td>0</td><td>1</td><td>0</td><td>0</td><td>250</td><td>Сброс или прерывание</td></tr>
<tr><td></td><td></td><td>0</td><td>1</td><td>0</td><td>1</td><td>500</td><td>Сброс или прерывание</td></tr>
<tr><td></td><td></td><td>0</td><td>1</td><td>1</td><td>0</td><td>1 000</td><td>Сброс или прерывание</td></tr>
<tr><td></td><td></td><td>0</td><td>1</td><td>1</td><td>1</td><td>2 000</td><td>Сброс или прерывание</td></tr>
<tr><td></td><td></td><td>1</td><td>0</td><td>0</td><td>0</td><td>0,0</td><td>Немедленный сброс</td></tr>
<tr><td></td><td></td><td colspan="5">PRE3-PRE0 > 1001</td><td>Зарезервировано</td></tr>
<tr><td>3</td><td>WDIR</td><td colspan="7">Бит разрешения прерывания от сторожевого таймера. Если этот бит устанавливается пользователем, сторожевой таймер будет вырабатывать прерывание вместо сброса системы по окончанию интервала тайм-аута. Это прерывание не может быть запрещено командой CLR EA и является прерыванием с неизменно высоким приоритетом. Если сторожевой таймер не используется для слежения за системой, его можно использовать в качестве системного таймера общего назначения, причем предварительный делитель в этом случае используется для установки периода тайм-аута, по окончании которого будет вырабатываться прерывание</td></tr>
<tr><td>2</td><td>WDS</td><td colspan="7">Бит состояния сторожевого таймера. Устанавливается контроллером сторожевого таймера для индикации того, что его тайм-аут закончился. Сбрасывается путем записи в бит нуля или при внешнем аппаратном сбросе. Бит WDS не сбрасывается при сбросе сторожевого таймера</td></tr>
</table>

Окончание табл. 1.20

Биты	Имя	Описание
1	WDE	Бит разрешения сторожевого таймера. Устанавливается пользователем для разрешения сторожевого таймера и обнуления его счетчиков. Если этот бит не переустанавливается целевой программой в течение периода тайм-аута, то таймер сгенерирует сброс системы или прерывание в зависимости от состояния бита WDIR. Сбрасывается следующими способами: программным сбросом бита WDE, сбросом сторожевого таймера (WDIR = «0»), аппаратным сбросом, прерыванием от PSM

0	WDWR	Бит разрешения записи в сторожевой таймер. Для записи данных в регистр WDCON требуется выполнить последовательность из двух команд: сначала следует установить бит WDWR, и только следующей командой следует записать данные в WDCON. Фрагмент программы, реализующий описанную последовательность, выглядит так:
		`CLR EA ;запретить прерывания на время записи в WDT`
		`SETB WDWR ;разрешить запись в WDCON`
		`MOV WDCON, #72h ;разрешить WDT с интервалом тайм-аута 2,0 с`
		`SETB EA ;снова разрешить прерывания (если требуется)`

1.11. Монитор источников питания PSM

Встроенный монитор источников питания PSM, по определению, предназначен для слежения за источниками питающих напряжений устройства AVDD и DVDD. Когда PSM программно разрешен, он сигнализирует программе о падении значений напряжений AVDD и DVDD ниже установленных пользователем порогов, которые можно выбрать из четырех возможных значений. Для правильной работы PSM напряжение AVDD должно быть не менее 2,7 В. Работа монитора источников питания управляется специальным регистром PSMCON, назначение битов которого описано в табл. 1.21. При падении напряжения ниже порога и в случае, если в специальном регистре приоритета и разрешения вторичных прерываний IEIP2 разрешено прерывание от PSM, монитор вызовет прерывание, установив бит PSMI в регистре PSMCON. Этот бит может быть программно сброшен только тогда, когда после подъема значения напряжения аварийного источника питания до величины, превышающей заданный порог, не истечет, по меньшей мере, 250 мс. Эта функция PSM дает возможность целевой программе избежать возможной порчи данных в рабочих регистрах, которая могла бы иметь место, если бы выполнение программы продолжалось при недостаточном уровне питающего напряжения.

CMPD	CMPA	PSMI	TPD1	TPD0	TPA1	TPA0	PSMEN

Таблица 1.21

Назначение битов специального регистра PSMCON

Биты	Имя	Описание
7	CMPD	Бит компаратора DV_{DD}. Этот бит доступен только для чтения и непосредственно отражает состояние выхода компаратора DV_{DD}. Считывание в бите «единицы» показывает, что уровень напряжения DV_{DD} выше установленного порога. Считывание в бите «нуля» показывает, что уровень напряжения DV_{DD} находится ниже установленного порога
6	CMPA	Бит компаратора AV_{DD}. Этот бит доступен только для чтения и непосредственно отражает состояние выхода компаратора AV_{DD}. Считывание в бите «единицы» показывает, что уровень напряжения AV_{DD} находится выше установленного порога. Считывание в бите «нуля» показывает, что уровень напряжения AV_{DD} находится ниже установленного порога
5	PSMI	Бит (флаг) прерывания от монитора источника питания. Этот бит устанавливается аппаратно, если любой из битов CMPA или CMPD сбрасывается, что указывает на пониженный уровень напряжения аналогового или цифрового источника питания. Бит PSMI может вызывать прерывание. Как только биты CMPD/CMPA устанавливаются (и остаются установленными), запускается аппаратный счетчик, отсчитывающий интервал 250 мс. Когда этот интервал истекает, бит PSMI сбрасывается. Бит PSMI также доступен для записи программно, однако, если при этом выход любого из компараторов будет находиться в низком состоянии, то программно сбросить бит PSMI не удастся
4	TPD1	Биты выбора порога для DV_{DD}
3	TPD0	Этими битами выбирается порог напряжения следующим образом: TPD1 | TPD0 | Выбранный порог для DVDD, В 0 | 0 | 4,63 0 | 1 | 3,08 1 | 0 | 2,93 1 | 1 | 2,63
2	TPA1	Биты выбора порога для AVDD
1	TPA0	Этими битами выбирается порог напряжения следующим образом: TPA1 | TPA0 | Выбранный порог для AVDD, В 0 | 0 | 4,63 0 | 1 | 3,08 1 | 0 | 2,93 1 | 1 | 2,63
0	PSMEN	Бит разрешения монитора источника питания PSM. Устанавливается пользователем для разрешения монитора источника питания. Сбрасывается пользователем для запрета монитора источника питания

Монитор питания имеет также аппаратную защиту от генерации прерываний при случайных кратковременных бросках питающего напряжения.

1.12. Последовательный интерфейс SPI

Набор периферийных узлов МК включает в себя аппаратный модуль стандартного синхронного последовательного интерфейса SPI, который обеспечивает возможность работы в полнодуплексном режиме, т. е. позволяет одновременно синхронно производить прием и передачу байта данных. Аппаратное построение интерфейса SPI таково, что физически он использует те же выводы микросхемы и внутреннюю логику, что и встроенный в МК периферийный интерфейс, совместимый со стандартом I^2C. В связи с этим, целевая программа может в каждый текущий момент времени задействовать для своих целей только один из названных интерфейсов. SPI программно конфигурируется для работы в режиме «ведущий» или в режиме «ведомый» и использует следующие выводы микроконвертора:

MISO (вход/выход, линия ввода-вывода данных)

Линия MISO (master in, slave out) автоматически конфигурируется как вход в режиме «ведущий» и как выход в режиме «ведомый». Для организации обмена линия MISO ведущего устройства должна подключаться к линии MISO ведомого устройства. Данные передаются последовательно по восемь бит, причем старший значащий разряд (СЗР) передается первым.

MOSI (вход/выход, линия ввода-вывода данных)

Линия MOSI (master out, slave in) автоматически конфигурируется как выход в режиме «ведущий» и как вход в режиме «ведомый». Для организации обмена линия MOSI ведущего устройства должна подключаться к линии MOSI ведомого устройства. Данные передаются последовательно по восемь бит, причем старший значащий разряд (СЗР) передается первым.

SCLOCK (вход/выход, линия ввода-вывода последовательного синхросигнала)

Синхросигнал, генерируемый «ведущим» устройством, выдается через линию SCLOCK и используется для синхронизации данных, передаваемых и принимаемых по линиям MOSI и MISO. В каждом периоде синхросигнала происходит передача и прием одного бита данных. Таким образом, один байт данных передается (принимается) за восемь периодов сигнала SCLOCK. Линия SCLOCK автоматически конфигурируется как выход в режиме «ведущий» и как вход в режиме «ведомый». В режиме «ведущий» с помощью битов CPOL, CPHA, SPR0 и SPR1 специального регистра SPICON можно управлять скоростью передачи, полярностью и фазой синхросигнала. В режиме «ведомый» биты CPHA и CPOL специального регистра SPICON необходимо программировать таким образом, чтобы установленные фаза и полярность совпадали с фазой и полярностью синхросигнала «ведущего» устройства, поскольку как в режиме «ведущий», так и «ведомый» выдача данных на линию происходит по одному фронту синхросигнала, а их фиксация в приемнике – по другому.

SS/ (вход, линия выбора «ведомого»)

Входная линия SS/ (slave select) в интерфейсе SPI используется только то-

гда, когда МК сконфигурирован как «ведомое» устройство. Подачей на ножку SS/ внешнего сигнала активного низкого уровня разрешается работа SPI «ведомого». Поскольку в режиме «ведомый» данные могут приниматься или передаваться устройством только при низком уровне SS/, это позволяет использовать МК в составе системы, состоящей из одного «ведущего» и нескольких «ведомых» устройств, подключенных к одной интерфейсной шине. Если бит CPHA в специальном регистре SPICON «ведомого» устройства установлен, то на входе SS/ может постоянно присутствовать внешнее напряжение низкого уровня. Если бит CPHA сброшен, то вход SS/ должен переводиться в низкое состояние до начала передачи или приема первого бита в байте и возвращаться в высокое состояние после передачи или приема последнего бита в том же байте. Программное чтение состояния входа SS/ в режиме «ведомый» осуществляется путем чтения бита SPR0 специального регистра SPICON.

Для программного управления интерфейсом SPI используется следующие специальные регистры:

SPICON (регистр управления SPI)

ISPI	WCOL	SPE	SPIM	CPOL	CPHA	SPR1`	SPR0

Адрес A8h, значение после сброса 04h, битовая адресация имеется.

Назначение битов специального регистра SPICON описано в табл. 1.22.

Таблица 1.22

Назначение битов специального регистра SPICON

7	ISPI	Бит (флаг) прерывания от SPI. Устанавливается аппаратно в конце каждой передачи по SPI. Сбрасывается непосредственно (целевой программой) или косвенно (путем чтения регистра SPIDAT).
6	WCOL	Бит ошибки (коллизии) при записи. Устанавливается аппаратно, если происходит запись в регистр SPIDAT в то время, когда идет передача по SPI.
5	SPE	Бит разрешения интерфейса SPI. Устанавливается пользователем для разрешения интерфейса SPI. Сбрасывается пользователем для разрешения интерфейса I2C.
4	SPIM	Бит выбора режима SPI «ведущий/ведомый». Устанавливается пользователем для разрешения режима «ведущий» (вывод SCLOCK является выходом). Сбрасывается пользователем для разрешения режима «ведомый» (вывод SCLOCK является входом).
3	CPOL	Бит выбора полярности синхросигнала. Устанавливается пользователем, чтобы пассивный уровень сигнала SCLOCK был высоким. Сбрасывается пользователем, чтобы пассивный уровень сигнала SCLOCK был низким.
2	CPHA	Бит выбора фазы синхросигнала. Устанавливается пользователем, чтобы данные передавались по переднему фронту сигнала SCLOCK. Сбрасывается пользователем, чтобы данные передавались по заднему фронту сигнала SCLOCK.

Окончание табл. 1.22

<table>
<tr><th>Биты</th><th>Имя</th><th colspan="3">Описание</th></tr>
<tr><td>1</td><td>SPR1</td><td colspan="3">Биты выбора скорости передачи данных SPI.</td></tr>
<tr><td>0</td><td>SPR0</td><td colspan="3">Эти биты устанавливают скорость передачи в режиме «ведущий» следующим образом:</td></tr>
<tr><td></td><td></td><td>SPR1
0
0
1
1</td><td>SPR0
0
1
0
1</td><td>Выбранная скорость передачи
fCORE/2
fCORE/4
fCORE/8
fCORE/16
В режиме «ведомый» SPI (SPIM= «0») логический уровень на внешнем выводе SS/ (ножка 13) можно прочитать путем чтения бита SPR0.
(fCORE – частота ядра)</td></tr>
</table>

SPIDAT (регистр данных SPI)

Адрес F7h, значение после сброса 00h, битовая адресация отсутствует.

В специальный регистр SPIDAT записывается байт, предназначенный для передачи по SPI, из него также читается байт, принятый по SPI.

В зависимости от состояния битов специального регистра SPICON интерфейс SPI МК в режиме «ведущий» будет осуществлять обмен данными в нескольких возможных режимах синхронизации. На рис. 1.22. приведены временные диаграммы сигналов на линиях интерфейса, иллюстрирующие его работу при различных значениях управляющих битов. Как можно видеть из рисунка, в момент окончания передачи каждого байта происходит установка флага прерывания от модуля SPI – ISPI в специальном регистре SPICON.

В режиме «ведущий» обмен данными через SPI производится следующим образом. Инициатором обмена всегда является «ведущий». Для осуществления обмена линия SCLOCK, которая является выходом, начинает генерацию последовательности из восьми синхроимпульсов, тактируя таким образом передачу и прием. Начало генерации инициируется программной инструкцией записи подлежащего передаче байта в специальный регистр SPIDAT. Частота импульсов на выходе SCLOCK определяется состоянием битов SPR0 и SPR1 в специальном регистре SPICON. Линия SS/ МК в режиме «ведущий» в обмене не участвует. В том случае, если необходимо, чтобы МК произвел выбор внешнего «ведомого» устройства, подав активный уровень на его вход SS/, для этой цели следует использовать линию ввода-вывода общего назначения в каком-нибудь порту МК. Одновременно с передачей байта по линии MOSI из «ведущего» в «ведомое» устройство по перепадам сигнала SCLOCK производится прием байта «ведущим» из «ведомого» по линии MISO. После генерации восьми синхроимпульсов передача байта из SPIDAT будет завершена, а принятый байт окажется во входном сдвиговом регистре. Сразу после этого в регистре SPICON аппаратно установится флаг ISPI и будет сгенерировано прерывание по завершению приема-передачи через SPI. Данные из сдвигового регистра перепишутся в регистр SPIDAT. Обработка прерывания будет производиться, если оно разрешено.

В режиме «ведомый» линия SCLOCK МК является входом. «Ведомое» устройство не может инициировать обмен через SPI, поэтому при программной записи в регистр SPIDAT передача не начинается. Однако, следует иметь ввиду, что когда «ведущий» начнет обмен, то ему будет передано текущее содержимое регистра SPIDAT «ведомого», которое к этому моменту должно быть заранее записано целевой программой. Во время передачи байта по SPI на ножку SS/ «ведомого» извне должен быть подан низкий уровень. В отличие от режима «ведущий», в режиме «ведомый» передача байта производится по линии MISO, а прием – по линии MOSI. При этом тактирование приема-передачи осуществляется внешним тактовым сигналом, подаваемым с«ведущего» устройства на ножку SCLOCK «ведомого».

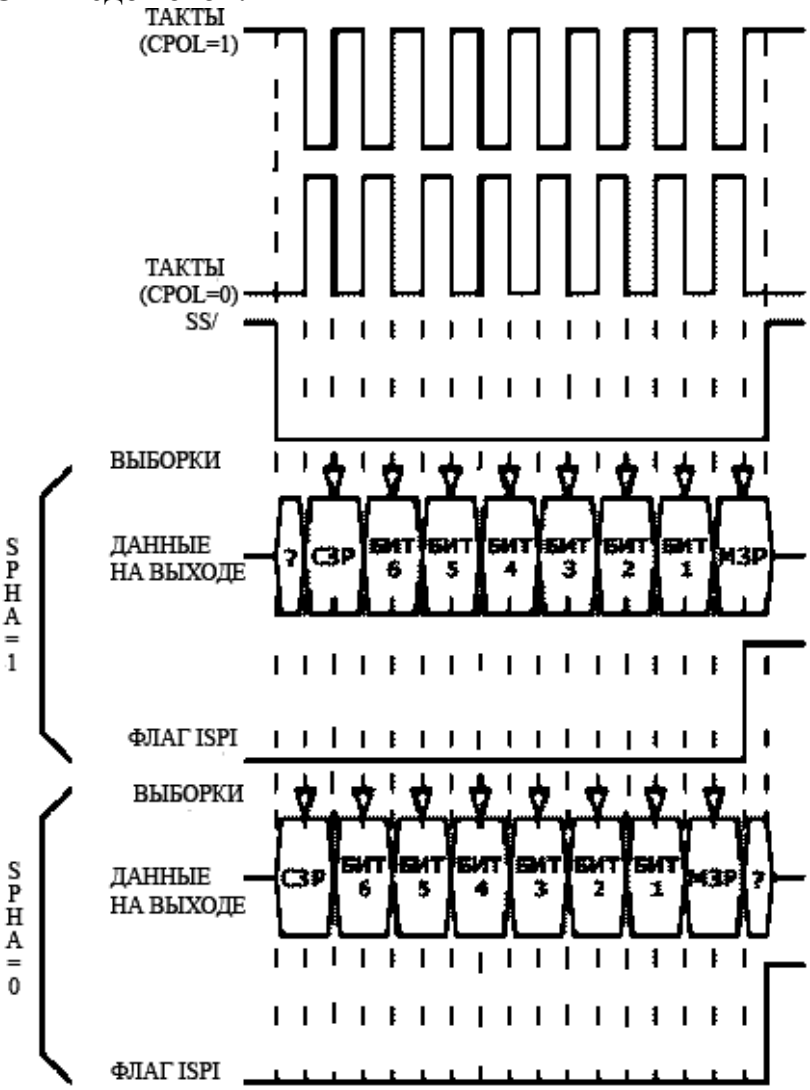


Рис. 1.22. Временная диаграмма интерфейса SPI

После прихода восьми синхроимпульсов передача байта из регистра SPIDAT будет завершена, а принятый байт окажется во входном сдвиговом регистре. Сразу после этого в регистре SPICON аппаратно установится флаг ISPI и будет сгенерировано прерывание по завершению приема-передачи через SPI. Данные из сдвигового регистра перепишутся в регистр SPIDAT. Обработка прерывания

будет производиться, если оно разрешено. Конец передачи (генерация прерывания) фиксируется по моменту приема восьми синхроимпульсов, если бит СРНА установлен, или по моменту установки сигнала на входе SS/, если бит СРНА сброшен.

1.13. Последовательный интерфейс, совместимый с I^2C

МК аппаратно поддерживает обмен по двухпроводному последовательному интерфейсу, совместимому со стандартом I^2C. Для более детального ознакомления с I^2C рекомендуется обратиться к [5], а применительно к семейству микроконверторов ADuC8XX – к [6]. Интерфейс I^2C использует те же выводы микросхемы и внутреннюю логику, что и встроенный в МК интерфейс SPI. Выбор (разрешение работы) одного из этих двух интерфейсов может быть произведен программно, путем установки или сброса бита SPE специального регистра SPICON. Интерфейс I^2C можно программно сконфигурировать как «программный ведущий» или как «аппаратный ведомый». В режиме «программный ведущий» обмен данными возможен на скоростях до 140 кбит/с, а в режиме «аппаратный ведомый» – до 3,4 Мбит/с. Имеющиеся на кристалле цепи фильтрации подавляют выбросы на линиях интерфейса SDATA и SCLOCK длительностью менее 50 нс с целью предотвращения ошибок при обмене.

Для организации шины I^2C используются линии, которые, в общем случае, должны быть «подтянуты» к «плюсу» источника питания с помощью внешних нагрузочных резисторов, как это показано на рис. 1.23. В МК «подтягивающие» резисторы уже имеются на кристалле, однако, по замечанию производителя, в системах с нескольким «ведомыми» могут понадобиться дополнительные внешние резисторы. Интерфейс МК, совместимый с I^2C, использует следующие выводы микроконвертора:

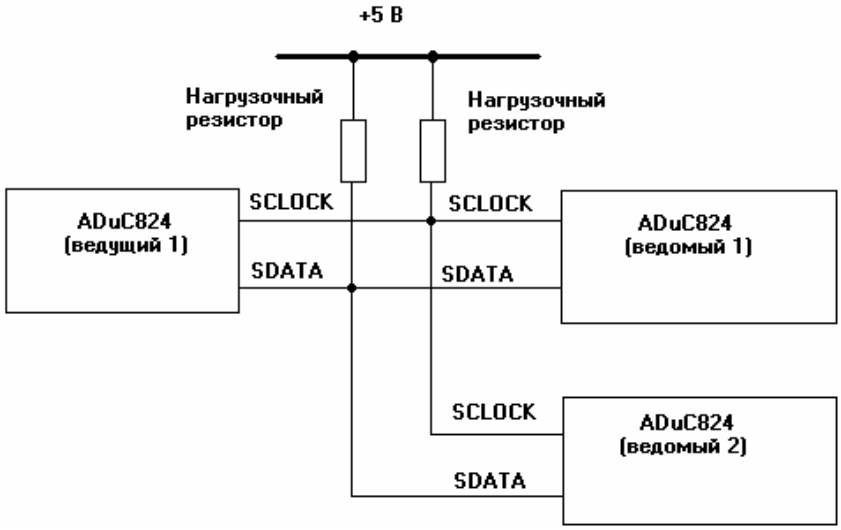


Рис. 1.23. Организация интерфейса I^2C

SCLOCK (вход/выход, линия ввода-вывода синхросигнала)

Сигнал на этой линии управляет передачей данных между «ведущим» и «ведомым» устройствами. Сигнал SCLOCK всегда генерируется «ведущим», однако «ведомый» может удерживать линию SCLOCK в низком уровне, когда он не готов продолжить обмен по шине. Такой режим называется «растягивание синхронизации» («clock stretching»). Каждый передаваемый или принимаемый бит данных тактируется одним импульсом сигнала SCLOCK.

SDATA (вход/выход, линия последовательного ввода-вывода данных)

Сигнал на этой линии используется для передачи и приема данных.

В соответствии со стандартом I^2C при передаче допускается изменение сигнала SDATA только в том случае, если сигнал SCLOCK находится в низком уровне. Перепады сигнала SDATA при высоком уровне SCLOCK трактуются другим участником обмена как условия начала или завершения передачи («START» или «STOP»).

Для программного управления интерфейсом используется следующие специальные регистры:

I2CCON (регистр управления I2C)

MDO	MDE	MCO	MDI	I2CM	I2CRS	I2CTX	I2CI

Адрес E8h, значение после сброса 00h, битовая адресация имеется.

Этот регистр содержит биты управления и статуса для программной настройки интерфейса I^2C. Назначение битов регистра I2CCON описано в табл. 1.23.

Таблица 1.23

Назначение битов специального регистра I2CCON

Биты	Имя	Описание
7	MDO	Бит выходных данных I2C (только для режима «программный ведущий»). Этот бит используется в интерфейсе «программный ведущий» для передачи данных по I^2C. Бит данных, помещенный программой в бит MDO, будет выведен на ножку SDATA, если бит разрешения вывода данных (MDE) установлен.
6	MDE	Бит разрешения вывода данных I^2C (только для режима «программный ведущий»). Устанавливается пользователем для назначения ножки SDATA на вывод данных (Tx). Сбрасывается пользователем для назначения ножки SDATA на ввод данных (Rx).
5	MCO	Бит выходного синхросигнала I^2C (только для режима «программный ведущий»). Этот бит используется для синхронизации передаваемых по I^2C данных в режиме «программный ведущий». Бит данных (значение синхросигнала), помещенный программой в бит MCO, будет выводиться на ножку SCLOCK.
4	MDI	Бит ввода данных I^2C (только для режима «программный ведущий»). Этот бит используется для приема данных по I^2C в режиме «программный ведущий». Внешний логический уровень (бит данных) на ножке SDATA помещается в бит MDI по сигналу SCLOCK, если бит разрешения вывода данных (MDE) сброшен.
3	I2CM	Бит выбора режима «ведущий/ведомый» I^2C. Устанавливается пользователем для разрешения режима I^2C «программный ведущий». Сбрасывается пользователем для разрешения режима I^2C «аппаратный ведомый».

Окончание табл. 1.23

Биты	Имя	Описание
2	I2CRS	Бит сброса I^2C (только для режима «аппаратный ведомый»). Устанавливается пользователем для сброса интерфейса I^2C. Сбрасывается кодом пользователя для нормальной работы I^2C.
1	I2CTX	Бит установки направления передачи I^2C (только для режима «аппаратный ведомый»). Устанавливается аппаратно, если через интерфейс осуществляется передача. Сбрасывается аппаратно, если через интерфейс осуществляется прием.
0	I2CI	Бит (флаг) прерывания от I^2C (только для режима «аппаратный ведомый»). Устанавливается аппаратно после того, как через интерфейс передан или принят байт данных. Сбрасывается автоматически, когда целевая программа производит чтение регистра I2CDAT.

I2CADD (регистр адреса I^2C)

Адрес 9Bh, значение после сброса 55h, битовая адресация отсутствует.

Регистр I2CADD содержит 7-разрядный адрес периферийного устройства (по умолчанию – 055h). Регистр доступен для записи и чтения в целевой программе и используется в соответствии с назначением только в режиме «аппаратный ведомый».

I2CDAT (регистр данных I^2C)

Адрес 9Ah, значение после сброса 00h, битовая адресация отсутствует.

В режиме «аппаратный ведомый» в регистр I2CDAT записывается байт, предназначенный для передачи по I^2C, из него также читается байт, принятый по I^2C. Таким образом, когда «ведущий» начнет обмен по шине, ему будет передано текущее содержимое регистра I2CDAT «ведомого», которое к этому моменту должно быть заранее записано целевой программой.

Чтение или запись регистра I2CDAT автоматически сбрасывает флаг прерывания от модуля интерфейса I^2C – бит I2CI.

Режим «программный ведущий» устанавливается путем установки бита I2CM специального регистра I2CCON. По определению, работа МК в этом режиме реализуется средствами целевой программы с минимальным участием аппаратных ресурсов интерфейса. В этом режиме программа должна организовывать обмен на уровне битов. Для того чтобы сконфигурировать линию SDATA как выход, перед выдачей через нее данных «ведомому», необходимо предварительно установить бит MDE регистра I2CCON. Для выдачи на линию SDATA логического уровня его значение следует записать в бит MDO регистра I2CCON. Линия SCLOCK в режиме «программный ведущий» всегда является выходом. Для выдачи на линию SCLOCK логического уровня его значение следует записать в бит MCO регистра I2CCON.

Перед приемом данных через линию SDATA бит MDE должен быть сброшен, что конфигурирует контакт SDATA как вход. Чтение состояния линии SDATA во время приема осуществляется путем чтения бита MDI регистра I2CCON при наличии тактового импульса на линии SCLOCK. Данные захваты-

ваются в бит MDI по нарастающему фронту этого импульса.

Таким образом, в режиме «программный ведущий» для организации обмена по шине требуется путем программных манипуляций с битами MDE, MDO, MCO и MDI специального регистра I2CCON формировать на линиях SDATA и SCLOCK нужным образом синхронизированные сигнальные последовательности данных и синхросигнала. Эти последовательности должны включать условия «START» и «STOP», адреса «ведомого» устройства, биты подтверждения, собственно данные и синхроимпульсы.

В режиме «программный ведущий» прерывания от модуля I2C не генерируются, поскольку моменты начала и конца процедуры обмена по шине определяются только целевой программой.

Аппаратное построение модуля интерфейса I^2C накладывает на его работу в режиме «программный ведущий» следующие функциональные ограничения:

– «ведущий» микроконвертор не распознает ситуацию «растягивания синхронизации», когда «ведомый» удерживает линию SCLOCK в низком уровне. Для устранения этого недостатка производитель рекомендует задействовать при обмене дополнительную линию – любой цифровой вход «ведущего», извне соединенный с линией SCLOCK, через который программа может читать текущее состояние линии SCLOCK;

– «ведущий» микроконвертор при наличии на шине I^2C нескольких ведущих не распознает конфликтных ситуаций и не поддерживает арбитраж при доступе к линии SDATA. Для устранения этого недостатка производитель рекомендует задействовать при обмене дополнительную линию – любой цифровой вход «ведущего», извне соединенный с линией SDATA, через который программа может читать текущее состояние линии SDATA.

Режим «аппаратный ведомый» устанавливается путем сброса бита I2CM специального регистра I2CCON. По умолчанию после сброса модуль SPI/I2C МК находится с выбранным интерфейсом I^2C именно в этом режиме (SPE=0, I2CM=0). «Ведомое» устройство по умолчанию ожидает выдачи на шину условия «START» от «ведущего». В регистр I2CADD «ведомого» после сброса записывается его адрес устройства (по умолчанию – 055h). Когда логика интерфейса «ведомого» определяет наличие на шине корректного условия «START», бита R/W, а также адреса, совпадающего с текущим содержимым регистра I2CADD «ведомого», происходит генерация прерывания от модуля I2C (устанавливается бит I2CI регистра I2CCON). Если это прерывание разрешено установкой бита ESI регистра IEIP2, то управление в программе будет передано по адресу его вектора – 003Bh. В очистке флага I2CI средствами программы обработки прерывания нет необходимости, так как при любом обращении программы к регистру I2CDAT с целью его записи или чтения бит I2CI будет автоматически аппаратно сброшен (программная же очистка I2CI переведет логику интерфейса I^2C в первоначальное состояние ожидания выдачи условия «START»). В регистре I2CDAT к моменту генерации прерывания будет находится переданный «ведущим» 7-битный адрес «ведомого» и бит R/W. Бит R/W, кроме то-

го, аппаратно скопируется в бит I2CTX регистра I2CCON, что дает возможность целевой программе определить алгоритм ее дальнейших действий. В случае, если I2CTX окажется установлен, т. е. «ведущий» ожидает передачи ему данных, «ведомому» следует произвести передачу путем программной записи подлежащего передаче байта в регистр I2CDAT. Если I2CTX окажется сброшен, т. е. «ведущий» продолжает передачу, то переданный им байт данных снова окажется в регистре I2CDAT. После окончания приема опять будет сгенерировано прерывание от модуля I2C – установится бит I2CI регистра I2CCON.

Аппаратная логика «ведомого» будет удерживать линию SCLOCK в низком уровне до тех пор, пока программой не будет очищен бит I2CI. Таким образом осуществляется «растягивание синхронизации», которое обеспечивает запрет передачи «ведущему» устройству до наступления готовности «ведомого» к приему.

Аппаратные средства интерфейса I^2C не предоставляют возможности различить прерывание по приему условия «START» с адресом «ведомого» и прерывания по приему байта данных. Такая возможность должна быть реализована программно путем слежения за последовательностью процедуры обмена. Бит флага прерывания I2CI устанавливается всякий раз, когда принят или передан полный байт данных с последующим битом подтверждения ACK. Если байт завершается битом неподтверждения NACK, прерывание генерироваться не будет. При распознавании на шине условии «STOP» логика интерфейса I^2C сбрасывается в первоначальное состояние ожидания выдачи условия «START».

Аппаратное построение модуля I2C накладывает на его работу в режиме «аппаратный ведомый» следующие функциональные ограничения:

– «ведомый» микроконвертер не распознает адрес общего вызова для всех устройств на шине («ceneral call address») – 0000000;

– «ведомый» микроконвертор не может обрабатывать условие «повторный START», использование которого допускается стандартом I^2C;

– «ведомый» микроконвертер не поддерживает режим 10-битной адресации.

1.14. Порты ввода-вывода

В этой и последующих двух главах дается краткое описание ряда периферийных программно доступных узлов МК. Вся эта периферия по функциональному назначению и способам управления является полностью совместимой с микроконтроллерами семейства 8051.

МК использует для обмена данными с внешними устройствами четыре параллельных порта ввода-вывода, имеющие стандартные номера с 0 по 3. В дополнение к общим функциям ввода-вывода некоторые порты могут работать с внешней памятью данных и программ, а отдельные линии других портов имеют альтернативные функции обслуживания внутренней периферии устройства. В общем случае, когда альтернативная функция данной конкретной линии порта МК разрешена, эту линию нельзя использовать как линию ввода-вывода общего назначения.

Порт 0 является 8-битным двунаправленным портом ввода-вывода,

имеющим линии с открытым стоком. Управление состоянием порта осуществляется путем записи в специальный регистр P0 (адрес 80h). Если в бит регистра P0 записать логическую единицу, то соответствующая этому биту линия порта 0 будет выходом с открытым стоком и «плавающим» уровнем собственного напряжения. В таком состоянии линии порта 0 можно использовать в качестве входов с высоким входным импедансом. Для корректной передачи линиями порта 0 высокого логического уровня при использовании их в качестве выходов необходимо подключение к этим линиям внешних резисторов, «подтягивающих» линии к «плюсу» источника питания. Помимо функций ввода-вывода общего назначения порт 0 мультиплексирован с младшими восемью разрядами магистрали адресов и данных, используемой при обращении к внешней памяти программ или данных. При работе в режиме обращения к внешней памяти порт использует внутренние схемы «подтягивания» к «плюсу» питания, если требуется генерировать в магистраль высокие логические уровни.

Порт 1 также является 8-битным портом, управляемым путем записи в специальный регистр P1 (адрес 90h). Линии порта 1 разделяются на две группы. P1.0 и P1.1 являются двунаправленными цифровыми линиями ввода-вывода с внутренними резисторами, «подтягивающими» к «плюсу» питания. Если в биты P1.0 и P1.1 регистра P1 записаны логические «единицы», на соответствующих им выходах порта 1 будут высокие уровни, обусловленные наличием «подтягивающих» резисторов. В таком состоянии эти линии можно использовать и как входы, однако, если источником внешнего сигнала низкого уровня является просто подключение к общему проводу, то через внутренние «подтягивающие» резисторы во внешние цепи из МК потечет ток. Если в биты P1.0 и P1.1 регистра P1 записаны логические «нули», то на соответствующим выходах порта 1 будут низкие уровни. В таком состоянии эти выходы способны принимать втекающий ток 10 мА, в то время как прочие линии этого и других портов способны принимать втекающий ток только до 1,6 мА.

Оставшиеся линии порта 1 (P1.2 – P1.7) можно программно сконфигурировать только как аналоговые входы (входы АЦП), аналоговый выход (выход ЦАП) или цифровые входы. По умолчанию после сброса эти линии конфигурируются как аналоговые входы, т. е. логические «единицы» записываются в соответствующие биты регистра P1. При записи логического «нуля» в бит регистра P1 соответствующая ему линия порта 1 конфигурируется как цифровой вход с высоким входным импедансом.

В табл. 1.24 указаны альтернативные функции линий порта 1.

Таблица 1.24

Альтернативные функции линий порта P1

Линия порта	**Альтернативная функция**
P1.0	T2 (внешний вход таймера-счетчика 2)
P1.1	T2EX (вход захвата/перезагрузки таймера-счетчика 2)

Порт 2 является двунаправленным портом с внутренними «подтягиваю-

щими» к «плюсу» питания резисторами. Порт управляется путем записи в специальный регистр P2 (адрес A0h). При записи в биты регистра P2 логических «единиц» на соответствующих им линиях порта 2 будут высокие уровни, обусловленные наличием «подтягивающих» резисторов. В таком состоянии эти линии можно использовать как входы, однако, если источником внешнего сигнала является просто подключение к общему проводу, то через внутренние «подтягивающие» резисторы во внешние цепи из МК потечет ток. Если в биты регистра P2 записаны логические «нули», то на соответствующих выходах порта 2 будут низкие уровни. Помимо функций ввода-вывода общего назначения порт 2 содержит старший байт адреса при выборке команды из внешней памяти программ и средний и старший байты адреса при организации доступа к внешнему 24-битному пространству памяти данных.

Порт 3 является двунаправленным портом с внутренними «подтягивающими» к «плюсу» питания резисторами. Порт управляется путем записи в специальный регистр P3 (адрес B0h). При записи в биты регистра P3 логических «единиц» на соответствующих им линиях порта 3 будут высокие уровни, обусловленные наличием «подтягивающих» резисторов. В таком состоянии эти линии можно использовать как входы, однако, если источником внешнего сигнала является просто подключение к общему проводу, то через внутренние «подтягивающие» резисторы во внешние цепи из МК потечет ток. Если в биты регистра P3 записаны логические «нули», то на соответствующих выходах порта 3 будут низкие уровни.

В табл. 1.25 указаны альтернативные функции линий порта 3.

Альтернативные функции линий P1.0, P1.1 и линий порта 3 можно активизировать, только если в соответствующие биты регистров P1 и P3 записаны логические «единицы». В противном случае на линии установится уровень логического «нуля».

Таблица 1.25

Альтернативные функции линий порта P3

Линия порта	**Альтернативная функция**
P3.0	RXD (входная линия последовательного асинхронного интерфейса UART) или вход/выход последовательных данных в Режиме 0.
P3.1	TXD (выходная линия UART) или выход последовательных синхросигналов в Режиме 0.
P3.2	INT0/ (вход вешнего прерывания 0)
P3.3	INT1/ (вход вешнего прерывания 1)
P3.4	T0 (внешний вход таймера-счетчика 0)
P3.5	T1 (внешний вход таймера-счетчика 1)
P3.6	WR/ (строб записи внешней памяти данных)
P3.7	RD/ (строб чтения внешней памяти данных)

1.15 Таймеры-счетчики, совместимые с семейством 8051

МК имеет в своем составе три 16-битных таймера-счетчика 0, 1 и 2. Аппаратная логика таймеров-счетчиков включена в состав устройства для того, чтобы освободить ядро от необходимости программной эмуляции таймеров-счетчиков при отсчетах временных интервалов. Каждый ТС состоит из пары 8-разрядных регистров: THx и TLx, где x = 0, 1, 2. Все три ТС можно программно сконфигурировать для работы либо в качестве таймеров, т. е. измерителей временных интервалов, либо в качестве счетчиков внешних событий.

В режиме «таймер» регистр TLx таймера-счетчика инкрементируется в каждом машинном цикле, т. е. его можно рассматривать как счетчик машинных циклов. Поскольку машинный цикл состоит из двенадцати периодов тактовой частоты, то максимальная счетная частота составит 1/12 тактовой частоты.

В режиме «счетчик» регистр TLx таймера-счетчика инкрементируется по отрицательному перепаду внешнего сигнала на соответствующем входе МК – T0, T1 или T2. В этом режиме состояние этого входа опрашивается в фазе S5P2 каждого машинного цикла. Когда в результате опроса выявляется высокий уровень на входе в одном цикле и низкий в следующем за ним, таймер-счетчик инкрементируется. Новое подсчитанное значение появится в регистре в фазе S3P1 машинного цикла, следующего за циклом, в котором был определен факт счетного перепада. Поскольку весь процесс занимает два машинных цикла (24 периода тактовой частоты), то максимальная счетная частота составит 1/24 тактовой частоты. На скважность импульсов внешних сигналов формальных ограничений не накладывается, но для того, чтобы гарантировать, что логический уровень внешнего сигнала будет захвачен хотя бы один раз до того, как произойдет его изменение, он должен сохранять свое состояние, по меньшей мере, в течение одного полного машинного цикла. Напоминаем, что тактовая частота ядра задается путем установки комбинации битов CD0 – CD2 в специальном регистре PLLCON.

Программное конфигурирование и управление режимами таймеров-счетчиков осуществляется путем записи в три специальных регистра: TMOD, TCON – регистры контроля и управления таймеров-счетчиков 0 и 1; T2CON – регистр контроля и управления ТС 2.

TMOD (регистр управления ТС 0 и 1)

Gate	C/T/	M1	M0	Gate	C/T/	M1	M0

Адрес 89h, значение после сброса 00h, битовая адресация отсутствует. Назначение битов регистра TMOD описано в табл. 1.26.

Таблица 1.26

Назначение битов специального регистра TMOD

Биты	Имя	Описание		
7	Gate	Бит управления стробированием Таймера 1. Устанавливается программой пользователя для разрешения работы ТС 1 только в случае, если на входе INT1/ высокий уровень и бит управления TR1 установлен. Сбрасывается программой пользователя для разрешения работы ТС 1 в случае, если бит управления TR1 установлен.		
6	C/T/	Бит выбора режима «таймер» или «счетчик» для ТС 1. Устанавливается программой для выбора режима «счетчик» (поступление счетных импульсов с ножки Т1). Сбрасывается программой для выбора режима «таймер» (работа от внутренних системных синхроимпульсов).		
5	M1	Бит 1 выбора режима работы ТС 1 (используется совместно с битом M0).		
4	M0	Бит 0 выбора режима работы ТС 1.		
		M1 0 0 1 1	M0 0 1 0 1	TH1 работает как 8-битный ТС, TL1 – как 5-битный предварительный делитель частоты. 16-битный ТС. TH1 и TL1 каскадируются. Предварительный делитель частоты отсутствует. 8-битный ТС с автоперезагрузкой. В TH1 находится величина, которая будет перегружаться в TL1 всякий раз по его переполнению. ТС 1 остановлен.
3	Gate	Бит управления стробированием Таймера 0. Устанавливается программой пользователя для разрешения работы таймера-счетчика 0 только в случае, если на входе INT0/ высокий уровень и бит управления TR0 установлен. Сбрасывается программой пользователя для разрешения работы ТС 0 в случае, если бит управления TR0 установлен.		
2	C/T/	Бит выбора «таймер» или «счетчик» для таймера-счетчика 0. Устанавливается программой для выбора режима «счетчик» (поступление счетных импульсов с ножки Т0). Сбрасывается программой для выбора режима «таймер» (работа от внутренних системных синхроимпульсов).		
1	M1	Бит 1 выбора режима работы ТС 0.		
0	M0	Бит 0 выбора режима работы ТС 0.		
		M1 0 0 1 1	M0 0 1 0 1	TH0 работает как 8-битный ТС. TL0 – как 5-битный предварительный делитель частоты. 16-битный ТС. TH0 и TL0 каскадируются. Предварительный делитель отсутствует. 8-битный ТС с автоперезагрузкой. В TH0 находится величина, которая будет перегружаться в TL0 всякий раз по его переполнению. TL0 – обычный 8-битный ТС, управляемый стандартным образом битами управления ТС0. TH0 – только таймер, управляемый битами управления ТС 1.

TCON (регистр управления ТС 0 и 1)

TF1	TR1	TF0	TR0	IE1[1]	IT1[1]	IE0[1]	IT0[1]

[1]Указанные биты не используются для управления ТС 0 и 1, а используются для слежения за уровнями внешних сигналов на ножках INT0/ и INT1/.

Адрес 88h, значение после сброса 00h, битовая адресация имеется. Назначение битов регистра TCON описано в табл. 1.27.

Таблица 1.27

Назначение битов специального регистра TCON

Биты	Имя	Описание
7	TF1	Флаг переполнения ТС 1. Устанавливается аппаратно по переполнению ТС 1. Сбрасывается аппаратно, когда программный счетчик (PC) переходит на подпрограмму обслуживания соответствующего прерывания.
6	TR1	Бит управления работой ТС 1. Устанавливается целевой программой для включения ТС 1. Сбрасывается целевой программой для выключения ТС 1.
5	TF0	Флаг переполнения ТС 0. Устанавливается аппаратно по переполнению ТС 0. Сбрасывается аппаратно, когда программный счетчик (PC) переходит на подпрограмму обслуживания соответствующего прерывания.
4	TR0	Бит управления работой ТС0. Устанавливается целевой программой для включения ТС 0. Сбрасывается целевой программой для выключения ТС 0.
3	IE1	Флаг внешнего прерывания 1 (INT1/). Устанавливается аппаратно по отрицательному перепаду или по низкому уровню внешнего сигнала (в зависимости от состояния бита IT1) на ножке INT1. Сбрасывается аппаратно при переходе программного счетчика на подпрограмму обслуживания соответствующего прерывания и только в случае, если прерывание активируется по перепаду. Если прерывание активируется по уровню, то флаг управляется внешним источником запроса, а не внутренней аппаратной логикой.
2	IT1	Бит выбора режима активирования внешнего прерывания 1. Устанавливается программой пользователя для активирования прерывания по отрицательному перепаду (т. е. по переходу «1–0»). Сбрасывается программой пользователя для активирования прерывания по уровню (низкому).
1	IE0	Флаг внешнего прерывания 0 (INT0/). Устанавливается аппаратно по отрицательному перепаду или по низкому уровню внешнего сигнала (в зависимости от состояния бита IT0) на ножке INT0. Сбрасывается аппаратно при переходе программного счетчика на подпрограмму обслуживания соответствующего прерывания и только в случае, если прерывание активируется по перепаду. Если прерывание активируется по уровню, то флаг управляется внешним источником запроса, а не внутренней аппаратной логикой.
0	IT0	Бит выбора режима активирования внешнего прерывания 0. Устанавливается программой пользователя для активирования прерывания по отрицательному перепаду (т. е. по переходу «1–0»). Сбрасывается программой пользователя для активирования прерывания по уровню (низкому).

Каждый ТС состоит из двух 8-разрядных регистров. Их можно использовать как независимые регистры или как один объединенный 16-разрядный регистр в зависимости от выбранного режима ТС.

TH0/TL0 (старший и младший регистры данных ТС 0)

Адрес 8Ch/8Ah, значение после сброса 00h/00h, битовая адресация отсутствует.

TH1/TL1 (старший и младший регистры данных ТС1)

Адрес 8Dh/8Bh, значение после сброса 00h/00h, битовая адресация отсутствует.

Режимы работы таймеров-счетчиков 0 и 1 идентичны, если обратное не оговаривается особо, поэтому, несмотря на то, что ниже пойдет речь только о режимах работы ТС 0, все сказанное следует относить и к ТС 1.

Режим 0 (режим 13-разрядного ТС)

В режиме 0 таймер-счетчик 0 имеет разрядность 13 и состоит из 8-разрядного счетчика (TH0) с 5-разрядным предварительным делителем (5 младших битов TL0) на входе. Структурная схема ТС 0 в режиме 0 приведена на рис. 1.24. Как только счетчик TH0 переполняется (переходит из состояния «FFh» в состояние «00h»), устанавливается флаг переполнения – бит TF0 регистра TCON, который может использоваться для генерации запроса на прерывание. Счетные импульсы попадают на вход предделителя счетчика в случае, если установлен бит TR0 регистра TCON и либо сброшен бит Gate регистра TMOD, либо на ножку INT0/ (P3.2) подан извне высокий уровень. При сброшенном бите Gate можно разрешать счет с входа INT0/, что позволяет измерять длительность положительного импульса внешнего сигнала. Старшие три бита регистра TL0 в этом режиме не определены и должны игнорироваться программой. Установка бита TR0 не очищает регистры TH0 и TL0.

Режим 1 (режим 16-разрядного ТС)

Режим 1 подобен режиму 0, за исключением того, что ТС 0 имеет разрядность 16. Структурная схема ТС 0 в режиме 1 приведена на рис. 1.25.

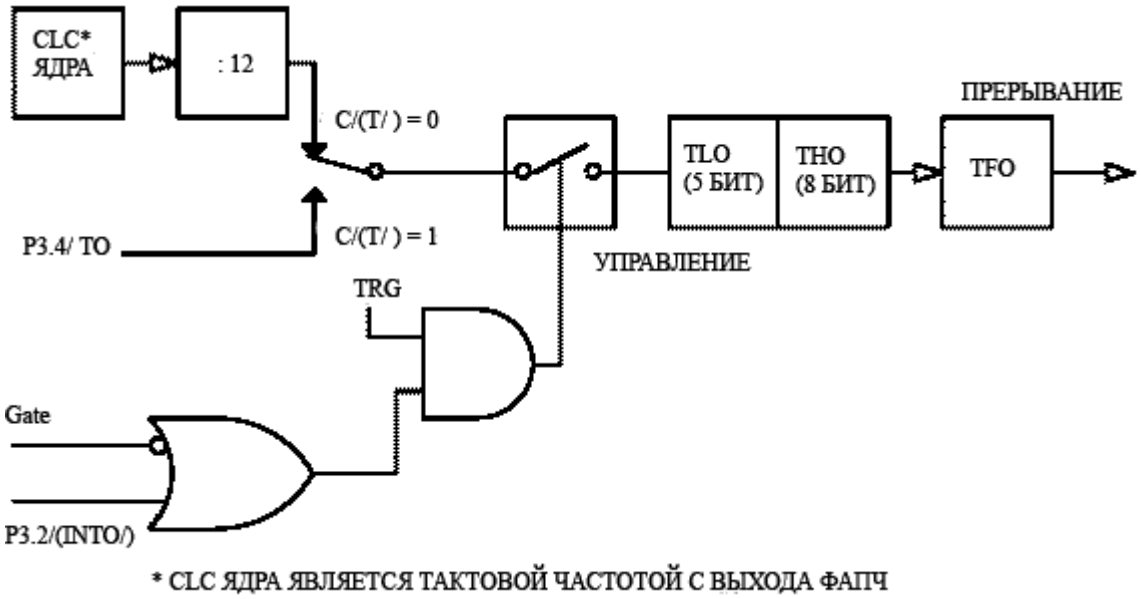


Рис. 1.24. ТС 0 в режиме 0

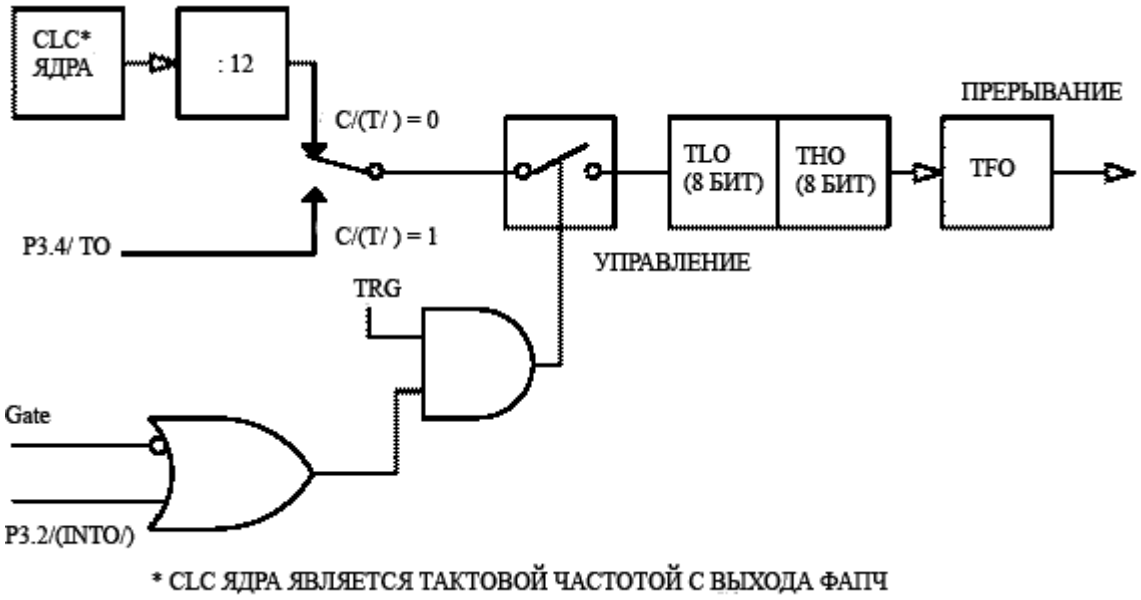


Рис. 1.25. ТС 0 в режиме 1

Режим 2 (режим 8-разрядного ТС с автозагрузкой)

В режиме 2 таймер-счетчик 0 имеет разрядность 8 и схему автоматической перезагрузки, как это показано на рис. 1.26. При переполнении TL0 происходит не только установка бита TF0, но и производится перезагрузка TL0 содержимым TH0, который может быть загружен заранее программно. В результате перезагрузки содержимое TH0 не изменяется.

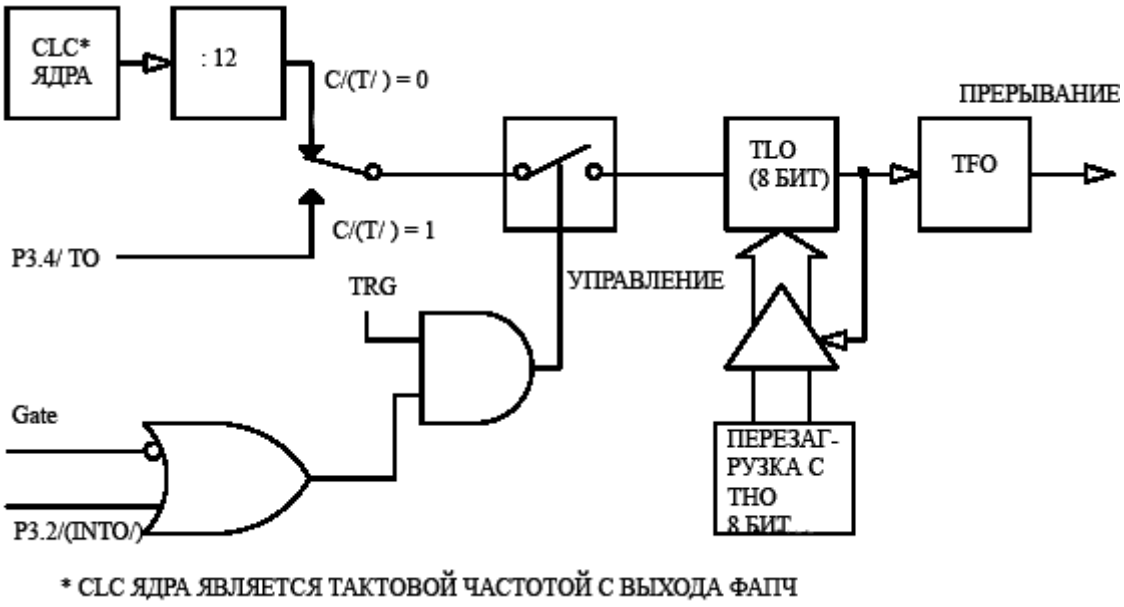


Рис. 1.26. ТС 0 в режиме 2

Режим 3 (режим двух 8-разрядных ТС)

В режиме 3 таймер-счетчик 0 и таймер-счетчик 1 функционируют различным образом. ТС 1 в режиме 3 остановлен и может использоваться, например, для хранения в своих регистрах TH1, TL1 двух байтов данных. В других режимах он переходит в подобное состояние при сбросе бита TR1. ТС 0 в режиме 3 организован как два отдельных несвязанных счетчика TL0 и TH0, как это показано на рис. 1.27. Счетчик TL0 использует для управления биты, относящиеся к таймеру-счетчику 0: C/T/, Gate, TR0, вход INT0/ и TF0. Счетчик TH0 может подсчитывать только машинные циклы и использует для управления биты, относящиеся к таймеру-счетчику 1: TR1 и TF1. Таким образом, при переполнении TH0 генерируется прерывание от ТС 1. Режим 3 предназначен для целевых приложений, требующих наличия дополнительного 8-разрядного таймера или счетчика. Когда ТС 0 находится в режиме 3, ТС 1 можно отключить или использовать в любом другом режиме как таймер или как счетчик с той только особенностью, что он не будет генерировать прерывания по переполнению. Кроме того, его можно использовать в качестве управляемого генератора скоростей обмена последовательного интерфейса UART.

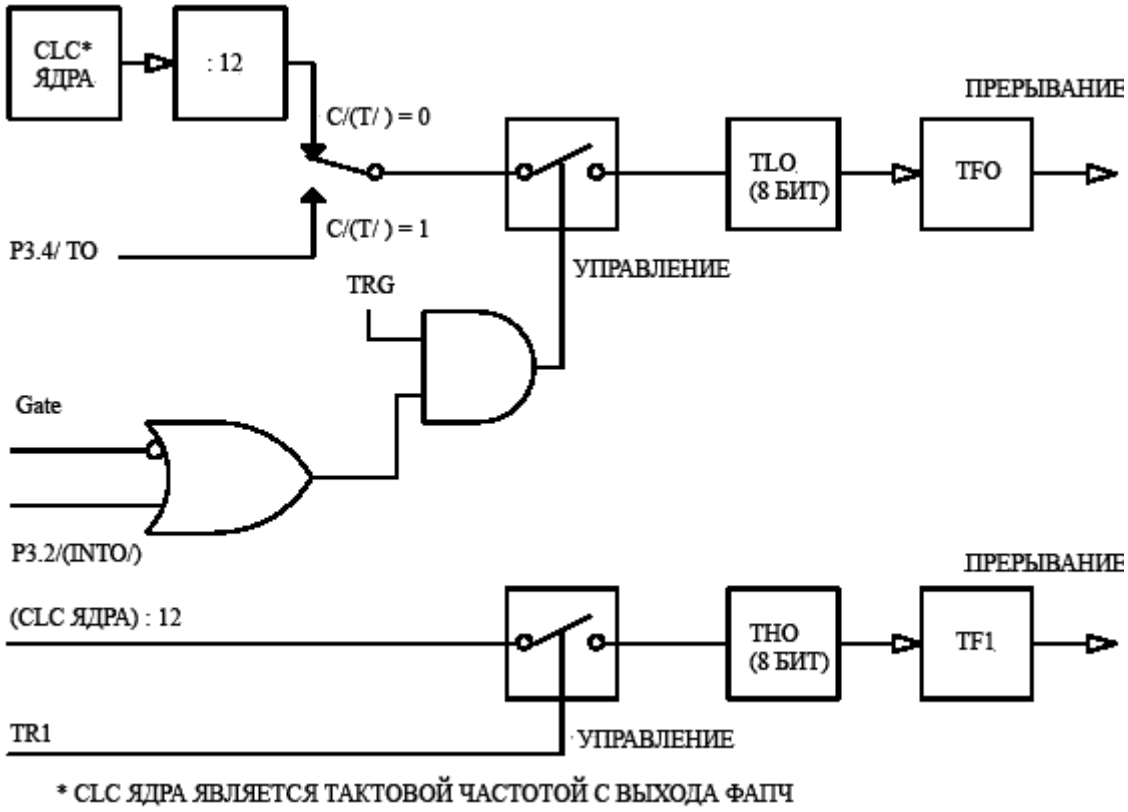


Рис. 1.27. ТС 0 как два отдельных несвязанных счетчика TL0 и TH0

T2CON (регистр управления ТС 2)

TF2	EXF2	RCLK	TCLK	EXEN2	TR2	CNT2	CAP2

Адрес C8h, значение после сброса 00h, битовая адресация имеется. Назначение битов регистра T2CON описано в табл. 1.28.

Таблица 1.28

Назначение битов специального регистра T2CON

7	TF2	Флаг переполнения ТС 2. Устанавливается аппаратно по переполнению таймера-счетчика 2. TF2 не будет устанавливаться, если либо бит RCLK, либо бит TCLK = 1. Сбрасывается программой пользователя.
6	EXF2	Внешний флаг ТС 2. Устанавливается аппаратно, когда происходит либо захват, либо перезагрузка по отрицательному перепаду на ножке T2EX при EXEN2=1. Сбрасывается программой пользователя.
5	RCLK	Бит разрешения синхросигнала приемника. Устанавливается пользователем для разрешения использования импульсов переполнения ТС 2 в качестве синхроимпульсов приемника последовательного порта в Режимах 1 и 3. Сбрасывается пользователем для разрешения использования импульсов переполнения ТС 1 в качестве синхроимпульсов приемника.

Окончание табл. 1.28

Биты	Имя	Описание
4	TCLK	Бит разрешения синхросигнала передатчика. Устанавливается пользователем для разрешения использования импульсов переполнения ТС 2 в качестве синхроимпульсов передатчика последовательного порта в Режимах 1 и 3. Сбрасывается пользователем для разрешения использования импульсов переполнения ТС 1 в качестве синхроимпульсов передатчика.
3	EXEN2	Флаг разрешения внешней операции для ТС 2. Устанавливается пользователем для разрешения захвата или перезагрузки по отрицательному перепаду на ножке T2EX, если ТС 2 не используется как источник синхросигналов для последовательного порта. Сбрасывается пользователем с целью, чтобы игнорировались все сигналы на ножке T2EX.
2	TR2	Бит пуска/останова ТС 2. Устанавливается пользователем для запуска ТС 2. Сбрасывается пользователем для остановки ТС 2.
1	CNT2	Бит выбора режима ТС 2: режим «таймер» или режим «счетчик». Устанавливается пользователем для установки режима «счетчик» (внешние счетные импульсы подаются на ножку T2). Сбрасывается пользователем для установки режима «таймер» (входной синхросигнал подается с вычислительного ядра).
0	CAP2	Бит выбора режима ТС 2: режим «захват» или режим «автозагрузка». Устанавливается пользователем для установки режима «захват» по отрицательному перепаду на ножке T2EX при условии, что EXEN2=1. Сбрасывается пользователем для установки режима автозагрузки по переполнению ТС 2 или по отрицательному переходу на ножке T2EX при EXEN2=1. В случае, если либо RCLK=1, либо TCLK=1, бит CAP2 игнорируется и ТС 2 автоматически загружается по переполнению.

ТС 2 имеет две пары связанных с ним 8-битных регистров данных. Они используются как регистры данных ТС и как регистры захвата-перезагрузки:

TH2/TL2 (старший и младший регистры данных ТС 2)

Адрес CDh/CCh, значение после сброса 00h/00h, битовая адресация отсутствует.

RCAP2H/RCAP2L (старший и младший регистры захвата-перезагрузки ТС 2)

Адрес CBh/CAh, значение после сброса 00h/00h, битовая адресация отсутствует.

Режимы работы ТС 2 выбираются путем установки комбинаций битов регистра T2CON, как это показано в табл. 1.29.

Таблица 1.29

Режимы работы таймера-счетчика 2,
выбираемые битами специального регистра T2CON

RCLK (или) TCLK	CAP2	TR2	Режим
0	0	1	16-битный с автозагрузкой.
0	1	1	16-битный с автозахватом.

Окончание табл. 1.29

RCLK (или) TCLK	CAP2	TR2	Режим
1	X	1	Генератор синхросигналов для последовательного порта.
X	X	0	Выключен.

Режим 16-разрядного ТС с автозагрузкой

В этом режиме существуют два варианта работы ТС 2, выбираемые состоянием бита EXEN2 в регистре T2CON. Если бит EXEN2 сброшен, то таймер-счетчик 2 работает как 16-разрядный таймер или счетчик и по переполнению устанавливает флаг TF2 в регистре T2CON, а также автоматически производит загрузку в свои регистры TL2, TH2 текущего содержимого регистров RCAP2L, RCAP2H, которые ранее были загружены программой. Если бит EXEN2 установлен, то по переполнению ТС 2 не только выполняются все вышеперечисленные действия, но также в случае появления внешнего отрицательного перепада на входе T2EX, произойдет еще одна перезагрузка регистров TL2, TH2 содержимым RCAP2L, RCAP2H и будет установлен флаг EXF2 в регистре T2CON. Установка флагов TF2 и/или EXF2 приводит к генерации прерывания по переполнению ТС 2. Структурная схема таймера-счетчика 2 в описанном режиме приведена на рис. 1.28.

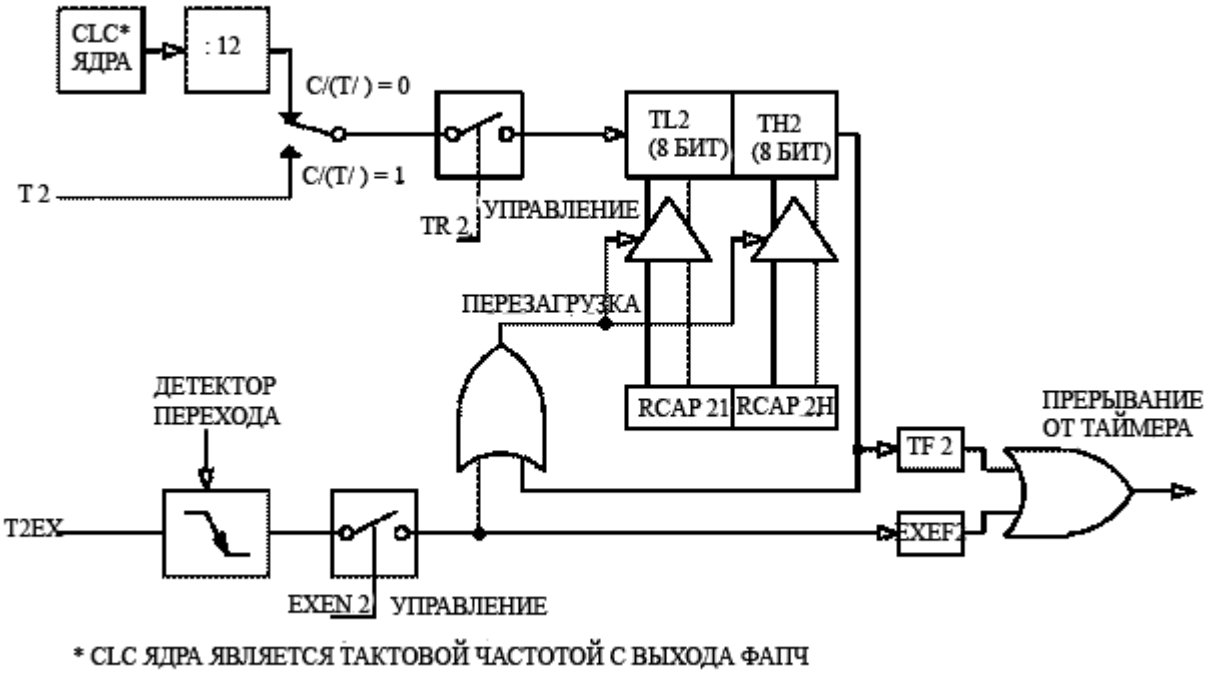


Рис. 1.28. ТС в режиме автозагрузки

Режим 16-разрядного таймера-счетчика с автозахватом

В этом режиме также существует два варианта работы, выбираемые состоянием бита EXEN2 в регистре T2CON. Если бит EXEN2 сброшен, то таймер-счетчик 2 работает как 16-разрядный таймер или счетчик и по переполнению устанавливает флаг TF2 в регистре T2CON. Если бит EXEN2 установлен, то по переполнению таймера-счетчика 2 не только устанавливается флаг TF2, но также в случае появления внешнего отрицательного перепада на входе T2EX автоматически произойдет захват (копирование) содержимого регистров таймера-счетчика 2 TL2, TH2 в регистры RCAP2L и RCAP2H соответственно. Кроме того, будет установлен флаг EXF2 в регистре T2CON. Установка флагов TF2 и/или EXF2 приводит к генерации прерывания по переполнению таймера-счетчика 2. Структурная схема ТС 2 в описанном режиме приведена на рис. 1.29.

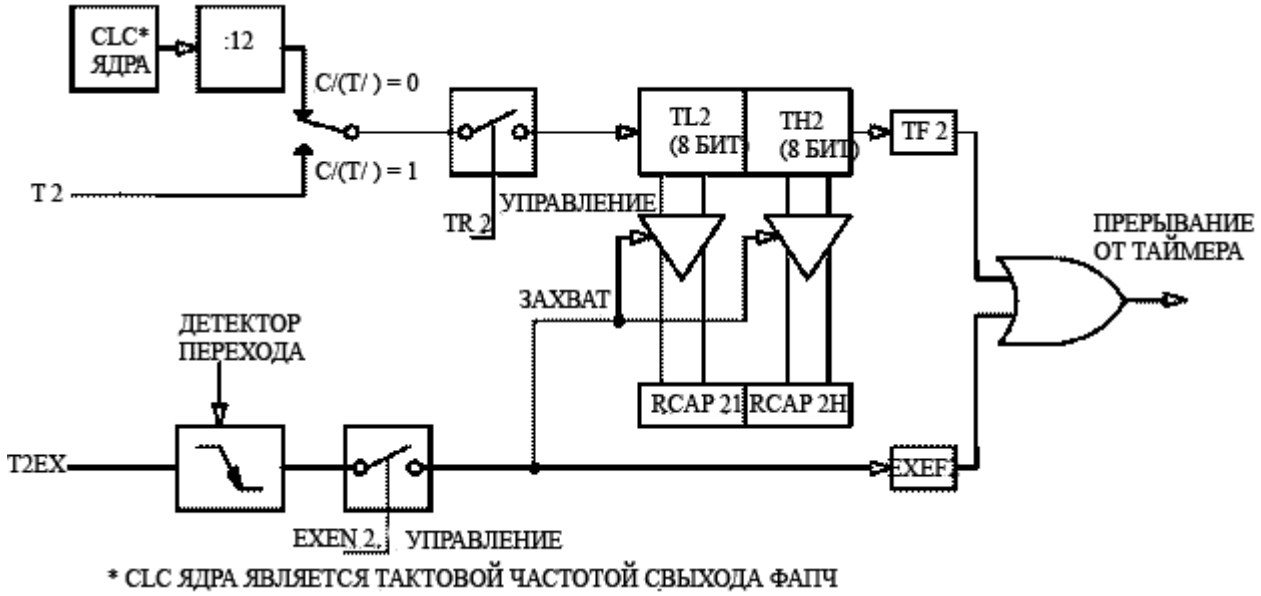


Рис. 1.29. ТС в режиме автозахвата

Режим генератора синхроимпульсов последовательного порта

Этот режим выбирается путем установки битов RCLK и/или TCLK регистра T2CON. В случае, если таймер-счетчик 2 используется для генерации синхроимпульсов последовательного порта, флаг прерываний TF2 не устанавливается. Таким образом, прерывания по переполнению ТС 2 генерироваться не будут, и нет необходимости их программно запрещать. Тем не менее, внешний отрицательный перепад на входе T2EX в случае, если бит EXEN2 установлен, вызовет установку флага EXF2 и генерацию прерывания. Таким образом, в этом режиме ножку T2EX можно использовать в качестве третьего источника внешних прерываний в дополнение к внешним прерываниям от ножек INT0 и INT1. Работа генератора синхроимпульсов последовательного порта будет описана далее.

1.16. Последовательный интерфейс UART

Последовательный порт МК является полнодуплексным, что означает, что он может одновременно передавать и принимать данные. UART МК имеет приемный буфер, что дает возможность начинать прием второго байта до того, как первый принятый байт будет считан из регистра приемника. Однако, если первый байт не считать до завершения приема второго байта, то содержимое первого байта будет потеряно. Физический интерфейс последовательного порта осуществляется через выводы микроконвертора RXD (P3.0) и TXD (P3.1). Интерфейс целевой программы к UART осуществляется через следующие специальные регистры:

SBUF (регистр буфера обмена UART)

Адрес 99h, значение после сброса 00h, битовая адресация отсутствует.

Через регистр SBUF осуществляется программный доступ к регистрам приема и передачи UART. Запись в SBUF загружает регистр передатчика, а чтение SBUF возвращает содержимое регистра приемника. Регистр передатчика и регистр приемника физически являются разными регистрами.

SCON (регистр управления последовательным портом UART)

SM0	SM1	SM2	REN	TB8	RB8	TI	RI

Адрес 98h, значение после сброса 00h, битовая адресация имеется.

Назначение битов специального регистра SCON описано в табл. 1.30.

Таблица 1.30

Назначение битов специального регистра SCON

Биты	Имя	Описание		
7	SM0	Биты выбора режима работы последовательного порта UART.		
6	SM1	Возможны следующие режимы работы:		
		SM0	SM1	Режим работы .
		0	0	Режим 0: сдвиговый регистр, фиксированная скорость обмена.
				(частота ядра/2).
		0	1	Режим 1: 8-битный UART, переменная скорость обмена.
		1	0	Режим 2: 9-битный UART, фиксированная скорость обмена.
				(частота ядра/64 или частота ядра/32).
		1	1	Режим 3: 9-битный UART, переменная скорость обмена.

Окончание табл. 1.30

5	SM2	Бит разрешения мультипроцессорной связи. Разрешает мультипроцессорную связь в Режимах 2 и 3. В Режиме 0 бит SM2 должен быть сброшен. В Режиме 1, если бит SM2 установлен, то флаг RI не будет активироваться, если не получен корректно стоп-бит. Если бит SM2 сброшен, то флаг RI будет установлен, как только будет получен байт данных. В Режимах 2 или 3 при установленном бите SM2 флаг RI не будет активироваться, если принятый девятый бит данных (RB8) равен «0». Если бит SM2 сброшен, то флаг RI будет установлен, как только будет получен байт данных.
4	REN	Бит разрешения приемника последовательного порта. Устанавливается программой пользователя для разрешения приема по последовательному порту. Сбрасывается программой пользователя для запрещения приема по последовательному порту.
3	TB8	Девятый бит при передаче по последовательному порту. Бит данных, помещенный в бит TB8, будет девятым битом данных при передаче в Режимах 2 и 3.
2	RB8	Девятый бит при приеме по последовательному порту. Девятый бит данных, принятый в Режимах 2 и 3, фиксируется в бите RB8. В Режиме 1 в бите RB8 фиксируется стоп-бит.
1	TI	Флаг прерывания от передатчика последовательного порта. Устанавливается аппаратно по концу восьмого бита в Режиме 1 и по началу стоп-бита в Режимах 1, 2, 3. Бит TI должен сбрасываться программой пользователя.
0	RI	Флаг прерывания от приемника последовательного порта. Устанавливается аппаратно по концу восьмого бита в Режиме 0 и в середине стоп-бита в Режимах 1, 2, 3. RI должен сбрасываться программой пользователя.

Модуль UART может функционировать в четырех различных режимах, выбор которых определяется установленной комбинацией битов SM0, SM1 специального регистра SCON.

Режим 0 (режим 8-разрядного сдвигового регистра)

Режим 0 выбирается путем сброса битов SM0, SM1 регистра SCON. В этом режиме данные в последовательном виде вводятся и выводятся через вывод RXD UART. Через вывод TXD выводятся синхроимпульсы сдвига. Передается или принимается восемь бит данных. Начало передачи инициируется программной инструкцией записи в регистр SBUF. Младший значащий бит (МЗР) в байте передается первым, как показано на рис. 1.30.

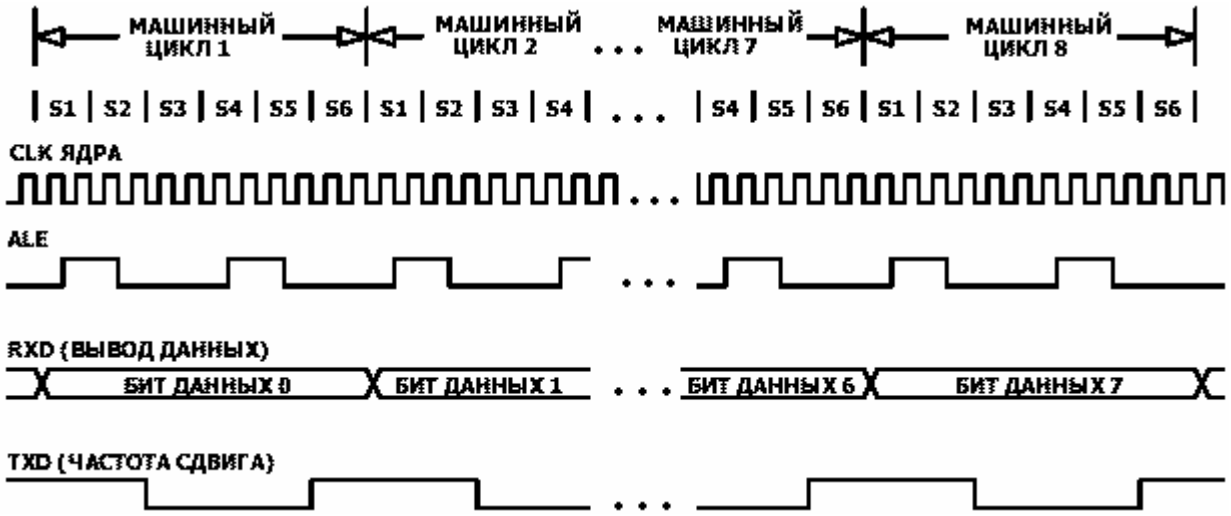


Рис. 1.30. Временная диаграмма передачи данных

Прием возможен, когда установлен бит разрешения приемника REN в регистре SCON и сброшен флаг прерывания от приемника UART RI в регистре SCON. При приеме данные поступают в устройство по линии RXD, а синхронизируется прием импульсами, генерируемыми устройством на линию TXD.

Режим 1 (режим 8-битного UART с переменной скоростью обмена)

Режим 1 выбирается путем сброса бита SM0 и установки SM1. Каждому передаваемому байту данных предшествует стартовый бит (логический «ноль»), каждая передача байта завершается стоповым битом (логическая «единица»). Скорость обмена определяется частотой переполнений таймера-счетчика 1 или таймера-счетчика 2, либо обоих этих таймеров-счетчиков в случае, когда один задает скорость передачи, а другой – приема.

Начало передачи инициируется программной инструкцией записи данных в регистр SBUF. При загрузке SBUF тем же сигналом в восьмой разряд сдвигового регистра передатчика загружается логическая «единица» (стоповый бит). Данные выводятся последовательно бит за битом до тех пор, пока на линию TXD не будет выведен стоповый бит. В этот момент автоматически установится флаг прерывания от передатчика UART TI, как показано на рис. 1.31.

Рис. 1.31. Временная диаграмма формирования прерывания по завершению передачи

Начало приема инициируется по автоматическому определению наличия отрицательного перепада на входе RXD. В предположении, что прием стартового бита прошел успешно, продолжается прием следующих битов байта данных. Стартовый бит пропускается, а восемь бит данных вводятся в сдвиговый регистр последовательного порта. Когда все восемь бит оказываются в регистре, автоматически происходят следующие события:

– восемь битов данных из сдвигового регистра приемника пересылаются в регистр SBUF;

– девятый (стоповый) бит пересылается в бит RB8 регистра SCON;

– устанавливается флаг прерывания от приемника UART RI в регистре SCON.

Перечисленные события произойдут только в том случае, если в момент генерации заключительного синхроимпульса бит RI оказывается сброшен и либо бит SM2 в регистре SCON сброшен, либо принятый стоповый бит имеет уровень логической «единицы», а SM2 установлен. Если ни одно из этих двух составных условий не выполняется, принятый байт данных безвозвратно теряется и флаг RI не устанавливается.

Режим 2 (режим 9-битного UART с фиксированной скоростью обмена)

Режим 2 выбирается путем установки бита SM0 и сброса SM1. В этом режиме UART работает как 9-битный с фиксированной скоростью обмена. Скорость обмена определяется по умолчанию частотой, равной величине «частота ядра»/64, хотя, путем установки бита SMOD в регистре PCON скорость обмена можно удвоить. Передаются и принимаются одиннадцать битов: стартовый бит (логический «ноль»), восемь бит данных, девятый (программируемый) бит и стоповый бит (логическая «единица»). Девятый бит часто используется программой в качестве бита паритета, хотя, в общем случае, его можно использовать произвольным образом, например, для передачи данных.

Начало передачи инициируется программной инструкцией записи данных в регистр SBUF. Девятый бит данных следует предварительно записать в бит TB8 регистра SCON. Когда начинается передача, восемь бит данных перегружаются из регистра SBUF в сдвиговый регистр передатчика (МЗР передается первым). Содержимое бита TB8 загружается в восьмой разряд сдвигового регистра (считая от нуля). Момент начала передачи привязан по времени к очередному генерируемому синхроимпульсу обмена. Как только стоповый бит выдается на линию TXD, автоматически устанавливается флаг TI в регистре SCON.

Алгоритм приема в режиме 2 аналогичен режиму 1. Единственное отличие заключается в том, что логика интерфейса анализирует состояние не стопового бита, а девятого принятого бита данных. Когда все восемь битов байта оказываются в сдвиговом регистре, автоматически происходят следующие события:

– восемь битов данных из сдвигового регистра приемника пересылаются в регистр SBUF;

– девятый бит данных пересылается в бит RB8 регистра SCON;

– устанавливается флаг прерывания от приемника UART RI в регистре SCON.

Перечисленные события произойдут только в том случае, если в момент генерации заключительного синхроимпульса бит RI оказывается сброшен и либо бит SM2 в регистре SCON сброшен, либо принятый девятый бит данных имеет уровень логической «единицы», а SM2 установлен. Если ни одно из этих двух составных условий не выполняется, принятый байт данных безвозвратно теряется и флаг RI не устанавливается.

Режим 3 (режим 9-битного UART с переменной скоростью обмена)

Режим 3 выбирается путем установки битов SM0 и SM1. В этом режиме UART работает как 9-битный последовательный порт с переменной скоростью обмена, которая определяется либо таймером-счетчиком 1, либо таймером-счетчиком 2. Функционирование 9-битного UART в режиме 3 идентично режиму 2, а выбор скорости обмена в режиме 3 осуществляется аналогично режиму 1.

Во всех четырех режимах работы UART передача инициируется программной инструкцией, использующей SBUF в качестве регистра назначения. Прием в режиме 0 инициируется условием – бит RI сброшен, бит REN установлен. В других режимах прием инициируется входящим стартовым битом, если бит REN установлен.

Скорость обмена в режиме 0 постоянна и определяется по формуле:

$$\text{Скорость обмена в режиме 0} = \text{«частота ядра»}/12,$$

где «частота ядра» – тактовая частота, выбранная путем установки комбинации битов CD0 – CD2 специального регистра PLLCON.

Скорость обмена в режиме 2 определяется значением бита SMOD в специальном регистре PCON. Если бит SMOD сброшен, скорость обмена составляет 1/64 частоты ядра. Если бит SMOD установлен, то – 1/32 частоты ядра, что количественно можно выразить следующей формулой:

$$\text{Скорость обмена в режиме 2} = (2^{SMOD}/64) \times \text{«частота ядра»}.$$

Скорость обмена в режиме 1 и режиме 3 определяется частотой переполнений с ТС 1 или ТС 2, либо обоих этих таймеров-счетчиков в случае, когда один задает скорость передачи, а другой – приема. В случае использования в качестве генератора скорости обмена таймера-счетчика 1 она определяется по формуле:

$$\text{Скорость обмена в режимах 1 и 3} = (2^{SMOD}/32) \times \text{«частота переполнений ТС 1»}.$$

Прерывание по переполнению ТС 1 при этом должно быть запрещено. В любом из трех рабочих режимов сам таймер-счетчик можно сконфигурировать либо как таймер, либо как счетчик. В большей части приложений его конфигурируют как 8–разрядный таймер с автозагрузкой (в специальный регистр TMOD записывается двоичный код «0010xxxx»). В этом случае скорость обмена определяется по формуле:

$$\text{Скорость обмена в режимах 1 и 3} = (2^{SMOD}/32)\times(\text{«частота ядра»}/(12\times[256-TH])),$$

где TH – десятичное значение содержимого регистра TH1.

Для генерации сравнительно невысоких скоростей обмена UART возможно использование таймера-счетчика 1 с разрешенным прерыванием по его переполнению и установленным режимом 16-битного таймера (в специальный регистр TMOD записывается двоичный код «0001xxxx»). Прерывание в этом случае используется для выполнения программной перезагрузки таймера. В табл. 1.31 приведено несколько возможных значений скоростей обмена для частот ядра 12,58 и 1,57 МГц. Следует отметить, что в случае асинхронного (старт-стопного) обмена через UART допустимая величина ошибки скорости (расхождение между стандартной и реальной скоростями) может составлять не более 5 %.

Таблица 1.31

Типовые значения скоростей обмена UART, получаемые при использовании в качестве генератора синхросигналов ТС 1

Стандартная (идеальная) скорость, бит/с	**Частота ядра, МГц**	**Значение SMOD**	**Код перезагрузки**	**Реальная скорость, бит/с**	**Ошибка, %**
9 600	12,58	1	–7 (F9h)	9 362	2,5
2 400	12,58	1	–27 (E5h)	2 427	1,1
1 200	12,58	1	–55 (C9h)	1 192	0,7
1 200	1,57	1	–7 (F9h)	1 170	2,5

При использовании в качестве генератора скорости обмена UART таймера-счетчика 2 он должен переполниться 16 раз для приема или передачи одного бита. Таким образом, скорость обмена при использовании таймера-счетчика 2 в общем случае определяется по формуле:

$$\text{Скорость обмена в режимах 1 и 3} = (1/16) \times \text{«частота переполнений таймера 2»}.$$

Если таймер-счетчик 2 используется в режиме 16-разрядного таймера с автозагрузкой, возможно получение более широкого диапазона скоростей обмена, чем при использовании таймера-счетчика 1. Таймер-счетчик 2 инкрементируется через каждые два синхроимпульса, а не один раз за машинный цикл, как таймер-счетчик 1, т. е. в шесть раз быстрее, поэтому доступны в шесть раз большие скорости обмена. Выбор таймера-счетчика 2 в качестве генератора

скорости обмена осуществляется путем установки битов TCLK и/или RCLK в регистре T2CON. Скорость обмена в этом случае определяется по формуле:

Скорость обмена в режимах 1 и 3 = «частота ядра»/(32 × [65536–(RCAP2H, RCAP2L)]),

где RCAP2H, RCAP2L – десятичные значения регистров RCAP2H, RCAP2L.

В табл. 1.32 приведено несколько возможных значений скоростей обмена, рассчитанных по указанной формуле для частот ядра 1,57 и 12,58 МГц. На рис. 1.32 приведена структурная схема использования ТС 2 в качестве генератора скорости обмена UART.

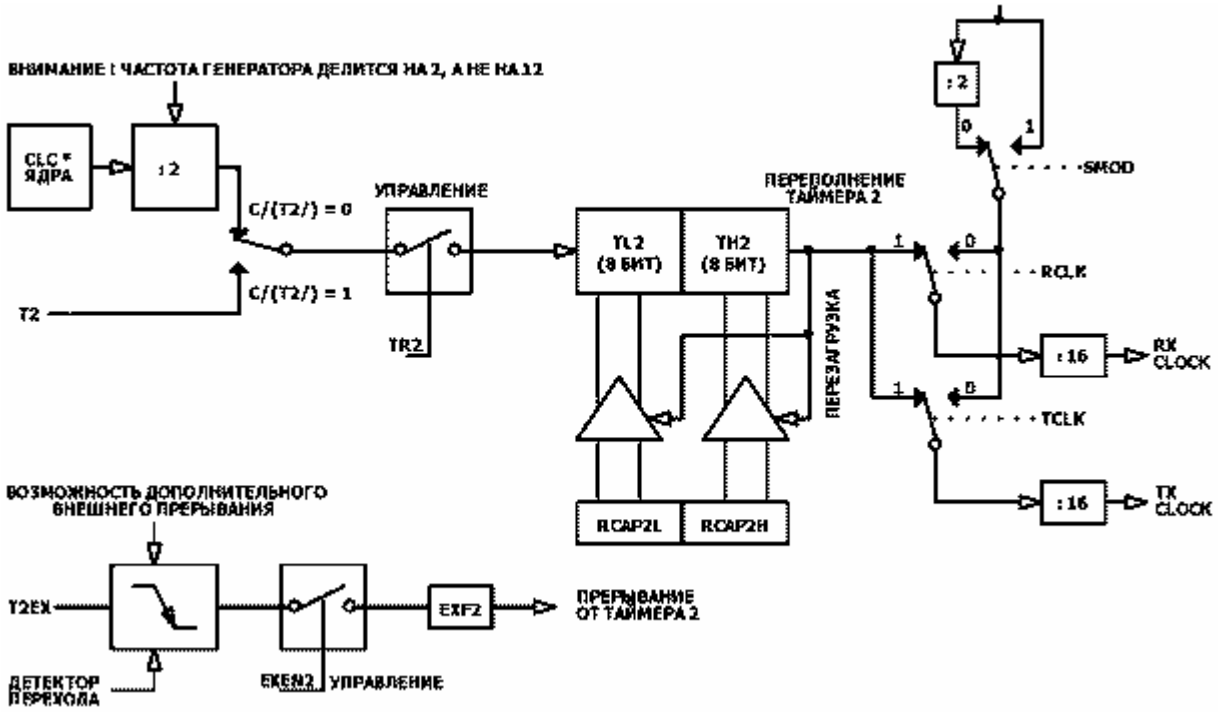


Рис. 1.32. Использования ТС 2 в качестве генератора скорости обмена UART

Таблица 1.32

Типовые значения скоростей обмена UART, получаемые при использовании в качестве генератора синхросигналов ТС 2

Стандартная (идеальная) скорость, бит/с	**Частота ядра, МГц**	**Величина RCAP2H**	**Величина RCAP2L**	**Реальная скорость, бит/с**	**Ошибка, %**
19 200	12,58	–1 (FFh)	–20 (ECh)	19 661	2,4
9 600	12,58	–1 (FFh)	–41 (D7h)	9 591	0,1
2 400	12,58	–1 (FFh)	–164 (5Ch)	2 398	0,1
1 200	12,58	–2 (FEh)	–72 (B8h)	1 199	0,1
9 600	1,57	–1 (FFh)	–5 (FBh)	9 830	2,4
2 400	1,57	–1 (FFh)	–20 (ECh)	2 458	2,4
1 200	1,57	–1 (FFh)	–41 (D7h)	1 199	0,1

1.17. Система прерываний

Внутренняя аппаратная логика прерываний МК обеспечивает обработку прерываний от двенадцати источников, которые по приоритету обслуживания делятся на два уровня. Программные настройка и конфигурирование системы прерываний осуществляются с помощью трех специальных регистров: IE, IP и IEIP2.

IE (регистр разрешения прерываний)

EA	EADC	ET2	ES	ET1	EX1	ET0	EX0

Адрес A8h, значение после сброса 00h, битовая адресация имеется. Назначение битов регистра IE описано в табл. 1.33.

Таблица 1.33

Назначение битов специального регистра IE

Биты	Имя	Описание
7	EA	Записывается пользователем для того, чтобы разрешить («1») или запретить («0») все источники прерываний.
6	EADC	Записывается пользователем для того, чтобы разрешить («1») или запретить («0») прерывание от АЦП.
5	ET2	Записывается пользователем для того, чтобы разрешить («1») или запретить («0») прерывание от таймера-счетчика 2.
4	ES	Записывается пользователем для того, чтобы разрешить («1») или запретить («0») прерывание от последовательного порта UART.
3	ET1	Записывается пользователем для того, чтобы разрешить («1») или запретить («0») прерывание от таймера-счетчика 1.
2	EX1	Записывается пользователем для того, чтобы разрешить («1») или запретить («0») внешнее прерывание 1.
1	ET0	Записывается пользователем для того, чтобы разрешить («1») или запретить («0») прерывание от таймера-счетчика 0.
0	EX0	Записывается пользователем для того, чтобы разрешить («1») или запретить («0») внешнее прерывание 0.

IP (регистр приоритета прерываний)

–	PADC	PT2	PS	PT1	PX1	PT0	PX0

Адрес B8h, значение после сброса 00h, битовая адресация имеется. Назначение битов регистра IP описано в табл. 1.34.

Таблица 1.34

Назначение битов специального регистра IP

Биты	Имя	Описание
7	–	Зарезервирован для дальнейшего использования.
6	PADC	Записывается пользователем для выбора приоритета прерывания от АЦП («1» – высокий, «0» – низкий).
5	PT2	Записывается пользователем для выбора приоритета прерывания от ТС 2 («1» – высокий, «0» – низкий).
4	PS	Записывается пользователем для выбора приоритета прерывания от последовательного порта UART («1» – высокий, «0» – низкий).
3	PT1	Записывается пользователем для выбора приоритета прерывания от ТС 1 («1» – высокий, «0» – низкий).
2	PX1	Записывается пользователем для выбора приоритета внешнего прерывания 1 («1» – высокий, «0» – низкий).
1	PT0	Записывается пользователем для выбора приоритета прерывания от ТС 0 («1» – высокий, «0» – низкий).
0	PX0	Записывается пользователем для выбора приоритета внешнего прерывания 0 («1» – высокий, «0» – низкий).

IEIP2 (регистр приоритета и разрешения вторичных прерываний)

–	PTI	PPSM	PSI	–	ETI	EPSM	ESI

Адрес A9h, значение после сброса 00h, битовая адресация имеется. Назначение битов регистра IEIP2 описано в табл. 1.35.

Таблица 1.35

Назначение битов специального регистра IEIP2

Биты	Имя	Описание
7	–	Зарезервирован для дальнейшего использования.
6	PTI	Записывается пользователем для выбора приоритета прерывания от счетчика временных интервалов (TIC) («1» – высокий, «0» – низкий).
5	PPSM	Записывается пользователем для выбора приоритета прерывания от монитора источников питания (PSM) («1» – высокий, «0» – низкий).
4	PSI	Записывается пользователем для выбора приоритета прерывания от последовательного порта SPI/I2C («1» – высокий, «0» – низкий).
3	–	Зарезервирован для дальнейшего использования, должен быть сброшен.
2	ETI	Записывается пользователем для того, чтобы разрешить («1») или запретить («0») прерывание от счетчика временных интервалов (TIC).
1	EPSM	Записывается пользователем для того, чтобы разрешить («1») или запретить («0») прерывание от монитора источников питания.
0	ESI	Записывается пользователем для того, чтобы разрешить («1») или запретить («0») прерывание от последовательного порта SPI/I2C.

Регистры разрешения прерываний используются для того, чтобы программно разрешить или запретить прерывания от конкретных источников. Регистры приоритета прерываний позволяют программно выбрать один из двух уровней приоритета обслуживания для каждого источника прерывания. Про-

грамма обработки прерывания с высоким уровнем приоритета может прервать выполнение программы обработки прерывания с низким уровнем приоритета, если последняя выполняется в момент генерации разрешенного прерывания от источника с высоким приоритетом. В случае, если два прерывания с разным приоритетом генерируются в один и тот же момент времени, прерывание с более высоким приоритетом будет обслуживаться первым. Обработка прерывания не может быть прервана обработкой другого прерывания с таким же уровнем приоритета. Если два прерывания с одинаковым уровнем приоритета генерируются одновременно, то порядок их обслуживания определяется внутренней логикой МК в соответствии с табл. 1.36, где указана очередность (внутренний приоритет) обслуживания прерываний с одинаковым уровнем приоритета.

Таблица 1.36

Очередность (внутренний приоритет) обслуживания прерываний, имеющих одинаковый уровень приоритета

Источник	**Внутренний приоритет**	**Описание**
IPSM	1 (высший)	Прерывание от монитора источников питания
WDS	2	Прерывание от сторожевого таймера
IE0	3	Внешнее прерывание 0
RDY0/RDY1	4	Прерывание от АЦП
TF0	5	Прерывание от таймера-счетчика 0
IE1	6	Внешнее прерывание 1
TF1	7	Прерывание от таймера-счетчика 1
II2C+ISPI	8	Прерывание от последовательного порта I2C/SPI
RI+TI	9	Прерывание от последовательного интерфейса UART
TF2+EXF2	10	Прерывание от таймера-счетчика 2
TII	11 (низший)	Прерывание от счетчика временных интервалов

Когда генерируется разрешенное прерывание, текущее содержимое программного счетчика автоматически помещается в стек, а значение вектора прерывания автоматически загружается в программный счетчик. Под вектором прерывания понимается адрес в программной памяти, по которому размещается первая инструкция программы обработки этого прерывания. Чаще всего, эта инструкция представляет собой безусловный переход, передающий управление на начало программного блока, осуществляющего собственно обработку прерывания. Прерывания от разных источников имеют соответственно разные вектора. Значения векторов прерываний от всех возможных источников МК приведены в табл. 1.37.

Таблица 1.37

Адреса векторов прерываний МК

Источник	**Адрес вектора**
IE0	0003h
TF0	000Bh
IE1	0013h

Окончание табл. 1.37

Источник	Адрес вектора
TF1	001Bh
RI+TI	0023h
TF2+EXF2	002Bh
RDY0/RDY1 (ADC)	0033h
II2C+ISPI	003Bh
PSMI	0043h
TII	0053h
WDS (WDIR= «1»)	005Bh

Одним из источников прерываний в МК может быть сторожевой таймер WDT. В случае, если бит WDIR в регистре управления сторожевым таймером WDCON установлен, при наступлении тайм-аута сторожевого таймера от него генерируется прерывание. Прерывание от WDT может использоваться программой, например, для ведения системного архива ошибок или для выяснения текущего состояния устройства с помощью анализа содержимого программного счетчика, стека и т. д. с целью определения причин, приведших к наступлению тайм-аута WDT. Прерывание от сторожевого таймера отличается от прерываний от других источников тем, что уровень его приоритета всегда остается высоким и не может быть изменен программно, а также тем, что его нельзя программно запретить, если сам WDT разрешен. Эти особенности гарантируют, что прерывание от WDT будет обработано при любых условиях. Необходимо отметить, что программная настройка сторожевого таймера на генерацию прерывания при наступлении тайм-аута возможна только тогда, когда значение периода WDT, задаваемое в регистре WDCON, больше нуля (табл. 1.20).

1.18. Тактовый генератор

Как уже было сказано выше, частота для тактирования МК вырабатывается с помощью встроенной системы ФАПЧ, которая умножает частоту 32 768 Гц на 384. Частота 32 768 Гц вырабатывается непосредственно внутренним генератором МК. Для обеспечения стабильной работы этого генератора на требуемой частоте необходимо подключить внешний кварцевый резонатор на частоту 32 768 Гц между выводами МК XTAL1 и XTAL2, как показано на рис. 1.33. Необходимости в подключении между этими выводами и общим проводом внешних конденсаторов в большинстве случаев нет, так как внутри МК между выводами XTAL1, XTAL2 и общим проводом уже включены конденсаторы номинальной емкостью по 12 пФ. Однако для некоторых типов резонаторов величина полной емкостной нагрузки при указанных условиях может не соответствовать рекомендациям производителей резонаторов и в этом случае для достижения устойчивой генерации допускается подключение дополнительных внешних конденсаторов.

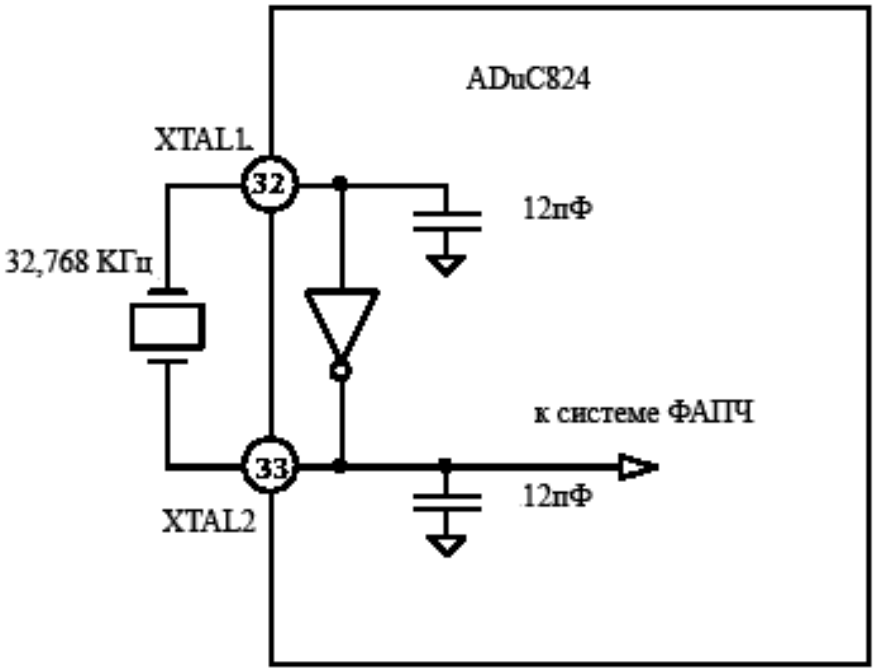


Рис. 1.33. Подключение кварцевого резонатора к МК

1.19. Подключение внешней памяти к микроконвертору

Помимо имеющейся на кристалле памяти программ и данных к МК можно подключать внешнюю память программ объемом до 64 кбайт и внешнюю память данных объемом до 16 Мбайт. Внешняя память программ может быть типа EPROM, EEPROM, FLASH и т. п., а внешняя память данных – типа SRAM (СОЗУ). Для функционирования в режиме с встроенной памятью программ на вывод EA/ МК следует подать напряжение высокого уровня, а для функционирования в режиме с внешней памятью программ – низкого. Когда на вывод EA/ подан высокий уровень, выполнение целевой программы после сброса начинается с адреса 0 адресного пространства размером 8 191 байт встроенной Flash/EE-памяти программ. Когда на вывод EA/ подан низкий уровень, выполнение целевой программы начинается с адреса 0 адресного пространства внешней памяти программ. Очевидно, что обращение к адресам больше 1FFFh (8191) возможно только при функционировании МК в режиме с внешней памятью программ.

Внешняя память программ, если она имеется, должна подключаться к МК так, как показано на рис. 1.34. Шестнадцать линий ввода-вывода портов 0 и 2 в этом случае используются в качестве смешанной шины данных и адресов при обращении к внешней памяти программ. Линии порта 0 образуют 8-разрядную двунаправленную мультиплексированную шину данных и адресов, которая яв-

ляется частью этой 16-разрядной шины. При выдаче из МК адреса программной памяти на линии порта 2 выдается содержимое старшего байта программного счетчика (PCH), а на линии порта 0 – младшего (PCL). Затем значение порта 0 «защелкивается» генерируемым МК сигналом строба ALE (Address Latch Enable) в регистре-защелке адреса. После этого порт 0 переходит в «плавающее» состояние в ожидании поступления из внешней памяти программ байта считываемых данных.

Под байтом данных здесь понимается байт кода программы, содержащейся во внешней памяти. Чтение байта данных из программной памяти МК производится в момент выдачи по линии PSEN/ МК стробирующего сигнала чтения. Адреса внешней программной памяти всегда имеют разрядность 16, даже когда размер используемой внешней памяти меньше, чем 64 кбайт. При функционировании МК в режиме с внешней памятью программ порты 0 и 2 нельзя использовать в качестве портов ввода-вывода общего назначения, однако, возможно их использование для обращения к внешней памяти данных. Доступ к внешней памяти программ и внешней памяти данных осуществляется через одну и ту же шину, однако с точки зрения программы память программ и память данных совершенно не связаны друг с другом. Например, МК может выполнять операцию чтения или записи над внешней памятью данных, выполняя в то же самое время инструкции из внешней памяти программ.

На рис. 1.35 показана функциональная схема подключения к МК внешнего ОЗУ объемом до 64 кбайт. Такая организация интерфейса является стандартной для любого микропроцессора, совместимого с семейством 8051.

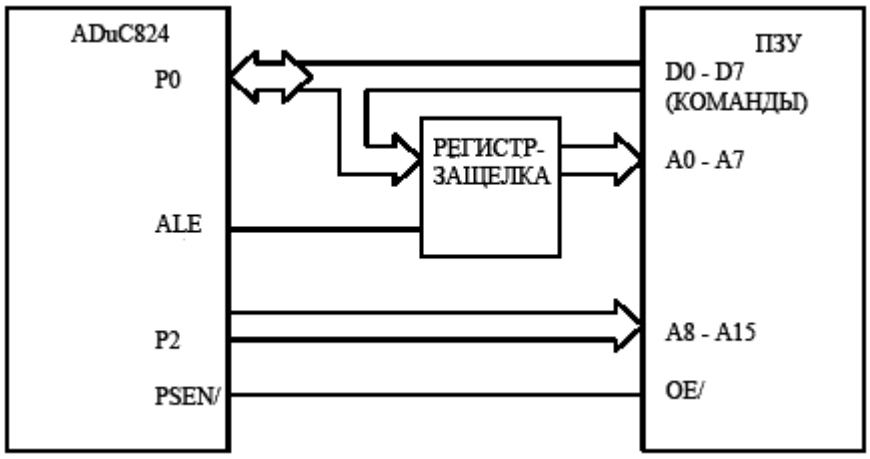


Рис. 1.34. Типовая схема подключения внешнего ОЗУ

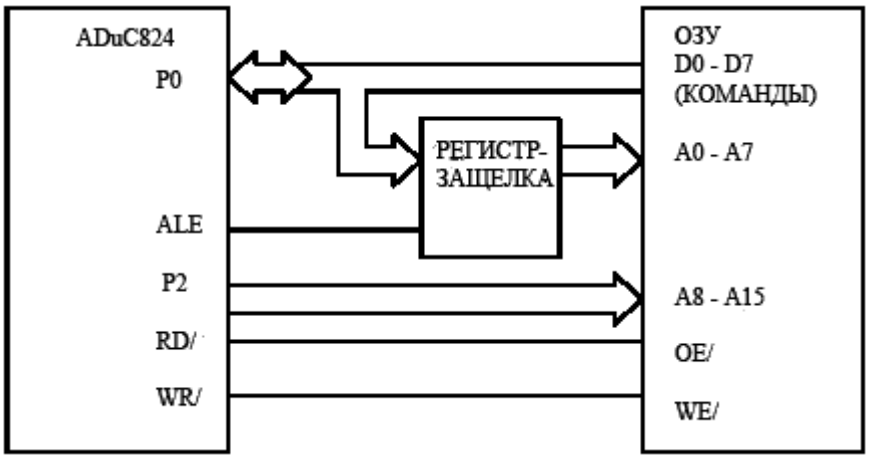


Рис. 1.35. Внешнее ОЗУ менее 64 кбайт

Когда необходимо подключить к МК внешнее ОЗУ объемом больше, чем 64 кбайт, то потребуется добавить в схему дополнительный 8-разрядный регистр-защелку, как это показано на рис. 1.36. В этом случае МК может адресовать внешнее ОЗУ объемом до 16 Мбайт.

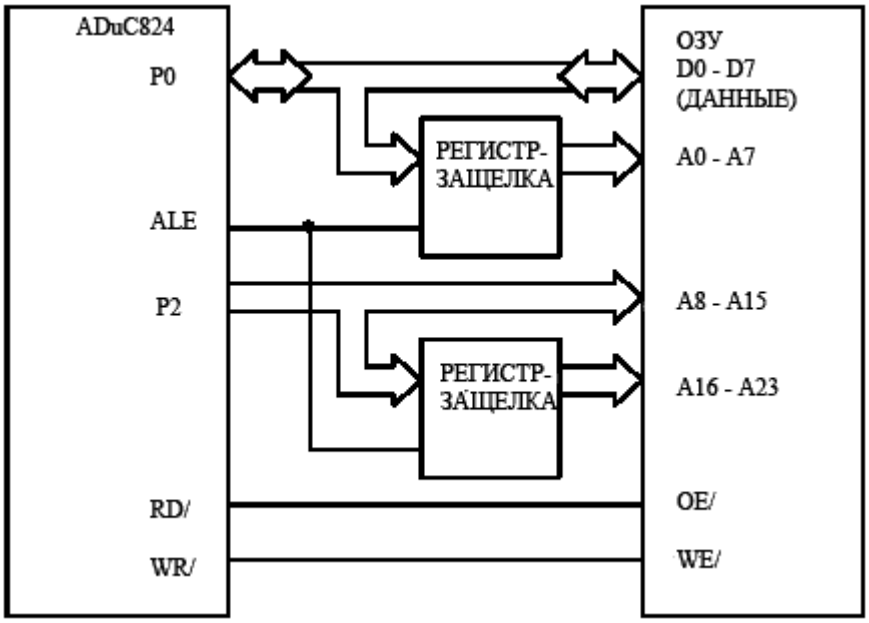


Рис. 1.36. Расширения адресного пространства

При функционировании МК с внешней памятью данных порт 0 работает как порт двунаправленной мультиплексированной шины адресов и данных. В качестве адресов порт выдает на шину содержимое младшего байта указателя данных (DPL), которое «защелкивается» импульсом ALE до того, как на шину микроконвертором (при выполнении записи) или внешним ОЗУ (при выполнении чтения) будет выдан байт данных. Порт 2 выдает на свою шину байт страницы указателя данных (DPP), который «защелкивается» сигналом ALE, после чего через порт выдается старший байт указателя данных (DPH). Если шина

порта 2 не содержит регистра-защелки, то ОЗУ игнорирует (пропускает без фиксации) DPP. Таким образом осуществляется стандартный для семейства 8051 доступ к внешней памяти данных объемом 64 кбайт. При наличии на шине регистра-защелки осуществляется доступ к внешней памяти данных объемом до 16 Мбайт. Временные диаграммы сигналов с указанием значений временных параметров для циклов записи и чтения при обращении к внешней памяти программ и данных МК представлены в приложениях 5–7.

1.20. Аппаратная организация сброса при включении питания

Для осуществления сброса МК при начальной выдаче на него питающего напряжения производитель микросхемы рекомендует использовать внешнюю схему выработки сигнала сброса при включении питания (POR – power on reset). К этой схеме предъявляются следующие требования. Она должна удерживать вход RESET МК в активном (высоком) состоянии, пока напряжение источника питания DVDD меньше, чем 2,5 В. После достижения напряжением DVDD уровня в 2,5 В схема должна снимать сигнал сброса RESET не раньше, чем через 10 мс, причем все это время напряжение DVDD не должно опускаться ниже 2,5 В. В момент снятия сигнала сброса значение напряжения DVDD должно быть не менее 2,7 В. Внешняя схема POR должна оставаться работоспособной при понижении напряжения питания DVDD до уровня 1,2 В. Временная диаграмма на рис. 1.37 иллюстрирует работу схемы POR при трех различных ситуациях: включение питания (power-up), «провал» напряжения питания (brownout) и снятие питания (power-down). Лучшее решение при выборе внешней схемы сброса POR, которое удовлетворяло бы перечисленным требованиям, заключается в использовании специализированных микросхем POR, например, ADM809, ADM810 фирмы Analog Devices. Рекомендуемые производителем схемы подключения ADM810 (с высоким активным уровнем) и ADM809 (с низким активным уровнем) показаны на рис. 1.38 и рис. 1.39. Некоторые микросхемы POR с низким активным уровнем, например, ADM809 можно использовать совместно с кнопкой дополнительного ручного сброса, как это показано на рис. 1.39.

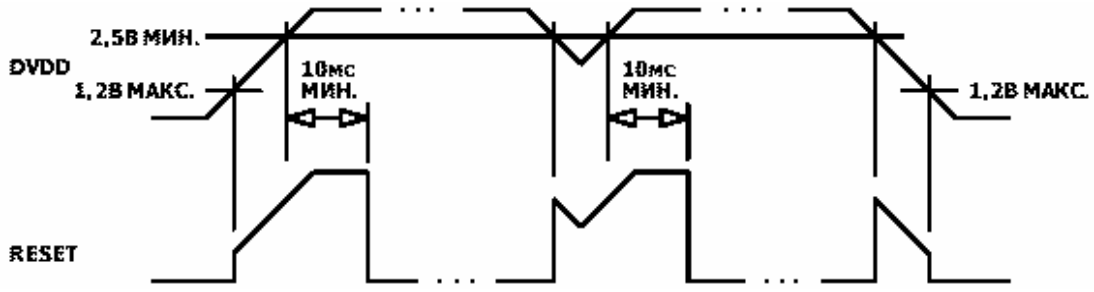


Рис. 1.37. Временная диаграмма работы POR

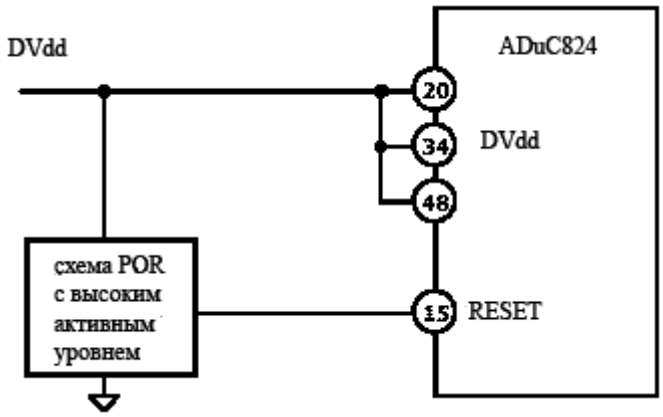


Рис. 1.38. Сброс МК по включению питания

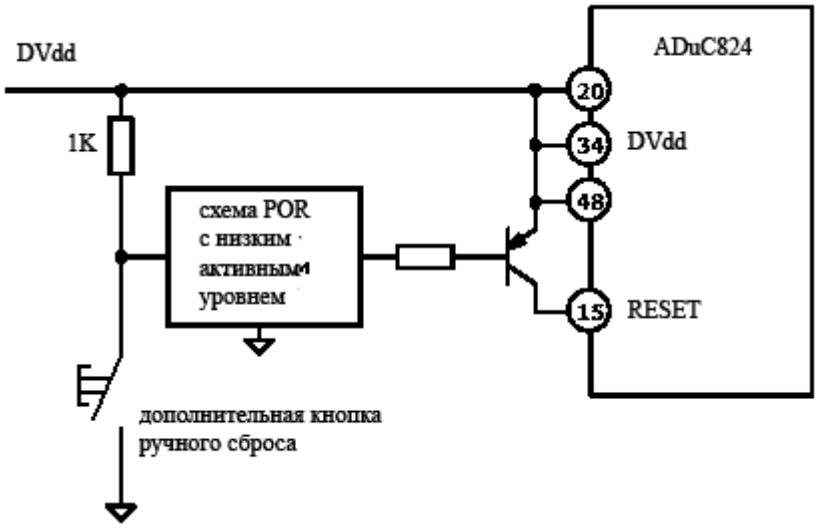


Рис. 1.39. Ручной сброс МК

1.21. Организация питания микроконвертора

Допустимые значения рабочего напряжения питания МК лежат в диапазоне от 2,7 до 5,25 В, а рекомендуемые производителем диапазоны питающих напряжений лежат в диапазонах от 2,7 до 3,6 В и от 4,75 до 5,25 В. Микросхема имеет раздельные цепи питания аналоговой (AVDD) и цифровой (DVDD) частей, что позволяет в значительной мере обеспечить отсутствие в цепи питания AVDD шумов из цепи питания DVDD, возникающих из-за наличия динамических сигналов с крутыми фронтами на цифровых линиях ввода-вывода устройства. Возможен режим работы МК с двумя различными источниками, значения напряжений которых не равны друг другу, например, при DVDD = 5 В и AVDD = 3 В. Типовая схема питания МК от двух источников показана на рис. 1.40. В качестве более дешевой альтернативы использования двух источников производитель рекомендует способ получения напряжения AVDD с низ-

ким уровнем шумов из напряжения DVDD путем включения между цепями AVDD и DVDD последовательно резистора малого сопротивления и индуктивности в виде ферритовой «бусинки», надетой на соединительный провод. Схема описанного варианта организации питания МК приведена на рис. 1.41. При таком способе получения напряжения AVDD им можно запитать и другие аналоговые узлы (ОУ, ИОН и т. п.) целевого устройства. Вне зависимости от того, используются ли два источника питания или один, в устройстве необходимо установить фильтрующие конденсаторы емкостью не менее 10 мкФ между общим проводом и каждой из цепей питания DVDD и AVDD. Помимо этого, между каждой ножкой микросхемы, через которую в нее поступают питающие напряжения AVDD или DVDD, и общим проводом необходимо установить фильтрующие конденсаторы емкостью 0,1 мкФ. Причем, места их соединения с цепями питающих напряжений должны располагаться как можно ближе к выводам микросхемы (в идеале конденсаторы должны быть подпаяны к самим ножкам), а проводники, соединяющие конденсаторы с общим проводом, должны быть как можно короче. Цепи аналоговой и цифровой «земель» ADuC824 должны соединяться между собой на плате разрабатываемого устройства только в одной точке, и в этой же точке они должны соединяться с общим проводом устройства. Подробнее об организации заземления будет рассказано ниже.

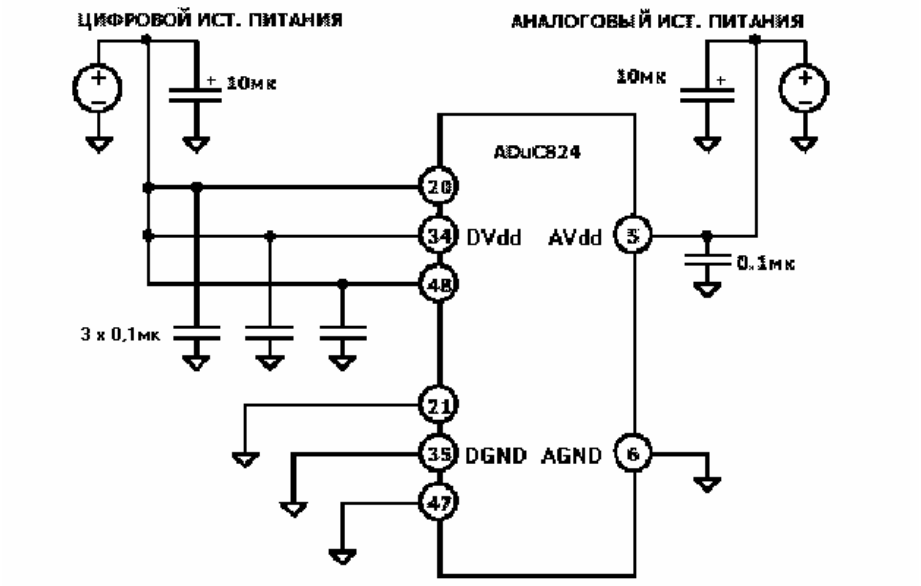


Рис. 1.40. Организация питания микроконвертора

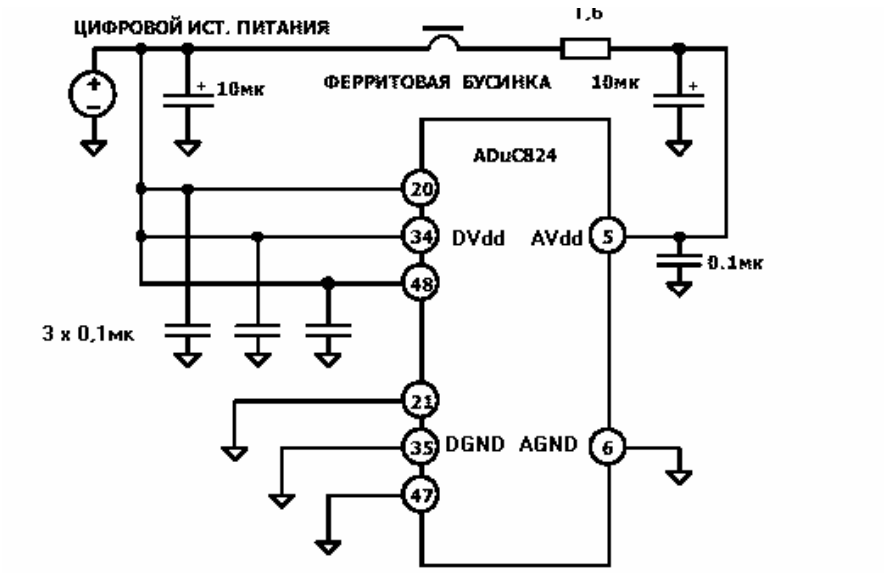


Рис. 1.41. Вариант питания МК

1.22. Режимы экономии энергопотребления

Потребляемый МК ток складывается из тока потребления вычислительного микропроцессорного ядра, которое питается от источника DVDD, и тока потребления аналоговой периферии (АЦП и ЦАП), которая питается от источника AVDD. В случае если часть аналоговой периферии в целевом приложении не используется, она может быть программно отключена и, соответственно, не будет потреблять тока. Прочая встроенная периферия (сторожевой таймер, монитор источника питания и т. д.) в совокупности потребляет ток пренебрежимо малый по сравнению с током потребления вычислительного ядра. Для того чтобы точно определить общий потребляемый МК ток в конкретном устройстве, необходимо также учесть токи, отдаваемые в нагрузки через параллельные и последовательные порты ввода-вывода, а также ток нагрузки модуля ЦАП. Кроме того, потребляемый от источника напряжения DVDD ток возрастает примерно на 5 мА при выполнении стирания и записи Flash/EE памяти.

Установка битов IDL (холостой режим) и PD (режим «питание снято») специального регистра управления питанием PCON позволяет соответственно переключить МК из нормального режима работы в холостой режим или полностью отключить от микроконвертора питание. В холостом режиме внутренний генератор частоты 32 768 Гц продолжает работать, но тактовая частота с выхода системы ФАПЧ на ядро не поступает. Внутренняя периферия в холостом режиме продолжает тактироваться и остается работоспособной. При переходе МК в холостой режим текущее состояние процессорного ядра сохраняется в стеке, а содержимое программного счетчика и других внутренних регистров фиксируется в них по состоянию на момент перехода. Линии ввода-вывода портов и выходы ЦАП также сохраняют свое состояние, а выходные линии ALE и PSEN/ переводятся в высокий уровень. МК выводится из холостого ре-

жима при поступлении любого разрешенного прерывания или после аппаратного сброса.

В режиме «снятое питание» не только не производится тактирование ядра, но и останавливается работа системы ФАПЧ. Работа внутреннего генератора либо запрещается, либо разрешается в зависимости от состояния бита управления питанием генератора OSC_PD в специальном регистре PLLCON (табл. 1.18). Работа модуля счетчика временных интервалов (TIC), получающего счетные импульсы непосредственно с выхода генератора, в режиме со снятым питанием может быть соответственно либо запрещена, либо разрешена. Работа всей внутренней периферии запрещена. Порты сохраняют свои выходные уровни, а выход ЦАП переходит в состояние с высоким выходным сопротивлением. Выходы ALE и PSEN/ переводятся в низкий уровень. Типовой ток потребления МК в режиме «снятое питание» составляет всего 5 мкА.

Вывести МК из режима «снятое питание» можно следующими способами:

1. Установить в высокий уровень сигнал на входе RESET, т. е. произвести «горячий» аппаратный сброс устройства. После сброса микроконвертор переходит в нормальный режим работы, все регистры принимают значения по умолчанию и выполнение программы начинается с вектора сброса 0000h, как только активный сигнал RESET будет снят.

2. Выключение-включение питания, т. е. «холодный» аппаратный сброс устройства. После сброса микроконвертор переходит в нормальный режим работы, все регистры принимают значения по умолчанию и выполнение программы начинается с вектора сброса 0000h, как только на устройство будет подано питание.

3. Прерывание от счетчика временных интервалов TIC. При передаче управления на начало программы обработки этого прерывания устройство переходит в нормальный режим работы и после выполнения инструкции RETI в конце программы обработки начинается выполнение инструкции, следующей за той, которая вызвала переход в режим «снятое питание».

4. Прерывание от модуля I2C или SPI. При передаче управления на начало программы обработки этого прерывания устройство переходит в нормальный режим работы и после выполнения инструкции RETI в конце программы обработки начинается выполнение инструкции, следующей за той, которая вызвала переход в режим «снятое питание». Следует отметить, что прерывание от I2C/SPI должно быть разрешено в режиме «снятое питание» путем установки бита SERIPD в специальном регистре PCON (табл. 1.4).

5. Прерывание от INT0. При передаче управления на начало программы обработки этого прерывания устройство переходит в нормальный режим работы и после выполнения инструкции RETI в конце программы обработки начинается выполнение инструкции, следующей за той, которая вызвала переход в режим «снятое питание». Следует отметить, что прерывание от INT0 должно быть разрешено в режиме «снятое питание» путем установки бита INT0PD в специальном регистре PCON (табл. 1.4).

1.23. Организация заземления и рекомендации по топологии печатной платы

При разработке практических конструкций с использованием МК для достижения максимального возможного разрешения аналого-цифровых преобразований следует, как и в случае применения любого другого высокоточного АЦП, корректно выполнить заземление в системе и правильно выбрать топологию печатной платы. Возможные схемы выполнения заземления и выбора топологии показаны на рис. 1.42.

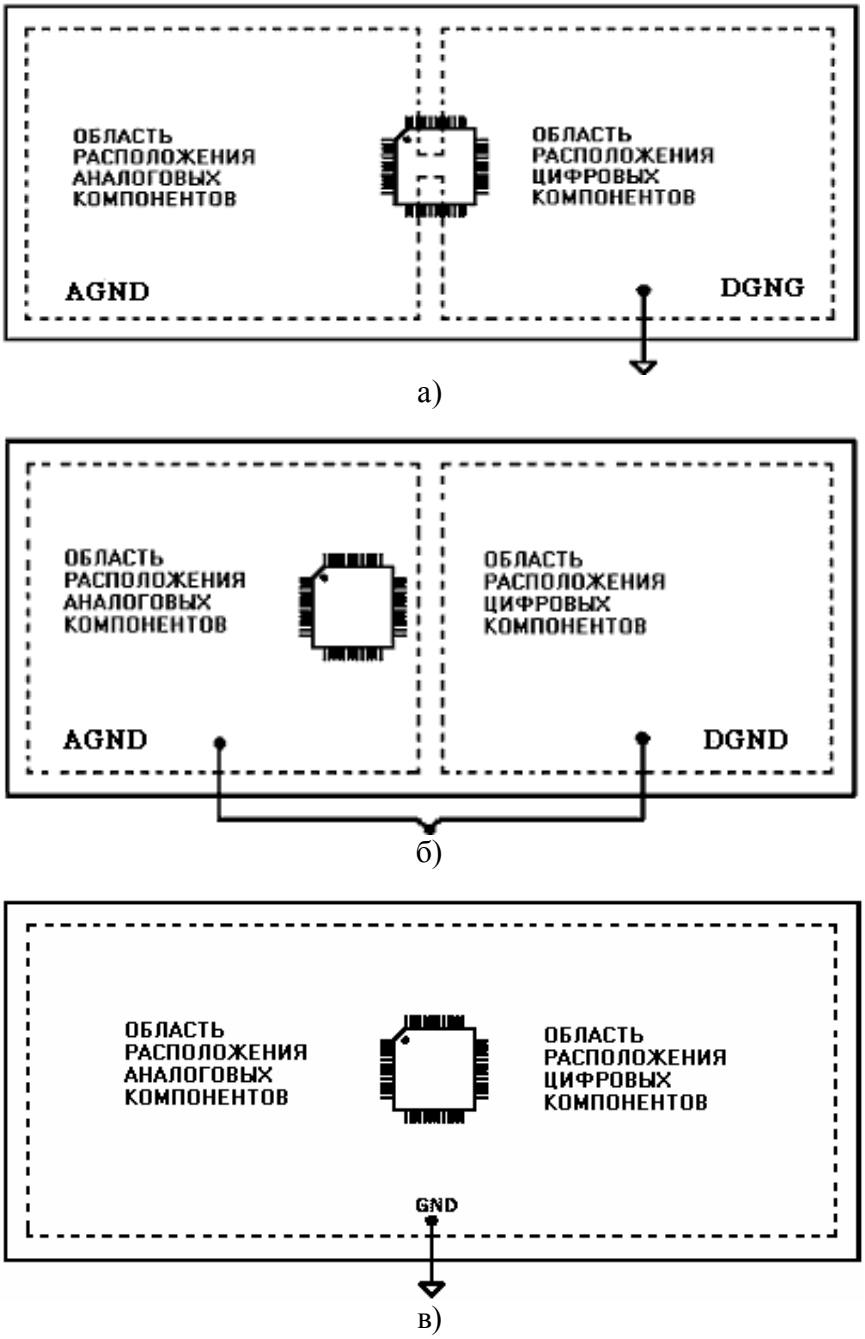


Рис. 1.42. Выполнение заземления на печатной плате

Несмотря на то, что МК имеет раздельные выводы аналоговой и цифровой «земель» (AGND и DGND), при выполнении заземления их не следует соединять соответственно с аналоговой и цифровой шинами «земли» на плате, кроме случая, когда эти две шины соединены между собой в непосредственной близости от МК, как это показано на рис. 1.42, а. В случае, если аналоговая и цифровая шины заземления соединяются между собой достаточно далеко от МК, например, на полюсе системного источника питания, выводы AGND и DGND микроконвертора следует подключать к шине аналоговой «земли», как показано на рис. 1.42, б. В системах с единственной шиной заземления следует размещать цифровые компоненты устройства отдельно от аналоговых, расположив их на разных частях платы таким образом, чтобы цифровые «земляные» (возвратные) токи не протекали вблизи протекания аналоговых «земляных» токов. Сам микроконвертор в этом случае можно разместить между цифровой и аналоговой областями платы, как показано на рис. 1.42, в. Единственную шину заземления удобно выполнить в виде сплошного проводящего слоя фольги, например, на оборотной стороне платы.

При разработке реальных устройств всегда следует помнить о том, что токи текут от источников питания в схему, а возвращаются обратно в источники по «земле». Токи, текущие от разных элементов схемы, не должны, по возможности, сливаться друг с другом до точки их втекания в шину заземления. Пути возврата всех токов должны соответствовать путям протекания этих токов к точкам назначения. Например, не следует питать элементы, находящиеся в аналоговой части платы, от цифрового источника DVDD (особенно для случая, показанного на рис. 1.42, б, так как это приведет к тому, что токи возврата в источник DVDD будут протекать через шину аналоговой «земли». Кроме того, следует, по возможности, уменьшать длину совместного пробега проводников с цифровым и аналоговым токами, расположенных, например, на разных сторонах платы. Всегда, когда это возможно, следует избегать создания больших неоднородностей на шинах «земли» (такие неоднородности формируются длинными проводниками, расположенными в одном слое), поскольку они увеличивают путь протекания возвратных токов. И наконец, следует выполнять все подключения в схеме к «земляной» шине проводниками возможно большего сечения и наименьшей длины.

Если по условиям работы целевого устройства на цифровые входы МК будут подаваться сигналы с крутыми фронтами (время нарастания или спада < 5 нс), то рекомендуется в разрыв проводников, подводимых к этим входам, установить резисторы сопротивлением 100–200 Ом, что приведет к сглаживанию фронтов входных сигналов и позволит избежать ухудшения точности работы АЦП из-за импульсных помех.

1.24. Система автоидентификации микроконвертора

В некоторых случаях желательно, чтобы целевая программа, выполняющаяся в микроконверторе, могла его идентифицировать. Например, для адаптации встроенного программного обеспечения к конкретной аппаратной платформе алгоритм должен содержать инструкции ветвления, которые в зависимости от результатов идентификации платформы передают управление в разные участки кода. МК располагает средствами обеспечения программной автоидентификации. В области SFR имеется специальный регистр CHIPID, который доступен только для чтения. Старшая тетрада этого регистра равна нулю для МК, единице для ADuC816, двум для ADuC834 и трем для ADuC836.

CHIPID (регистр кода автоидентификации)

Адрес C2h, значение всегда 0xh, битовая адресация отсутствует.

1.25. Аппаратные средства загрузки, отладки и эмуляции

Для обеспечения возможности внутрисхемной загрузки МК из персонального компьютера необходимо подключить к UART МК микросхему согласования уровней интерфейса RS-232. Схема подключения согласующей микросхемы и разъема стандарта RS-232 показана на рис. 1.43. В качестве согласующей микросхемы можно использовать устройства ADM202, MAX232.

Для установки МК в режим внутрисхемной загрузки необходимо замкнуть перемычку, соединяющую вывод PSEN/ с общим проводом через резистор сопротивлением 1 кОм, а затем произвести сброс МК. После завершения сброса устройство будет готово к загрузке новой программы. Если перемычку разомкнуть, то устройство возвратится в режим обычной работы только после проведения повторного сброса. Ножка PSEN/, вообще говоря, является выходом, но по спаду сигнала RESET, который имеет место при включении питания или при «горячем» ручном сбросе, МК производит чтение состояния этой ножки как входа. В связи с этим, необходимо заметить, что, если какая-то внешняя цепь может «подтягивать» вывод PSEN/ к низкому уровню в моменты подачи питания или сигнала сброса, то, таким образом, возможен случайный переход в режим внутрисхемной загрузки. Целевая программа в этом случае выполняться не будет. Для недопущения такой ситуации следует убедиться в отсутствии внешних сигналов, способных влиять на состояние вывода PSEN/.

С аппаратной точки зрения осуществление отладки через последовательный порт абсолютно идентично осуществлению последовательной загрузки, описанной выше. Таким образом, режим последовательной загрузки и режим отладки через последовательный порт можно рассматривать как один режим работы, используемый для двух различных целей. Отладчик, имеющийся в со-

ставе МК, является резидентным, поэтому для реализации сеанса отладки «в системе» не потребуется никакой внешней памяти.

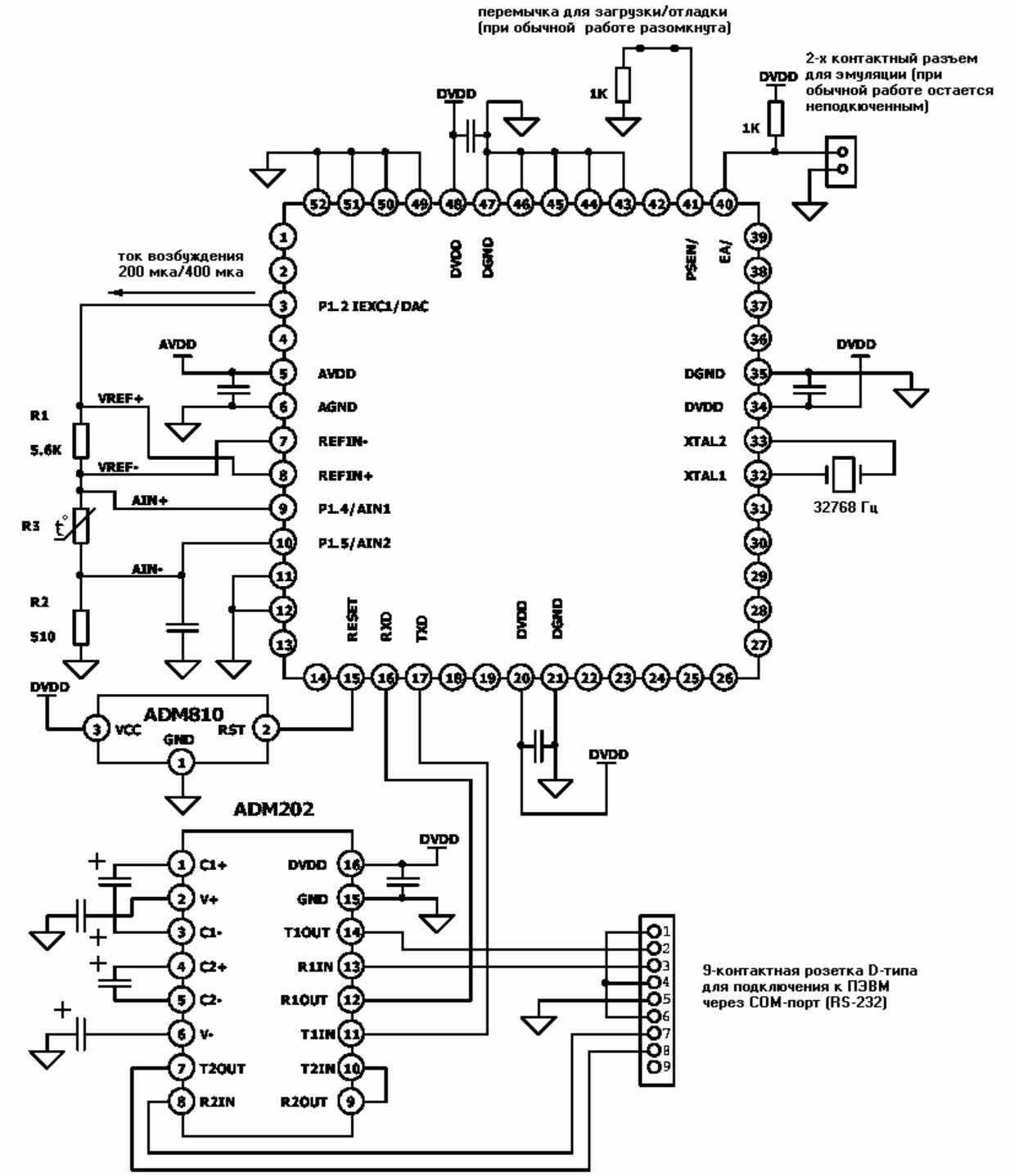


Рис. 1.43. Типовая схема включения микроконвертора

Наряду с внутрисхемным отладчиком в МК встроен специализированный контроллер внутрисхемной эмуляции (single-pin in-circuit emulation – ICE), осуществляемой через один внешний контакт – ножку EA/. При нормальной работе для выполнения программы из внутренней или из внешней памяти требуется подать на этот вывод внешний соответственно высокий или низкий ло-

гический уровень, как было описано выше. Для разрешения режима эмуляции с помощью одного внешнего контакта требуется подключить вывод EA/ к «плюсу» напряжения питания через резистор сопротивлением 1 кОм, как показано на рис. 1.43. Тогда через двухконтактный разъем (EA/, DGND), также показанный на рисунке, можно будет подключить к МК головку-пробник эмулятора. Для обеспечения совместимости с одноконтактным эмулятором фирмы «Accutron Limited» (www.accutron.com) рекомендуется использовать двухконтактный разъем с шагом 0,1 дюйма (2,54 мм) «Friction Lock» фирмы «Molex» (www.molex.com).

1.26. Типовая схема включения микроконвертора

Типовая схема включения МК приведена на рис. 1.43. На рисунке МК используется для выполнения аналогового измерения температуры с терморезистором в качестве температурного датчика. Показанную на рисунке схему подключения датчика принято называть четырехпроводной. Ее особенность заключается в том, что ток возбуждения подводится к датчику по одной паре проводов, а измеряемое напряжение снимается с датчика по другой паре, причем точки соединения одноименных проводов в парах находятся непосредственно на контактах датчика. Благодаря этому через измерительные провода течет очень малый ток и, соответственно, на них падает очень малое напряжение, что позволяет без потерь точности измерений располагать датчик на значительном расстоянии от измерителя. Внешнее дифференциальное опорное напряжение создается током, протекающим через резистор R1. Этот же ток протекает и через терморезистор R3, создавая на нем дифференциальное напряжение, пропорциональное его температуре. Это напряжение подается на положительный и отрицательный входы основного АЦП (AIN1 и AIN2) микроконвертора. Резистор R2 включен в схему для того, чтобы абсолютная величина аналогового напряжения на отрицательном входе АЦП (в данном случае на входе AIN2) была не меньше нижнего предельного значения, установленного производителем (AGND+100 мВ). Как уже отмечалось выше, изменения тока возбуждения не оказывают влияния на точность измерений, так как измеряемое напряжение на терморезисторе и опорное напряжение на резисторе R1 при изменении этого тока меняются пропорционально. Вместе с тем, резистор R1 должен иметь малое значение ТКС, чтобы свести к минимуму температурную зависимость опорного напряжения и связанную с ней ошибку АЦП.

1.27. Вопросы для самоконтроля

1. Перечислите основные параметры и характеристики микроконверторов.
2. Нарисуйте и поясните структурную схему микроконвертора.
3. В чем заключаются особенности распределения памяти МК?
4. Нарисуйте и поясните программную модель МК.
5. Каково назначение и состав регистров специальных функций?
6. Перечислите регистры специальных функций.
7. Как повлиять на частоту обновления выходных данных основного АЦП?
8. Как установить максимальное значение источника тока?
9. Объясните схему основного АЦП.
10. Чем определяется разрешающая способность основного АЦП?
11. Чем дополнительный АЦП отличается от основного?
12. Как встроенный датчик температуры связан с другими блоками МК?
13. Какие могут быть диапазоны входных напряжений АЦП?
14. Как настроить АЦП для работы с биполярными сигналами?
15. Как осуществляется калибровка АЦП?
16. Каково назначение встроенных в основной АЦП генераторов постоянного тока 100 нА?
17. Как подключить к МК внешний ИОН?
18. В каких случаях используются внешний ИОН?
19. Поясните принцип работы дельта-сигма АЦП.
20. Каково назначение суммирующего каскада в АЦП?
21. Что необходимо делать раньше – калибровку «нуля» или «верхнего предела диапазона»? Почему?
22. Чем отличается Flash-память от других типов памяти?
23. Как влияет температура на число рабочих циклов Flash-памяти?
24. Какие возможны способы загрузки программного кода в память МК?
25. Нарисуйте и поясните схему перевода МК в режим последовательной загрузки.
26. Как производится параллельное программирование МК?
27. Объясните принципы защиты Flash-памяти, используемые в МК.
28. Нарисуйте схему регистрового интерфейса массива Flash/EE-памяти данных.
29. Перечислите команды управления Flash/EE-памятью данных.
30. Охарактеризуйте встроенный ЦАП.
31. Как производится управление модулем ЦАП?
32. Как работает встроенная система ФАПЧ?
33. Как уменьшить энергопотребление МК с помощью ФАПЧ?
34. Какова минимальная рабочая частота ядра МК?
35. Зачем нужен счетчик временных интервалов NIC?
36. Нарисуйте и поясните схему модуля TIC.
37. Каково назначение сторожевого таймера?
38. Как устроен сторожевой таймер?
39. Как работает монитор источников питания?
40. Каким должно быть напряжение AVDD для правильной работы PSM?

41. Чем защищен блок PSM от коротких бросков питающих напряжений?
42. Можно ли одновременно использовать интерфейсы SPI и I^2C в МК?
43. В чем особенность интерфейса SPI?
44. Какую роль играет сигнал SS интерфейса SPI?
45. Нарисуйте временную диаграмму работы интерфейса SPI.
46. В каких режимах может работать МК по интерфейсу SPI?
47. Как функционирует интерфейс I^2C?
48. Зачем нужны подтягивающие резисторы в интерфейсе I^2C?
49. В каких случаях устанавливается бит флага прерывания I2CI?
50. Приведите характеристики портов ввода-вывода МК.
51. Перечислите альтернативные функции портов ввода-вывода МК.
52. Как активизируются альтернативные функции портов ввода-вывода?
53. Сколько таймеров-счетчиков встроено в МК?
54. Перечислите регистры управления таймерами-счетчиками.
55. Объясните суть режима 13-разрядного таймера-счетчика.
56. Как сделать таймер-счетчик с автозагрузкой?
57. Как осуществить функцию автозахвата таймером-счетчиком?
58. Что вы знаете о работе таймера-счетчика в режиме генератора синхроим-пульсов?
59. Интерфейс UART в МК полудуплексный или полнодуплексный?
60. Какие есть режимы работы интерфейса UART в МК?
61. Опишите режим работы UART с переменной скоростью обмена.
62. С какой скоростью может работать блок UART в МК?
63. Как можно использовать таймер-счетчик в качестве генератора скорости обмена по UART?
64. Какова система прерываний МК?
65. Какие вы знаете вектора прерываний МК?
66. Нарисуйте, как подключить кварцевый резонатор к МК.
67. Какой кварцевый резонатор можно использовать вместе с МК?
68. Как подключить к МК внешнюю память с параллельным интерфейсом?
69. Какой размер может иметь внешняя память с параллельным интерфейсом?
70. Нарисуйте возможные схемы сброса МК по включению питания.
71. Какие есть особенности в организации питания МК?
72. Какие есть режимы энергосбережения в МК?
73. Можно ли выключать питание АЦП при работающем ядре МК?
74. Как на печатной плате необходимо соединять аналоговую и цифровую «землю»?
75. Что хранится в регистре CHIPID?
76. Нарисуйте типовую схему включения МК.
77. Каковы функции вывода PSEN/?
78. Как подключить МК к ПК по последовательному интерфейсу?
79. Для каких целей можно использовать МК?
80. Какие вы знаете аналоги рассмотренного в учебном пособии МК?

2. Инструментальные средства поддержки ADuC824

Для использования ADuC824 в целевых проектах производителем разработан комплекс инструментальных средств поддержки разработки-отладки приложений MicroConverter QuickStart Development System, включающий программный пакет системы разработки-отладки, отладочную плату с установленной на ней микросхемой ADuC824BS и набором элементов аппаратной «обвески», блок питания платы и интерфейсный кабель связи платы с компьютером. Для запуска процесса инсталляции программного пакета системы разработки-отладки MicroConverter QuickStart Development System на жесткий диск компьютера следует запустить на исполнение файл setup.exe из пакета. По умолчанию инсталляцию рекомендуется производить в каталог C:\ADuC. Программные инструментальные средства для ADuC824 полностью совместимы с предыдущей версией QuickStart Development System, то есть если более ранняя версия пакета уже установлена на жесткий диск в каталог C:\ADuC, то более поздняя версия при запуске на инсталляцию по умолчанию обновит часть компонентов пакета в том же каталоге.

Пакет системы разработки-отладки включает следующие основные программные инструментальные средства:

– кросс-ассемблер Metalink 8051;
– последовательный загрузчик WSD;
– отладчик DeBug;
– программный симулятор ADsim;
– программный анализатор АЦП WASP;
– компилятор языка Си (с ограничением размера кода 2 кбайт).

Пакет в целом и некоторые из перечисленных программ по отдельности бесплатно доступны на сайте фирмы Analog Devices: www.analog.com/microconverter.

Ниже будут подробно рассмотрены все перечисленные программы за исключением компилятора языка Си. Причина этого кроется в невозможности в рамках одного учебного пособия изложить еще и принципы написания и оптимизации приложений на языке высокого уровня. Дело в том, что для 51-совместимых микроконтроллеров трансляция написанного на языке Си исходного текста программы приводит к затратам программной памяти, совершенно несоразмерным со степенью функционального «наполнения» алгоритма этой программы. По некоторым оценкам, реализация отдельных алгоритмов на ассемблере дает экономию в размерах объектного кода по сравнению с реализацией на языке Си почти на порядок. Для микроконтроллеров с небольшим объемом памяти программ, каковым является ADuC824, разработчик не может позволить себе такую расточительность. Кроме того, программа, написанная на языке ассемблера, работает гораздо эффективнее программы, написанной на

языке Си. Эти утверждения в меньшей степени относятся к микроконтроллерам, имеющим архитектуру с несколькими аккумуляторами (регистровый файл) как, например, устройства с ядром AVR фирмы Atmel.

2.1. Демонстрационная плата ADuC824 Evaluation Board

Принципиальная схема эволюционной платы ADuC824 и другая документация доступны на сайте http://www.analog.com. Подробное описание платы, включающее таблицы распайки разъемов и перечень комплектующих элементов приведено в [7]. Принимая во внимание сравнительно высокую стоимость платы (около $ 200) для осуществления экспериментов с МК представляется целесообразным самостоятельно изготовить ее упрощенный вариант. Часть аппаратной «обвязки» МК, установленной на фирменной эволюционной плате (внешняя оперативная память данных, внешняя энергонезависимая память команд, буферные регистры шины подключения внешней памяти, внешние операционные усилители для модуля ЦАП), не является необходимой для большинства приложений. Принципиальная схема упрощенного варианта платы, в котором отсутствуют перечисленные компоненты, и для которого далее будут рассматриваться различные демонстрационные программы, приведена на рис. 2.1.

Плата включает в себя следующие основные элементы:

- микроконвертор DD1 ADuC824BS с внешним супервизором питающего напряжения DA3 ADM709;
- узел связи с ПК образованный приемопередатчиком интерфейса RS-232 DA4 MAX323CPE и разъемом XS1;
- прецизионный источник опорного напряжения DA2 AD780 с прецизионным резистивным делителем (R4, R5) на выходе;
- стабилизатор питающего напряжения DA1 ADP667, обеспечивающий выходное стабилизированное напряжение DVDD, равное +5 В или +3,6 В.

Кроме перечисленных узлов, на плате смонтированы кнопки управления SB1–SB6, транзисторный ключ, нагрузкой которого является светодиод HL1, а также буквенно-цифровой жидкокристаллический индикатор со встроенным контроллером управления HG1. Внешние цепи и устройства, необходимые для работы целевых программ, подключаются к микроконвертору через разъемы XP1, XP4 – XP12 типа PLS с помощью соединительных проводов.

Питание эволюционной платы осуществляется от внешнего источника постоянного напряжения +7…15 В с максимальным током нагрузки 250… 300 мА. Конструктивно эволюционная плата выполнена в соответствии с приведенными выше рекомендациями (рис. 1.42, в).

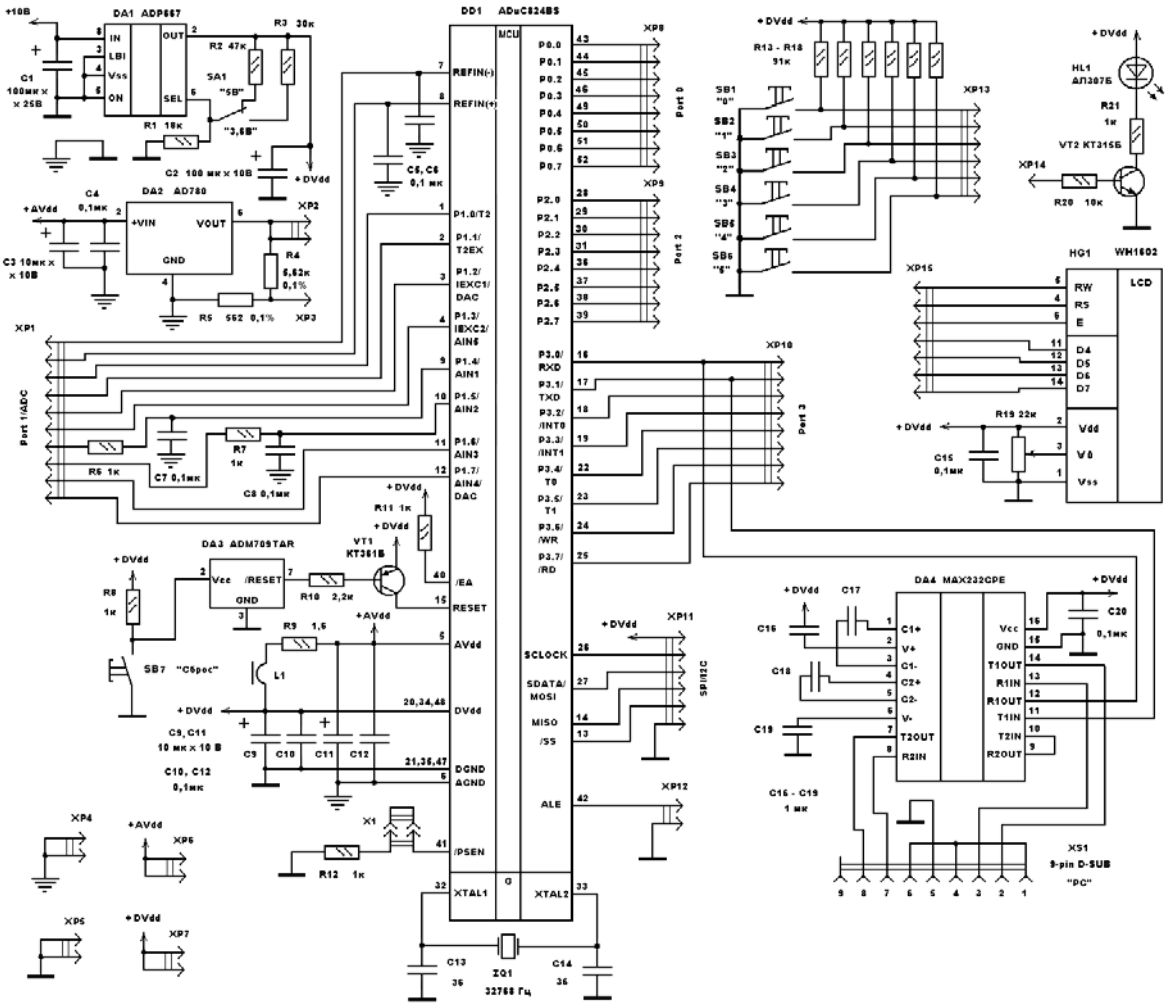


Рис. 2.1. Принципиальная электрическая схема стенда

В качестве основы для нее может быть использован кусок двустороннего фольгированного стеклотекстолита размером 150×200 мм, на верхней стороне которого смонтированы все элементы схемы, а нижняя используется в качестве общего провода. Плата должна быть разделена поперек длинной стороны на цифровую и аналоговую области, в которых расположены соответственно цифровые и аналоговые компоненты. Микроконвертор DD1 установлен посередине между двух областей. Участки с неудаленной фольгой, используемые в качестве шин цифровой и аналоговой «земель» на верхней стороне платы разделены между собой по границе областей полосой удаленной фольги. Соединение шин цифровой и аналоговой «земли» с общим проводом осуществляется перемычками из провода большого сечения через сквозные отверстия в плате.

2.2. Ассемблер Metalink 8051

Ассемблер транслирует файл исходного текста программы, подготовленный в любом текстовом редакторе в файл листинга с расширением .lst и объектный файл в шестнадцатеричном формате с расширением .hex. Файл листинга отображает результаты трансляции и возможные ошибки в исходном

тексте. Объектный файл используется непосредственно для программирования микроконвертора с помощью последовательного загрузчика.

Для трансляции файла с подготовленным исходным текстом следует двойным щелчком мыши на значке «ASM51» запустить исполняемый файл ассемблера, а затем в появившемся окне DOS указать имя файла, содержащего исходный текст, с указанием пути к нему, например, C:\PR_ADUC\TIC12\tic.asm, и нажать на клавишу «Enter». Если после завершения трансляции ассемблер выведет сообщение «ASSEMBLY COMPLETE, 0 ERRORS FOUND» (рис. 2.2), то это значит, что трансляция завершена успешно, формальных ошибок в исходном тексте программы не найдено и из файла исходного текста созданы два одноименных файла (с расширениями .hex и .lst).

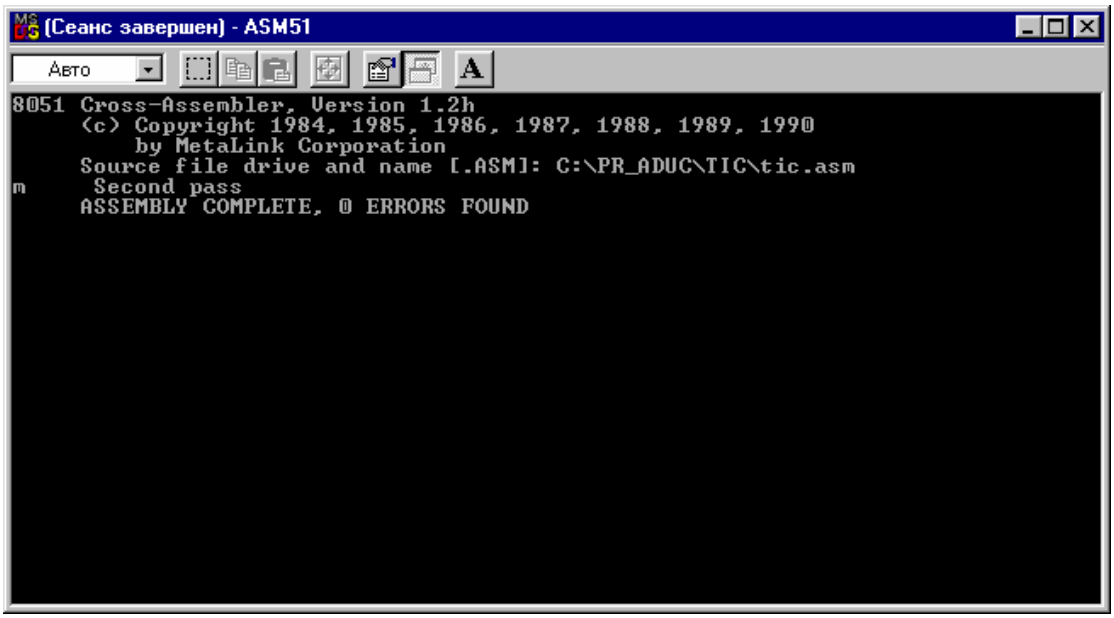


Рис. 2.2. Успешное завершение трансляции программы

Если ассемблер в сообщении указывает на наличие ошибок, найденных при трансляции, например, как показано на рис. 2.3, то необходимо исследовать файл листинга на предмет уточнения местоположения ошибок, затем исправить их в файле исходного текста, снова произвести его трансляцию и действовать таким образом до получения сообщения об отсутствии ошибок. Если ассемблер выдает сообщение, указывающее на неудачу при чтении диска A: или на фатальную ошибку при открытии файла, то это может быть связано с невозможностью найти файл, на который имеется указание о включении в файл исходного текста директивой $INCLUDE, например, $INCLUDE: mod824. В связи с этим, перед запуском трансляции следует удостовериться, что все включенные директивой $INCLUDE в исходный текст файлы находятся в одном каталоге с исполняемым файлом ассемблера asm51.exe, либо в самих директивах $INCLUDE указан путь к включенному файлу, например, $INCLUDE: C:\ADuC\Code\mod824.

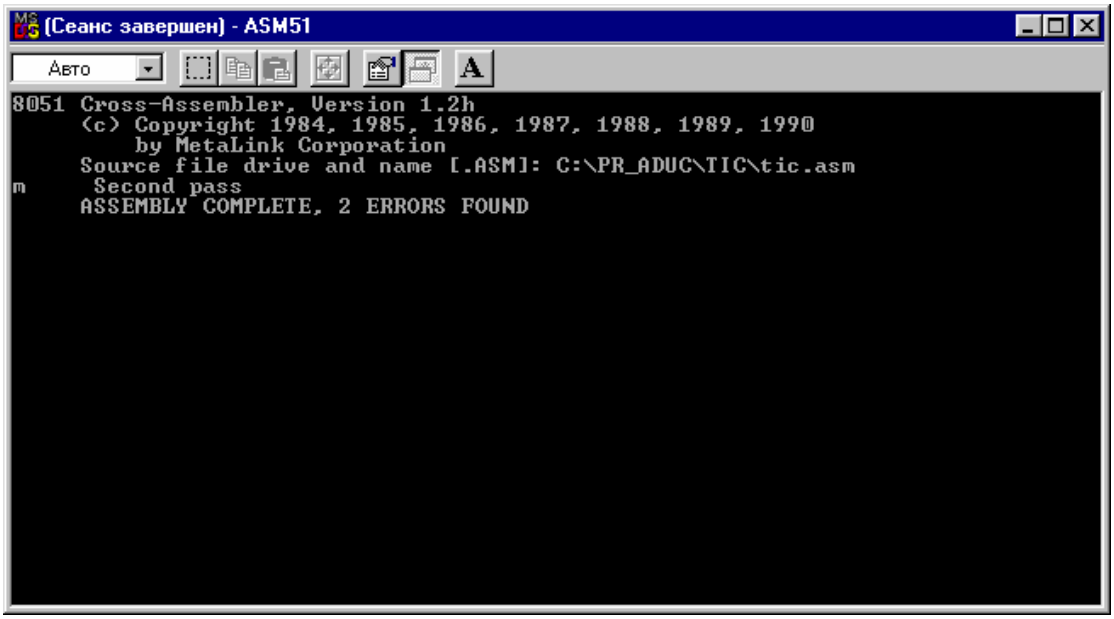


Рис. 2.3. Сообщение об ошибках трансляции

Описание лексики и директив ассемблера Metalink 8051 не рассматривается в рамках этого учебного пособия. Для ознакомления с ними рекомендуется обратиться к [8]. Этот документ поставляется в составе пакета MicroConverter QuickStart Development System. В приложении 14 приводится набор инструкций ассемблера с указанием их мнемоник, кодов операций и выполняемых действий.

2.3. Последовательный загрузчик WSD

WSD – это работающая под Windows программа, которая позволяет загрузить полученный в результате трансляции шестнадцатеричный объектный файл во FLASH-память программ микроконвертора через COM-порт компьютера. Кроме загрузки памяти программ WSD также обеспечивает загрузку FLASH-памяти данных, установку в разных комбинациях битов защиты и различных опций выполнения программы в кристалле. Перед запуском WSD необходимо с помощью стандартного интерфейсного кабеля RS-232 подключить к компьютеру отладочную плату, замкнуть на ней перемычку X1, разрешив этим последовательную загрузку, а затем включить питание платы. После запуска WSD двойным щелчком мыши на значке «WSD» программа должна обнаружить подключенную к компьютеру плату и над правым верхним углом поля состояния окна программы появится надпись «ADuC824 version X.XX», а в поле состояния появится сообщение «Configuration: COM1, 9600 baud RESETTING PART: OK», указывающее на установленное соединение с платой (рис. 2.4).

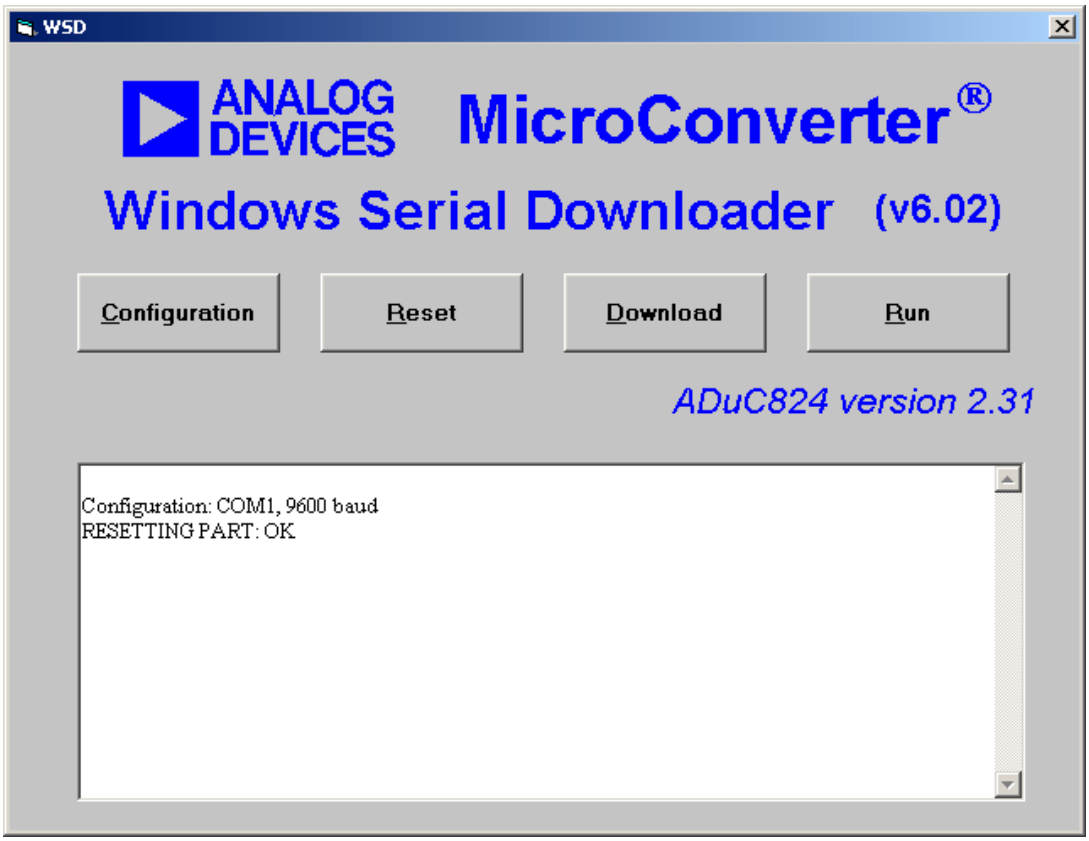


Рис. 2.4. Установление связи ПК с отладочной платой

Для загрузки объектного файла в МК необходимо нажать кнопку «Download» и в появившемся окне дерева файлов и каталогов произвести выбор нужного файла (с расширением .hex). Загрузка начнется при нажатии на кнопку «Open» («Открыть»). В процессе загрузки в поле состояния WSD появится сообщение о стирании памяти микроконвертора, например, «ERASING CODE AND DATA...OK», а затем, как только файл будет загружен, – сообщение об окончании загрузки, например, «DOWNLOADING CODE [C:\PR_ADUC\TIC12\tic12.hex].....OK». Протокол последовательной загрузки микроконвертора позволяет производить загрузку и последующее выполнение целевого кода с различными опциями, которые могут быть выбраны в окне конфигурации, открываемое кнопкой «Configuration» в окне WSD (рис. 2.5). Здесь имеются опции выбора и настройки последовательного порта компьютера, опции выбора типа памяти микроконвертора для последующих стирания и загрузки, опции выбора защиты загруженного кода от считывания и опции выбора способа выполнения целевого кода.

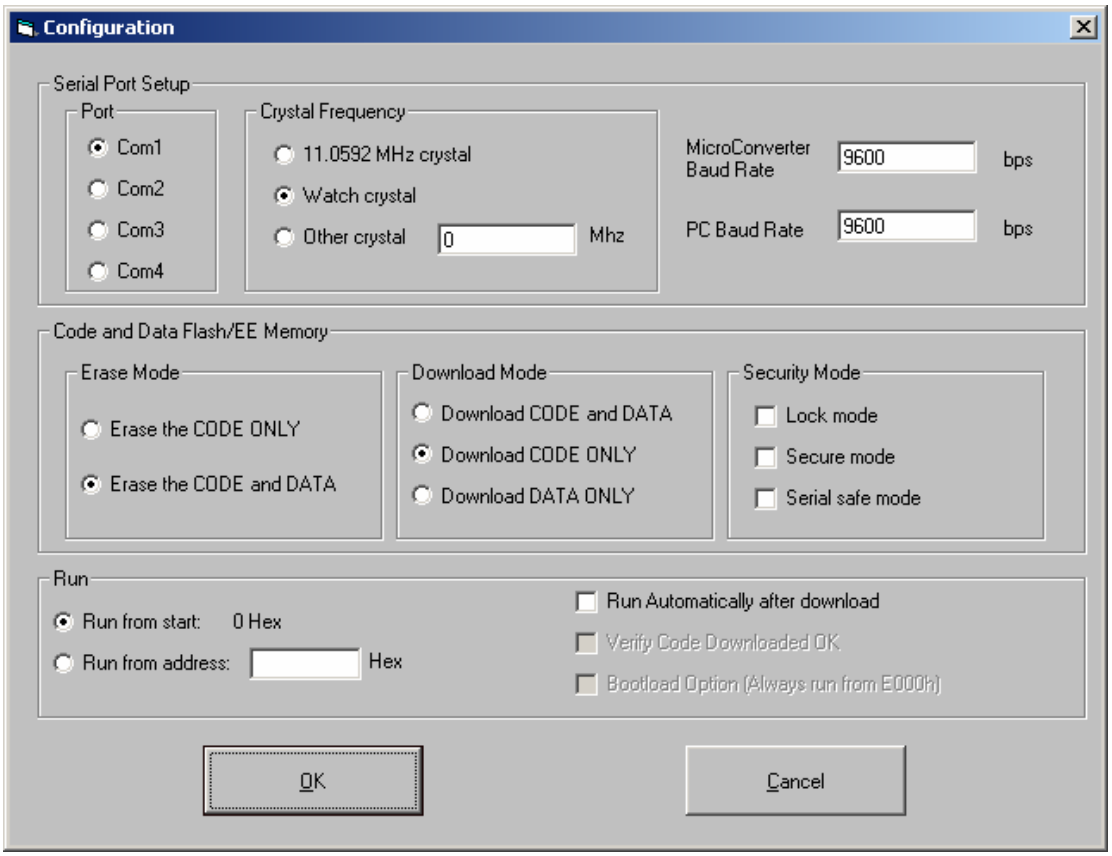


Рис. 2.5. Настройки программы WSD

В группе опций способа выполнения целевого кода «Run» можно задать следующие условия:

– автоматическое выполнение кода после загрузки «Run automatically after download», когда при замкнутой перемычке X1 на плате целевая программа начинает автоматически выполняться сразу после загрузки;

– выполнение кода, инициируемое пользователем после загрузки программы нажатием кнопки «Run» главного окна WSD.

В группе опций «Run» можно также задать адрес отличный от 0000h, с которого начнется выполнение загруженной программы. Для этого следует отменить автоматическое выполнение кода после загрузки, а затем при замкнутой перемычке X1 на плате произвести загрузку программы с указанием в поле опции «Run from address» требуемого адреса программной памяти, например, 57. После нажатия на кнопку «Run» программа начнет выполняться с указанного адреса, а в поле состояния появится сообщение «Run Program from Address 000057h…OK».

Группа опций стирания памяти ADuC824 «Erase MODE» позволяет избирательно производить очистку памяти микроконвертора. Можно очистить всю имеющуюся на кристалле память («Erase the CODE and DATA memory on the target») или только память программ («Erase the CODE memory only»).

Группа опций «Download Mode» позволяет избирательно производить загрузку целевой программы. Можно загружать целевой код только в память

программ («Download CODE only»), только в память данных («Download DATA only») или во всю имеющуюся на кристалле память («Download CODE and DATA»).

Особое внимание перед началом загрузки целевого кода следует уделить опции защиты кода от считывания «Security Mode». Необходимо учитывать, что после установки опции запрещения последовательной загрузки «Serial safe mode» загрузить код в МК удастся только один раз. После этого останется доступным только режим параллельной загрузки.

Поскольку WSD для связи с микроконвертором использует последовательный порт (UART), то следует учитывать, что любые другие программные продукты, поддерживающие связь с ADuC824 по последовательному порту, например, отладчик ADuC или программа WASP, о которых пойдет речь ниже, при одновременном открытии будут конфликтовать с WSD, что приведет к невозможности осуществления загрузки кода. Кроме того, многие программы, эмулирующие режим простого терминала, даже после закрытия их окон продолжают занимать COM-порт компьютера, что не позволит корректно запустить WSD. В этом случае рекомендуется произвести перезагрузку ПК.

2.4. Отладчик DeBugV2

Отладчик – приложение Windows, которое позволяет пользователю отлаживать выполнение своего кода непосредственно в кристалле. При работе отладчика использует последовательный порт микроконвертора UART. Отладчик обеспечивает доступ ко всем периферийным устройствам ADuC824. В ходе отладочной сессии можно производить выполнение целевой программы в пошаговом режиме или по контрольным точкам.

Для корректного запуска отладчика на отладочной плате следует разрешить режим последовательной загрузки, замкнув перемычку X1. Затем следует подать питание на плату, предварительно подключив ее интерфейсным кабелем к COM-порту компьютера. Запуск отладчика производится щелчком левой клавиши мыши по значку «DeBugV2» (запускается файл C:\ADuC\DeBudV2\aduc.exe). После этого откроется окно «Aduc pseudo emulator: new session», а поверх него по умолчанию автоматически откроются окно сессии «Session» и окно мастера сессии «Session Wizard». В последнем появится сообщение, предлагающее создать файл сессии отладки (рис. 2.6).

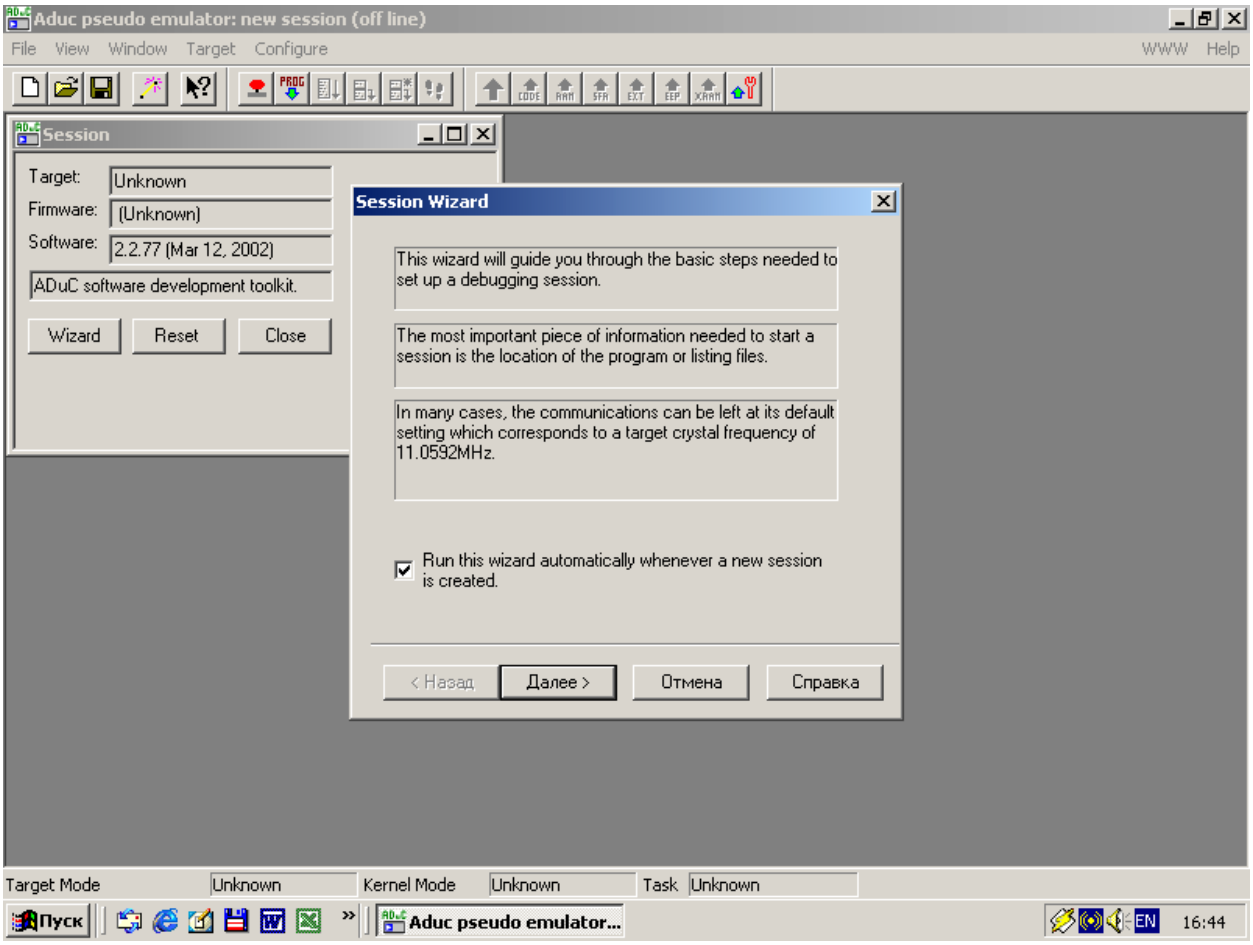


Рис. 2.6. Запуск отладчика

Если мастер сессии автоматически не запускается, то следует нажать на кнопку «Wisard» в окне «Session» или выбрать подопцию «Run Wisard Session» в опции меню «Configure». Затем, нажимая на кнопку «Next» («Далее») в окне мастера сессии, следует выбрать нужный COM-порт компьютера для связи с платой (окно «Comms Port») и файл листинга для отладки (окно «Files»), указав путь к нему или найдя его с помощью кнопки «Browse». Настройка значения тактовой частоты в окне «Baud Rate» принимается по умолчанию равной 11,0592 МГц. В заключительном окне «Session Wizard» нажимается кнопка «Finish» («Готово»). В руководстве по QuickStart Development System [2] рекомендуется для демонстрации работы отладчика выбрать для отладки уже имеющийся на диске в каталоге ADuC файл C:\AduC\DeBugV2\adctrig.lst, однако, при выполнении программы, листинг которой содержится в этом файле, ADuC824 использует внешнюю память данных, которая в рассматриваемом нами упрощенном варианте эволюционной платы отсутствует. В связи с этим, была написана простая программа, на примере которой можно продемонстрировать возможности отладчика. Ее исходный текст находится в файле deb.asm и приведен на рис. 2.7. Принципиальная схема, которую необходимо собрать для работы программы в ADuC824, приведена на рис. 2.8. Программа периодически с большой частотой зажигает и гасит

светодиод, подключенный к линии ввода-вывода порта ADuC824 через токовый ключ, если кнопка, подключенная к линии ввода-вывода другого порта ADuC824, не нажата.

```
;-------------------------------------------------------------------------------
; Программа для демонстрации работы отладчика ADuC824.
; Происходит мигание светодиода с высокой частотой.
; При нажатии на кнопку светодиод зажигается и горит, пока
; удерживается кнопка.
; Светодиод подключен к линии _OUT_LED через токовый ключ.
; Прерывания не используются.
;-------------------------------------------------------------------------------
$INCLUDE (C:\ADuC\mod824)
        $INCLUDE (C:\PR_ADUC\Deb\824.inc)
;-------------------------------------------------------------------------------
;Описание битов, регистров и констант
;-------------------------------------------------------------------------------
;Порты и линии ввода-вывода
PORT_KNOP EQU P0 ;порт кнопок

PORT_IND EQU P2 ;порт индикации

_OUT_LED EQU P2_0 ;выход подключения токового ключа
              ;светодиода

_IN_KNOP EQU P0_0 ;вход кнопки

;Начало исполняемого кода----------------------------------------------------
ORG 0h
AJMP Lab_START ;идти на начало осн программы

;Начало осн программы--------------------------------------------------------
ORG 100h
Lab_START: MOV SP,#080h ;определить указатель стека
MOV PLLCON,#00000000b ;уст макс частоту ядра (12,58 МГц)

MOV PORT_KNOP,#11111111b ;сделать вх все линии порта кн
MOV PORT_IND,#00000000b ;сделать вых все линии порта инд

CLR _OUT_LED ;погасить светодиод

;Начало основного цикла------------------------------------------------------
La_OSN: JB _IN_KNOP,La_1 ;нажата ли кнопка ?
SETB _OUT_LED ;кнопка нажата, зажечь светодиод
LJMP La_OSN ;закрыть основной цикл

La_1: NOP ;
       NOP ;
       NOP ;
       NOP ;
       NOP ;
       NOP ;
       NOP ;
       NOP ;
       SETB _OUT_LED ;зажечь светодиод
       NOP ;
       CLR _OUT_LED ;погасить светодиод
LJMP La_OSN ;закрыть основной цикл

;Конец исполняемого кода
END
```

Рис. 2.7. Программа для демонстрации возможностей отладчика

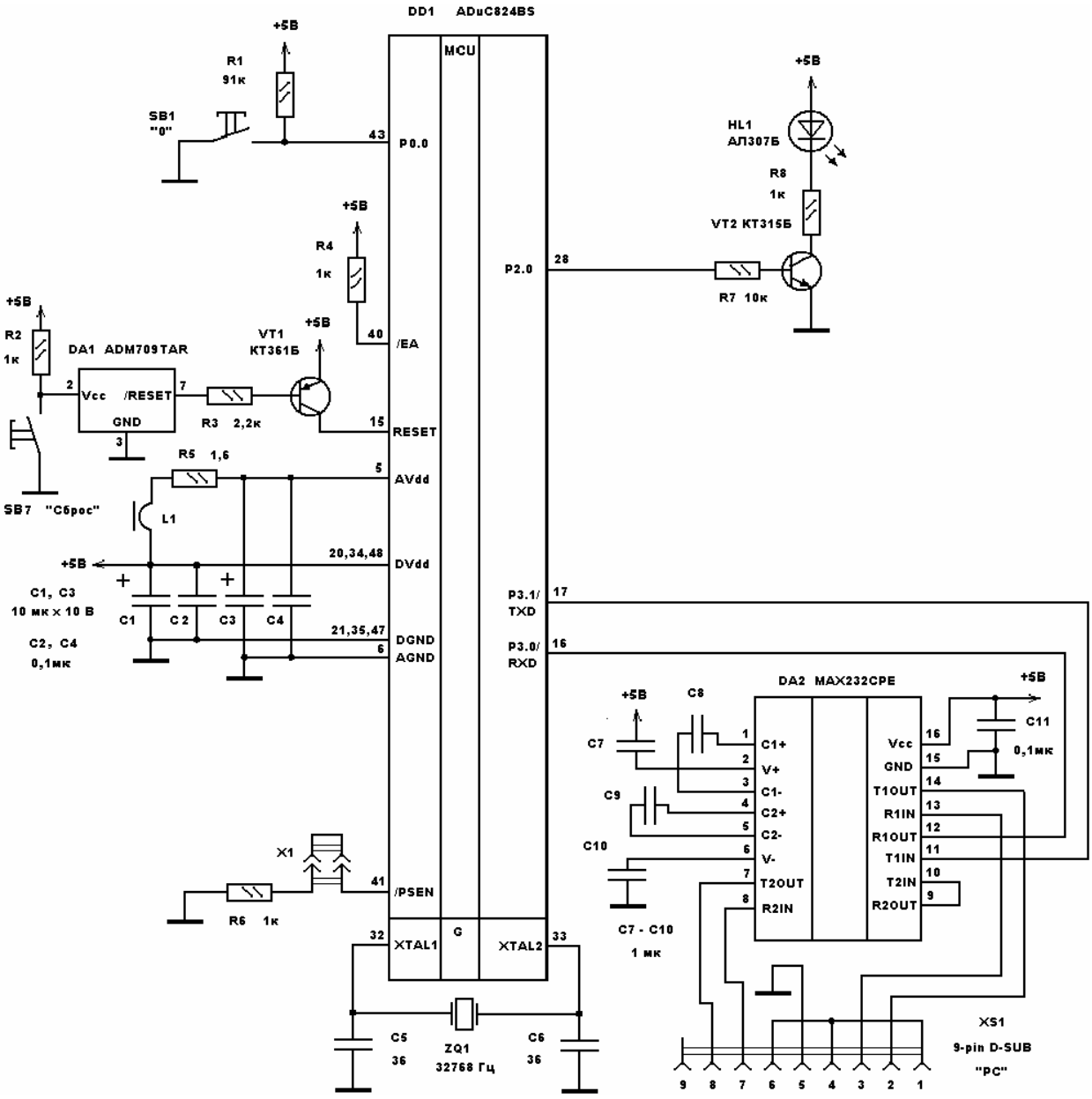


Рис. 2.8. Схема для демонстрации возможностей отладчика

В случае удержания кнопки в нажатом положении, светодиод будет светиться. Из-за высокой частоты миганий внешне кажется, что при ненажатой кнопке светодиод не мигает, а светится с малой яркостью, а при нажатой – с большой.

После выбора в окне «Files» мастера сессии файла deb.lst и закрытия заключительного окна «Session Wizard», как было указано выше, откроется окно листинга программы «DEB.LST». Одновременно с этим внизу главного окна отладчика строка «Target Mode Unknown Kernel Mode Unknown Task Unknown» сменится на строку «Target Mode Kernel Kernel Mode Idle Task idle». Сравните рис. 2.6 и рис. 2.9. Если этого не произойдет, то следует выбрать в опции меню «Target» команду «Reset» или нажать на кнопку «Reset target» на инструментальной панели.

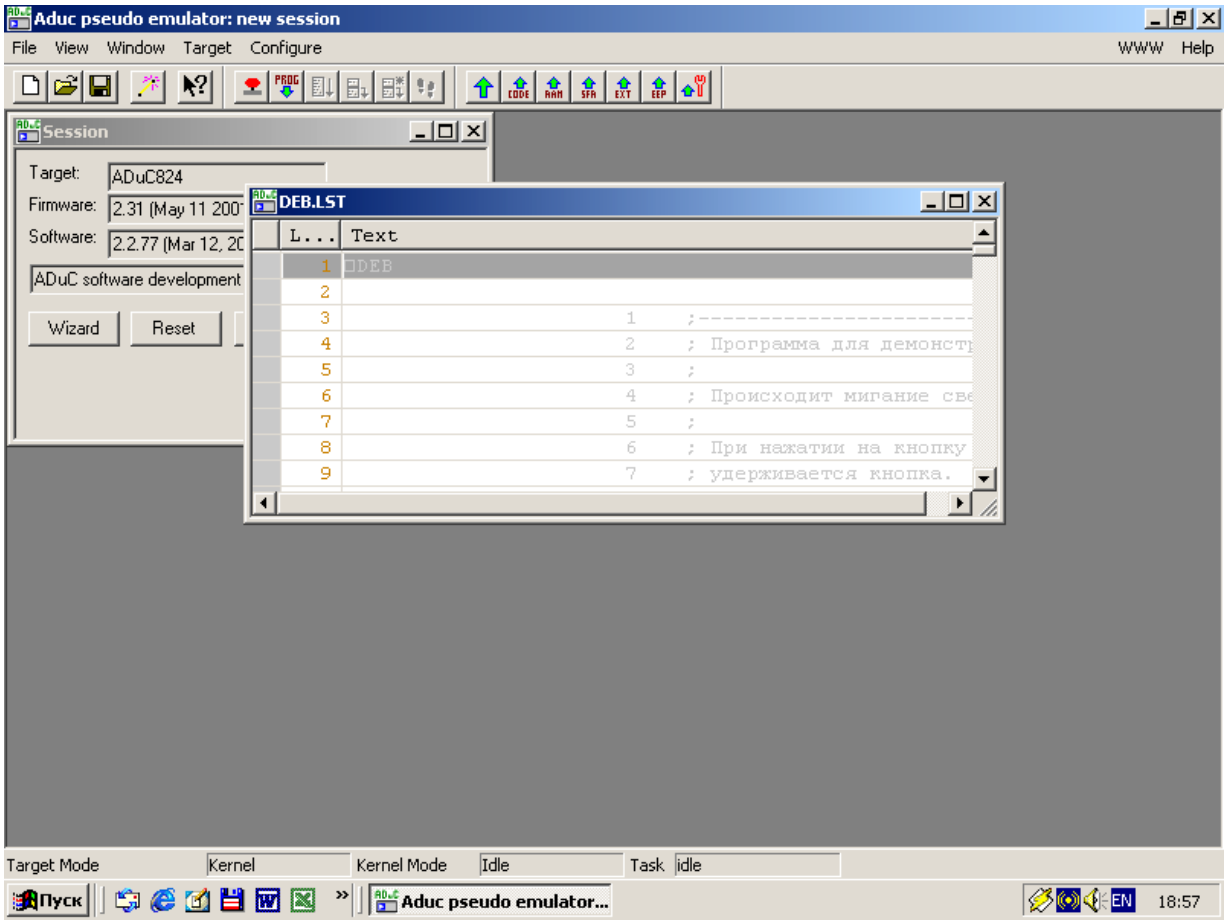


Рис. 2.9. Просмотр листинга отлаживаемой программы

Кроме окна листинга (открывается вручную командой «Program» опции меню «View») в сессии отладки рекомендуется открыть и разместить еще несколько окон, команды открытия которых находятся в подопции «Memory» опции «View».

Окно «Internal RAM» показывает карту распределения внутреннего ОЗУ ADuC824. Если подвести курсор к любому значению адреса или данных в этом и других окнах отображения ресурсов ADuC824 подопции «Memory» и однократно щелкнуть правой клавишей мыши, откроется список, в котором можно выбрать формы представления отображаемого числового значения (десятичную, шестнадцатеричную, двоичную или одновременно несколько форм).

Окно «External RAM» показывает карту распределения внешнего (подключаемого) ОЗУ микроконвертора. Поскольку в нашем случае внешняя память данных (и программ) отсутствует, в сессии отладки мы это окно, как и окно «External Code» открывать не будем.

Окна «SFRs» и «Analog SFRs» отображают состояние регистров специальных функций соответственно цифровых и аналоговых модулей ADuC824.

Окна «Flash EE/Data» и «Flash EE/Code» отображают состояние Flash-памяти ADuC824 соответственно данных и программ.

Окно «Registers» отображает содержимое регистров обслуживания процессорного ядра 8051.

Окно «BITs» отображает содержимое побитно адресуемой области внутреннего ОЗУ микроконвертора.

Для загрузки кода программы в кристалл следует выбрать в опции меню «Target» команду «Download Program» или нажать на кнопку «Download» на инструментальной панели. Процесс загрузки графически отобразится в открывшемся окне (рис. 2.10). Обратите внимание, что коды команд программы после загрузки появятся в окне содержимого программной памяти «Flash/EE Code».

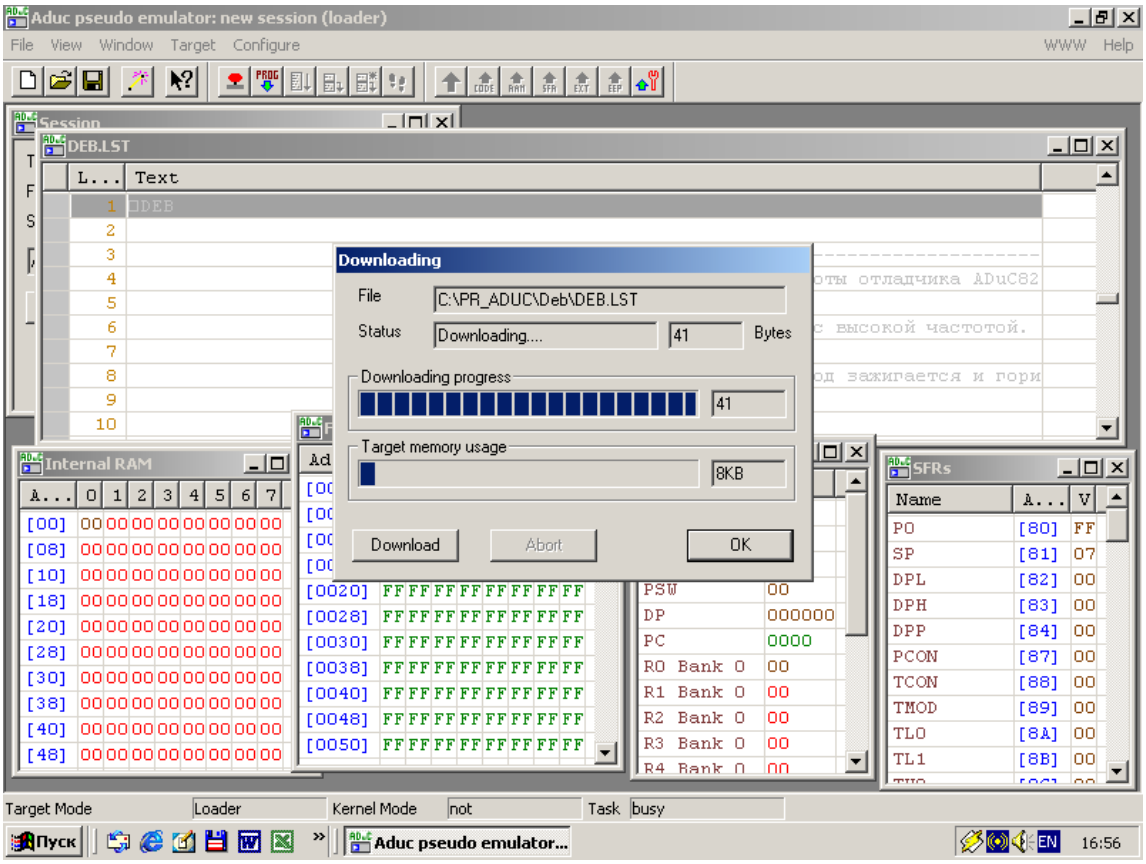


Рис. 2.10. Информация о процессе загрузки программы

Для выполнения в «железе» загруженной в кристалл программы следует выбрать в опции меню «Target» команду «Run downloaded Program» или нажать на кнопку «Run from the Start» на инструментальной панели. Перед запуском программы на выполнение дважды щелкните правой кнопкой мыши в окне листинга на строке «272 0113 02010E LJMP La_OSN», установив по этому адресу программной памяти контрольную точку (точку останова). При этом обратите внимание, что на левом краю строки появится указание на наличие контрольной точки в виде круглой красной метки (рис. 2.11).

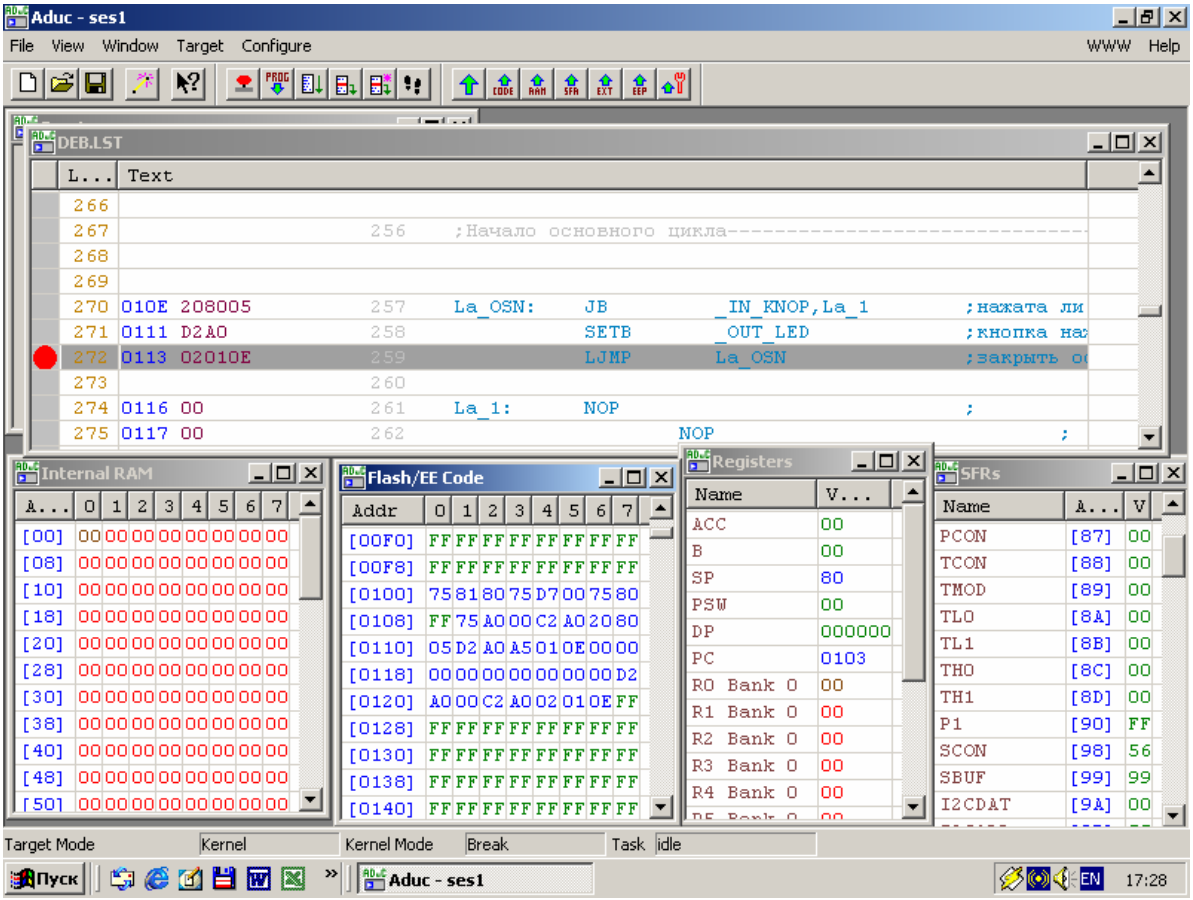


Рис. 2.11. Окно с красной меткой остановки

После запуска программы на выполнение откроется диалоговое окно «Waiting» с надписью «Waiting for RESET or BREAK notification from target». При этом программа в кристалле начнет выполняться в реальном времени, о чем можно судить по свечению светодиода с пониженной яркостью (миганию с высокой частотой). Поскольку подключенная к ADuC824 кнопка пока не нажата, управление в программе не попадает в ветвь, где находится установленная нами контрольная точка, и процесс выполнения ничем не прерывается. Однако, стоит нажать на кнопку, и управление пойдет по этой ветви и дойдет до контрольной точки. На этом адресе выполнение будет остановлено и откроется окно «Target response!» с указанием адреса и причины остановки. В данном случае – «Target broken at adress 000113» и «Breakpoint at line 272 in DEB.LST» (рис. 2.12).

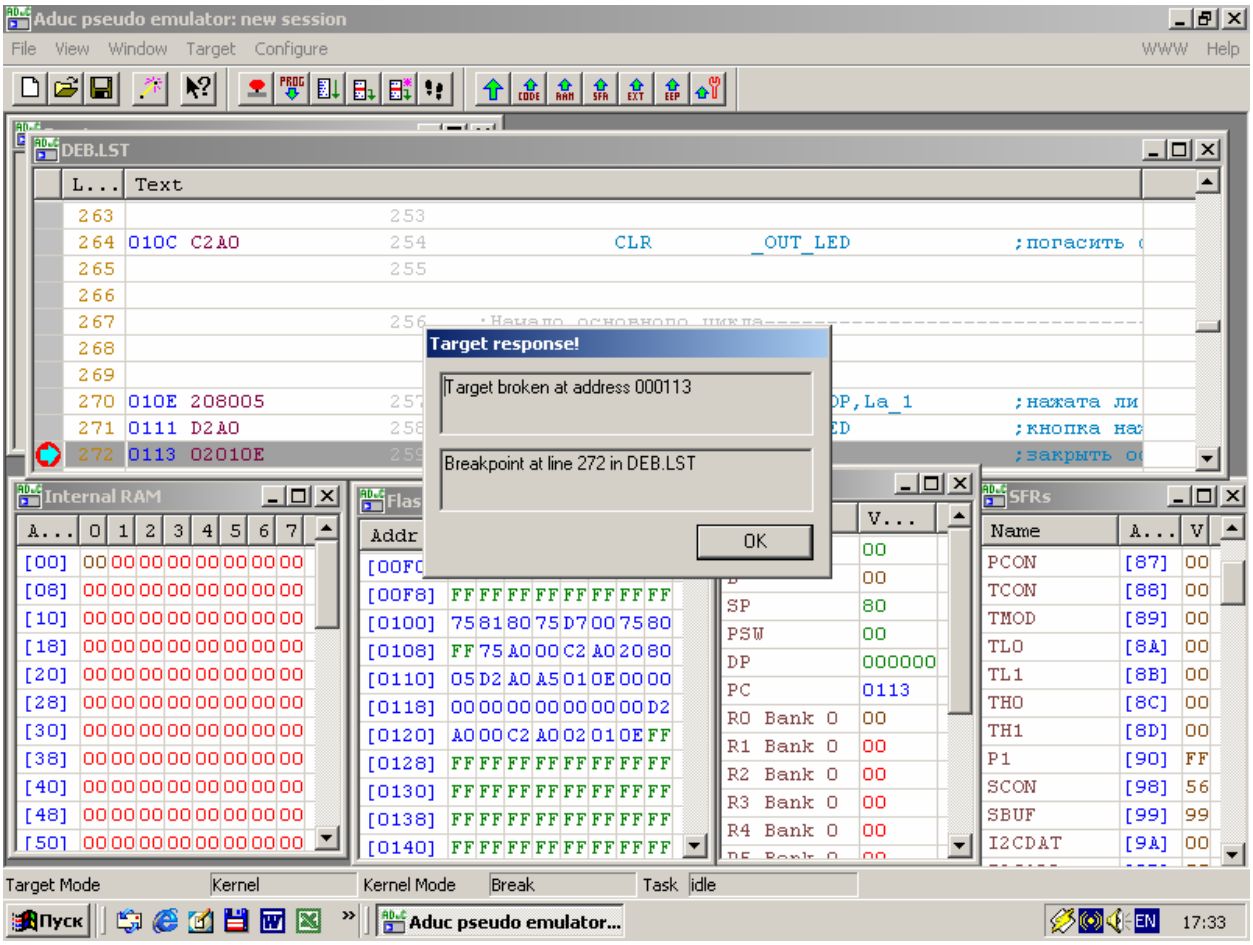


Рис. 2.12. Сообщение об остановке выполнения программы

После остановки светодиод будет светиться постоянно (с большой яркостью), так как предшествовавшая контрольной точке и уже выполненная инструкция SETB _OUT_LED установила линию подключения светодиода в высокий уровень. Для снятия контрольной точки следует дважды щелкнуть правой кнопкой мыши в окне листинга на строке, где она была установлена. Круглая красная метка при этом исчезнет. Одновременно снять все ранее установленные контрольные точки можно с помощью команды «Clear All Breakpoints» из списка, открываемого щелчком правой клавиши мыши в окне «Breakpoints» опции меню «View».

Для возобновления выполнения программы после остановки в контрольной точке служит команда «Resume» из опции меню «Target». При выборе этой команды откроется окно «About to restart», в котором запрашивается подтверждение на возобновление выполнения программы с адреса, где произошла остановка. В случае, если требуется возобновить выполнение с какого-то другого адреса программной памяти, можно воспользоваться командой «Resume special» из опции меню «Target». При выборе этой команды откроется одноименное окно, в поле метки «Addr» которого следует установить требуемый адрес. Кроме того, в этом окне можно

модифицировать содержимое нескольких специальных регистров вычислительного ядра микроконвертора.

Помимо выполнения целевого кода в режиме реального времени отладчик позволяет производить выполнение в пошаговом режиме. Для этого в опции меню «Target» имеется команда «Single Step», которая дублируется кнопкой на контрольной панели. При выполнении каждой инструкции программы (при подаче этой команды) открывается окно «Uploading..» с отображением процесса выгрузки данных из МК в компьютер.

При пошаговой отладке следует обратить внимание на следующий важный момент. В ходе выполнения программы цвет числовых значений данных в любом открытом окне ресурсов ADuC824 из подопции «Memory» различен. По умолчанию синим и зеленым цветом отображаются значения данных, которые в ходе отладочной сессии хотя бы один раз копировались из микроконвертора в компьютер по последовательному порту и, следовательно, их значения (в соответствующих окнах) совпадают с их реальными значениями в «железе». Синий цвет указывает на то, что значение данных (содержимое регистра или бита) было изменено программой в ходе выполнения последней инструкции. Зеленый цвет указывает на то, что значение в ходе выполнения последней инструкции не менялось. Красный цвет указывает на то, что их значения в ходе сессии отладки еще ни разу не копировались из МК в компьютер, и, следовательно, могут отличаться от их реальных значений в МК. При отладке в режиме реального времени смена цветов данных в окнах, т. е. их выгрузка в компьютер происходит только после останова на контрольной точке и нажатия на кнопку «OK» в окне «Target response!». Для того чтобы вручную привести отображаемые по умолчанию в окнах подопции «Memory» значения данных в соответствие с их реальными значениями, отладчик предоставляет в распоряжение пользователя следующий способ. Если подвести курсор к любому значению данных в окне и дважды щелкнуть мышкой, то откроется окно «Read/Write Memory Location», в поле которого «Current Value» можно по нажатию кнопки «Read» произвести чтение этого значения из МК (рис. 2.13). При этом цвет значения данных в окне, где его первоначально «пометили», сменится на зеленый, если до этого он был иным, а само значение изменится, став равным значению в кристалле.

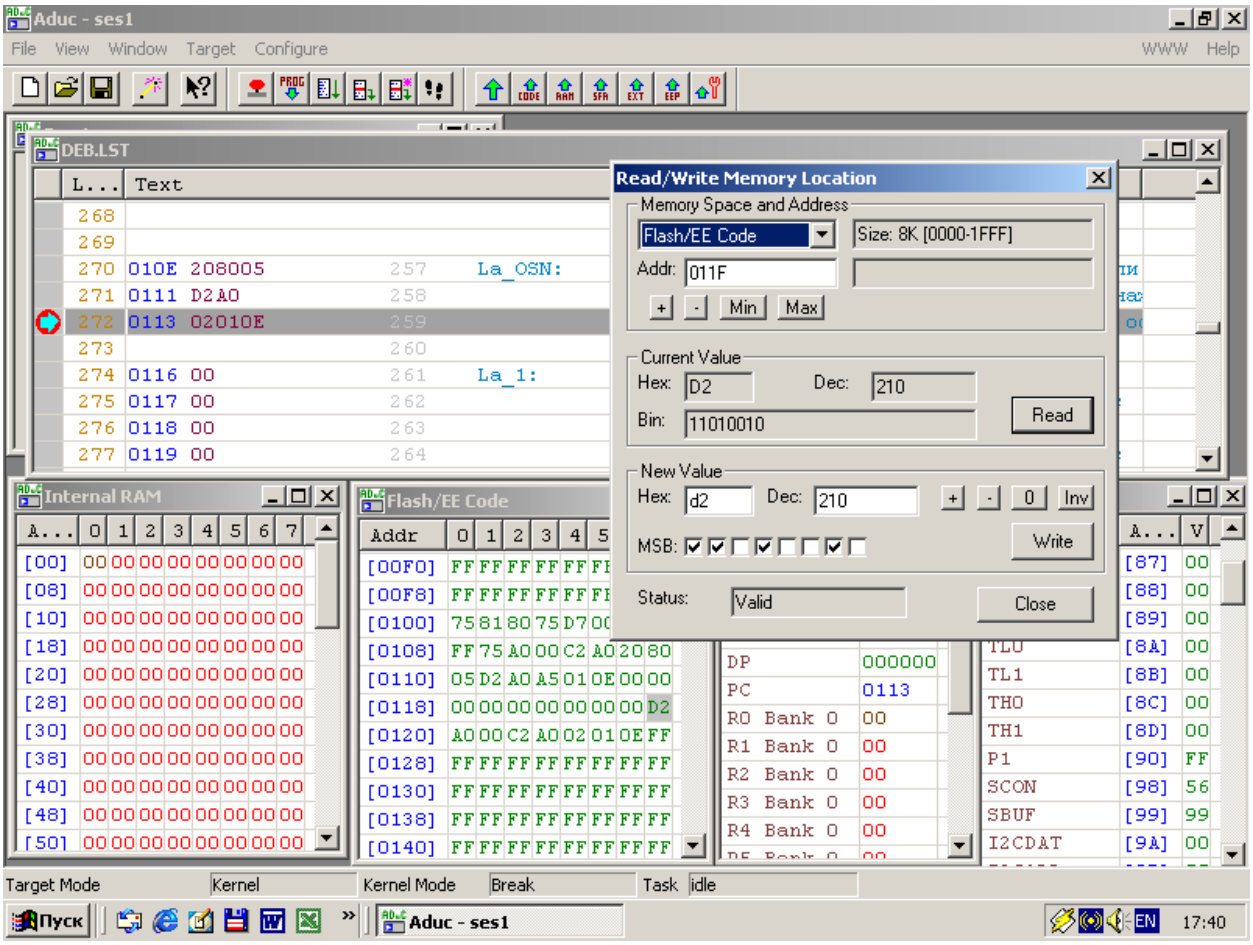


Рис. 2.13. Чтение данных из МК

Кроме побайтного избирательного чтения данных по произвольному адресу имеется возможность читать из кристалла содержимое сразу всего блока данных каждого типа памяти ADuC824. Чтение запускается группой команд из подопции «Upload» опции меню «Target». Во время чтения открывается окно «Uploading», в поле «Status» которого указывается, из какой области памяти ADuC824 производится чтение, а также графически отображается процесс выгрузки данных из МК в компьютер. Таким способом можно выгрузить данные из внутреннего ОЗУ (команда «Internal Ram»), внешнего ОЗУ («External Ram»), регистров специальных функций («SFRs»), программной памяти («Flash/EE Code»), энергонезависимой памяти данных («Flash/EE Data»).

Помимо чтения данных из кристалла отладчик предоставляет пользователю еще одну очень важную возможность – модифицировать значения данных непосредственно в кристалле. В окне «Read/Write Memory Location», о котором шла речь выше, имеется поле «New Value», где можно набрать новое значение данных и, нажав на кнопку «Write», передать это значение в ADuC824 (рис. 2.13). При этом следует иметь в виду, что если указанным способом модифицировать какие-либо данные в программной памяти ADuC824, то это будет осуществлено в «железе» и сразу отобразится в

окне «Flash EE/Code», но содержимое окна листинга при такой модификации никак не изменится. При последующем запуске на выполнение инструкции в программе будут выполняться не в соответствии с их мнемоническими изображениями в окне листинга, а в соответствии с новым, измененным и записанным в кристалл содержимым программной памяти, что поначалу может вызвать недоумение пользователя.

Сказанное можно пояснить на следующем примере. Модифицируем одну из инструкций в программной памяти МК таким образом, чтобы при выполнении программы это сразу стало заметно. Пусть это будет инструкция, расположенная в памяти по адресу 011F и в окне листинга имеющая мнемонику SETB _OUT_LED. Она зажигает светодиод. Как можно видеть из листинга, инструкция двухбайтовая, ее код – D2A0. Изменим первый байт (код операции) на значение C2, чтобы получилась инструкция с кодом C2A0, которая в соответствии с таблицей команд контроллеров семейства 8051 имеет мнемонику CLR _OUT_LED, т. е. наоборот, гасит светодиод. Таким образом, при выполнении получившейся программы в случае, если кнопка не нажата, светодиод не должен светится вообще, так как в программной ветви, по которой пойдет управление не будет инструкций, которые его зажигают. А инструкций, гасящих светодиод, будет теперь две. Для осуществления модификации наведем курсор на число D2 по адресу 011F в окне «Flash EE/Code» и изменим его на C2 описанным выше способом. После запуска программы на выполнение в пошаговом режиме можно легко убедиться в том, что в результате выполнения инструкции «283 011F D2A0 SETB _OUT_LED» светодиод теперь не загорается или гаснет, если он был включен до этого.

Теперь удостоверимся в том, что произведенные изменения программы действительно сохранились в «железе». Снимем все контрольные точки (это важно!), закроем отладчик, отказавшись от предложения сохранить файл сессии, отключим питание эволюционной платы, разомкнем перемычку X1 и включим питание платы снова. Программа начнет выполняться с учетом внесенных изменений, т. е. физическая модификация программной памяти действительно имела место.

Любопытно, что если в сеансе отладки оставить в тексте листинга программы контрольные точки, то они будут работать и в «железе», останавливая выполнение программы при попадании на них управления. В этом легко убедиться, установив контрольную точку, например, в строке «272 0113 02010E LJMP La_OSN» и проделав все изложенное в предыдущем абзаце. Теперь при нажатии на кнопку светодиод будет загораться, затем управление попадет на адрес контрольной точки, и светодиод останется гореть до проведения аппаратного сброса ADuC824.

Этот факт можно объяснить следующим образом. При установке контрольной точки по некоторому адресу содержимое байта программной памяти по этому адресу заменяется на значение A5, что сразу же отображается в окне «Flash EE/Code» (рис. 2.11). Код команды по адресу 0113 LJMP La_OSN

после установки на ней контрольной точки изменится с 02010E на A5010E. Коду операции A5 в списке команд контроллеров семейства 8051 не соответствует никакая команда, поэтому при передаче управления на этот адрес и произойдет остановка выполнения.

Опция меню «Configure» отладчика включает в себя подопцию настройки параметров сессии отладки «Session Settings», команду ручного запуска мастера сессии «Run Session Wizard», подопцию настройки параметров самого отладчика «System settings» и команду сброса (установки в состояние по умолчанию) параметров отладчика «Restore System Defaults».

Подопция «Session Settings» включает в себя страницу выбора COM-порта компьютера для связи с ADuC824 («Comms Port»), страницу выбора скорости обмена между компьютером и ADuC824 («Baud Rate»), страницу выбора файла листинга для отладки («Files»), страницу выбора размера внешнего ОЗУ при его наличии («External Memory») и страницу установки параметров выгрузки данных из ADuC824 при отладке («Upload»). Последняя страница, очевидно, представляет для пользователя наибольший интерес. В ней можно задать, из каких типов памяти ADuC824 данные будут выгружаться в компьютер автоматически после выполнения очередной инструкции в пошаговом режиме или после остановки в контрольной точке в режиме реального времени. Внешний вид окна, открытого на этой странице, показан на рис. 2.14. Для запрещения выгрузки данных из какого-либо типа памяти ADuC824 следует в соответствующей этому типу строке окна «Upload» сделать пометку в графе «No».

Для разрешения выгрузки всей памяти данного типа делается пометка в графе «All». Чтобы разрешить выгрузку какой-либо части данного типа памяти делается пометка и указываются границы области выгрузки в графе «Section (in hex)». Пометив для всех имеющихся типов памяти кристалла графу «All», пользователь будет в максимальной степени информирован о текущем состоянии ресурсов микроконвертора в каждый момент дискретного отладочного времени, однако выгрузка данных из-за их большого объема станет занимать значительное реальное время. Кроме того, пользователю нет необходимости получать информацию из тех областей и типов памяти, про которые точно известно, что их содержимое при отладке данной программы не изменится. Исходя из этих соображений, следует настроить окно «Upload» таким образом, чтобы достичь компромисса между временем выгрузки и требуемыми объемом и типами выгружаемых данных. Можно запретить появления на время выгрузки окна «Uploading..», убрав метку из поля «Show Progress Bar when uploading».

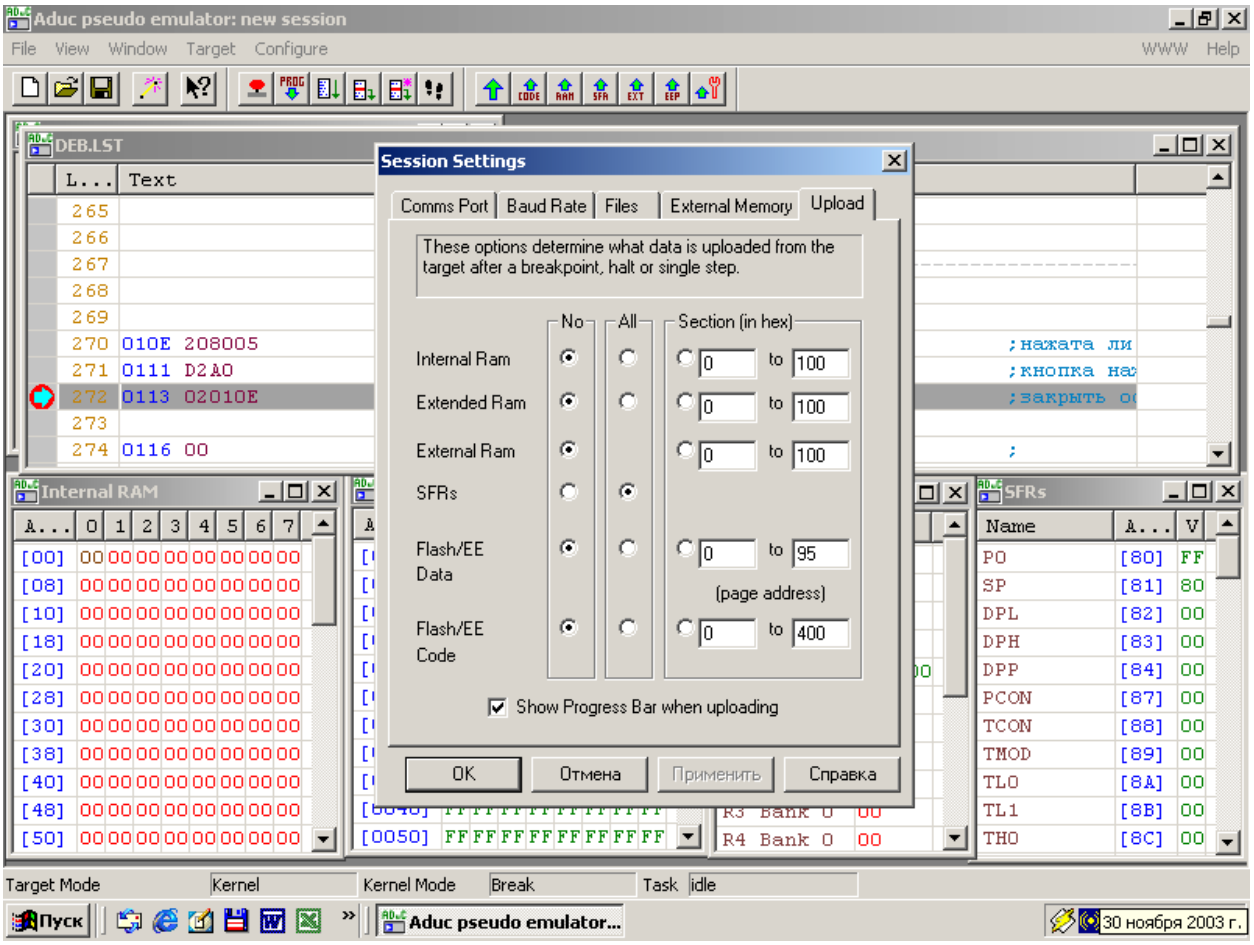


Рис. 2.14. Работа с различными типами памяти кристалла

На страницах подопции «System settings» настраиваются некоторые параметры отладчика и процесса отладки.

На странице «Run» в поле метки «Program Start Address» можно установить адрес, с которого начнется выполнение программы при отладке, а также будет ли во время выполнения открываться диалоговое окно «Waiting». По умолчанию программа начинает выполняться с адреса 0x0000.

На странице «Reset» задается поведение отладчика при подаче команды «Reset». Здесь, например, можно установить, чтобы сброс производился автоматически после загрузки файла сессии («Reset the target after loading the session file(s)»). В поле «Erase mode» можно установить, будет ли во время сброса стираться память программ и данных или только память программ.

На странице «Colors» можно выбрать цвета отображения адресов, данных, меток и контрольных точек взамен их цветов, заданных по умолчанию. Если вы желаете возвратиться к цветам, установленным по умолчанию, используйте команду «Defaults».

На странице «Refresh» можно изменять скорость обновления изображения на экране в сеансе отладки. Меньшая скорость обновления высвобождает больше времени для обмена с микроконвертором и уменьшает мерцание изображения на экране монитора.

На странице «Memory Windows» можно выбирать форму представления значений чисел в окнах, открывающихся из подопции «Memory» опции «View» (шестнадцатеричная, десятичная и двоичная форма).

На странице «Program Windows» можно задать некоторые особенности отображения окон отладчика, а также разрешить или запретить перемещение курсора при нажатой правой клавише мыши.

На странице «Step» можно задавать детали работы отладчика при выполнении программ в пошаговом режиме. Сюда входит разрешение или запрещение открытия диалоговых окон во время и после выполнения каждой инструкции, а также разрешение или запрещение передачи данных из ADuC824 в компьютер после каждого шага. Последняя установка задается в поле метки «Upload data from the target» и по умолчанию рекомендуется оставить ее включенной.

На странице «Startup» задаются особенности работы отладчика при его запуске на выполнение. Здесь, например, можно разрешить или запретить автоматический запуск мастера сессии при открытии новой сессии («Run the session wizard whenever a new session is created»).

На странице «Downloader» можно задать детали работы встроенного в отладчик блока загрузки программного кода. Можно разрешить или запретить автоматический старт загрузки кода в ADuC824 сразу после открытия диалогового окна «Download», а также автоматическое закрытие этого окна после окончания загрузки.

Страница «Hardware Breakpoints» содержит опции, включение которых позволяет оптимизировать и ускорить процесс установки контрольных точек. Поскольку информация о наличии и местоположении контрольных точек физически записывается в программную память кристалла, то здесь они называются аппаратными контрольными точками. Ускорение и оптимизация достигаются путем использования особым образом организованного алгоритма записи, позволяющего не перезаписывать всю память из-за нескольких модифицируемых байтов.

Страница «Resume» содержит опции, управляющие поведением отладчика при возобновлении процесса выполнения программы после остановки в контрольной точке. Пользователь может разрешить или запретить открытие диалоговых окон, сообщающих об остановке и возобновлении выполнения, а также установить, будет ли при возобновлении выполнения открываться диалоговое окно «Waiting».

Страница «Break» содержит опции, управляющие поведением отладчика при достижении контрольной точки. Пользователь может разрешить или запретить открытие диалоговых окон, сообщающих об остановке выполнения в результате достижения контрольной точки, а также разрешить или запретить передачу данных из ADuC824 в компьютер после остановки на контрольной точке. Последняя установка задается в поле метки «Upload selected data ranges from the target» и по умолчанию рекомендуется оставить ее включенной.

Для сохранения установленных нами параметров настройки сессии следует выбрать в опции меню «File» команду «Save» или нажать на соответствующую кнопку на инструментальной панели отладчика. Файл с настройками параметров сессии будет по умолчанию сохранен под именем ses1.ses в одном каталоге с загруженным файлом листинга. При выборе команды «Save As…» можно вручную выбрать имя файла сессии для сохранения и путь к нему в открывшемся окне дерева файлов и каталогов, как это показано на рис. 2.15.

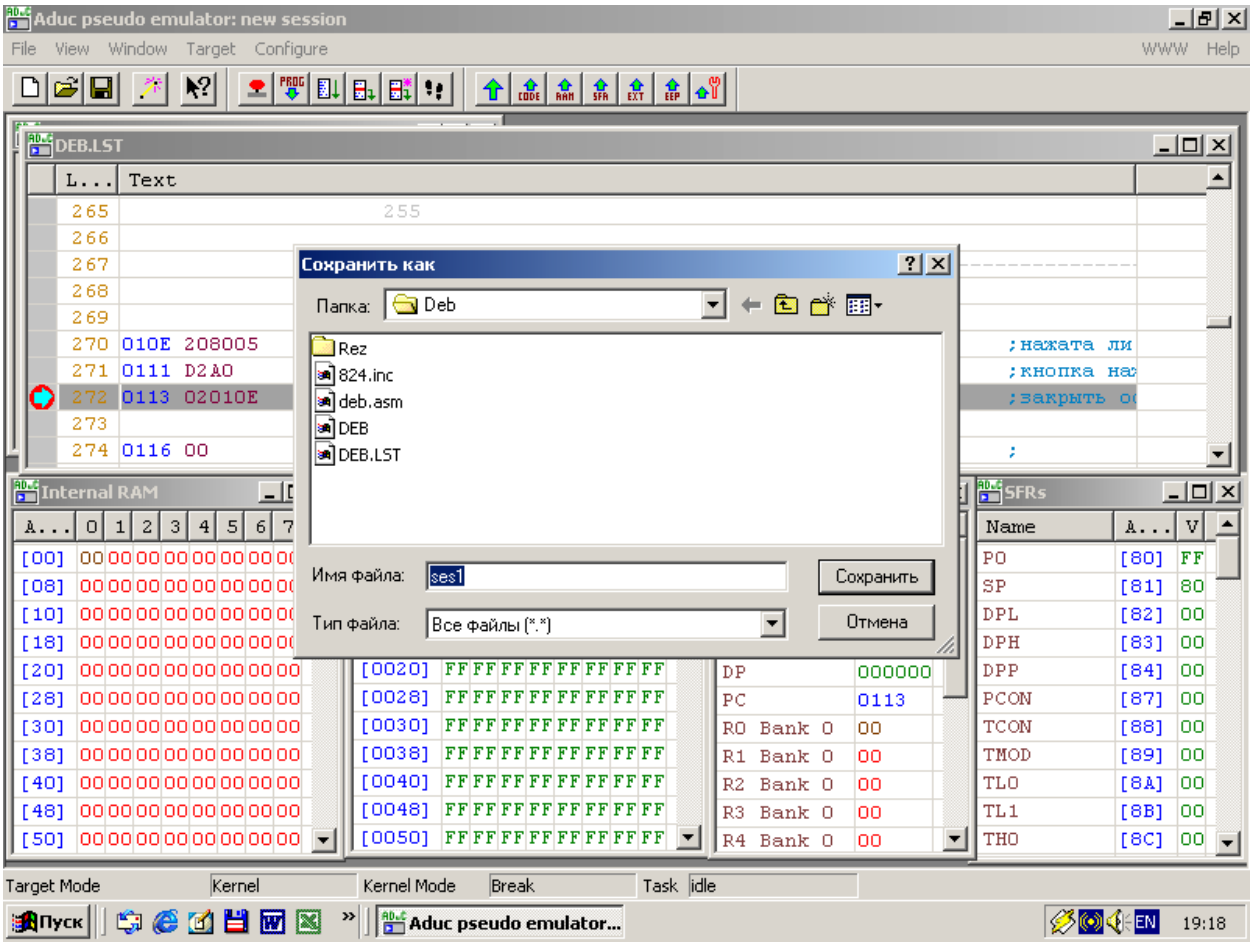


Рис. 2.15. Сохранение файла сессии

Чтобы открыть существовавший ранее файл сессии, следует при автоматическом запуске мастера сессии «Session Wizard» отказаться от создания нового файла сессии, затем в опции меню «File» выбрать команду «Open..» или нажать на соответствующую кнопку на инструментальной панели отладчика. Выбранный в открывшемся окне дерева файлов и каталогов файл с расширением .ses будет загружен в отладчик, но для запуска программы на выполнение придется произвести загрузку кода в кристалл, выбрав в опции меню «Target» команду «Download Program».

2.5. Симулятор ADSIM

ADSIM – это приложение Windows, которое полностью программно моделирует на ПК все функциональные возможности микроконвертора, включая АЦП и ЦАП. Симулятор имеет удобный интуитивный интерфейс и располагает несколькими стандартными способами отладки целевых программ, включая возможность создания множества контрольных точек и пошагового выполнения. Симулятор как инструментальное средство разработчика удобно использовать для обучения программированию ADuC824, а также для предварительной отладки целевого кода до его переноса в кристалл. Симулятор идеально подходит для отладки программных блоков целевой программы, не требующих взаимодействия с внешней средой микроконвертора, например, подпрограмм двоичной арифметики, преобразования форматов представления чисел, обращения к табличным функциям и т. п. ADSIM точно моделирует АЦП, включая возможность выбора времени преобразования, автокалибровку и симуляцию температурного датчика на кристалле. Применение симулятора при разработке позволяет сберечь ресурс количества циклов записи-стирания программной памяти используемого микроконтроллера, хотя для ADuC824 последнее не является актуальным.

Открытие симулятора производится двойным щелчком левой клавиши мыши на иконке ADSim в меню START. Затем необходимо произвести выбор устройства (микроконвертора) для симуляции. В опции меню «Configuration» следует выбрать пункт «Project Options» и в поле «Processor Options» открывшегося окна выбрать из выпадающего списка ADuC824, как это показано на рис. 2.16. Тактовая частота ADuC824 не может быть выбрана пользователем, так как симулятор сам устанавливает ее равной 32768 Гц. В поле «Warnings» окна «Project Options» пользователь может определить, при возникновении каких ситуаций в процессе симуляции целевого кода симулятор будет генерировать предупреждения.

В разных версиях симулятора поля могут иметь различный вид. После выбора требуемого устройства следует удостовериться, что верхняя строка окна симулятора начинается с надписи «ADuC824».

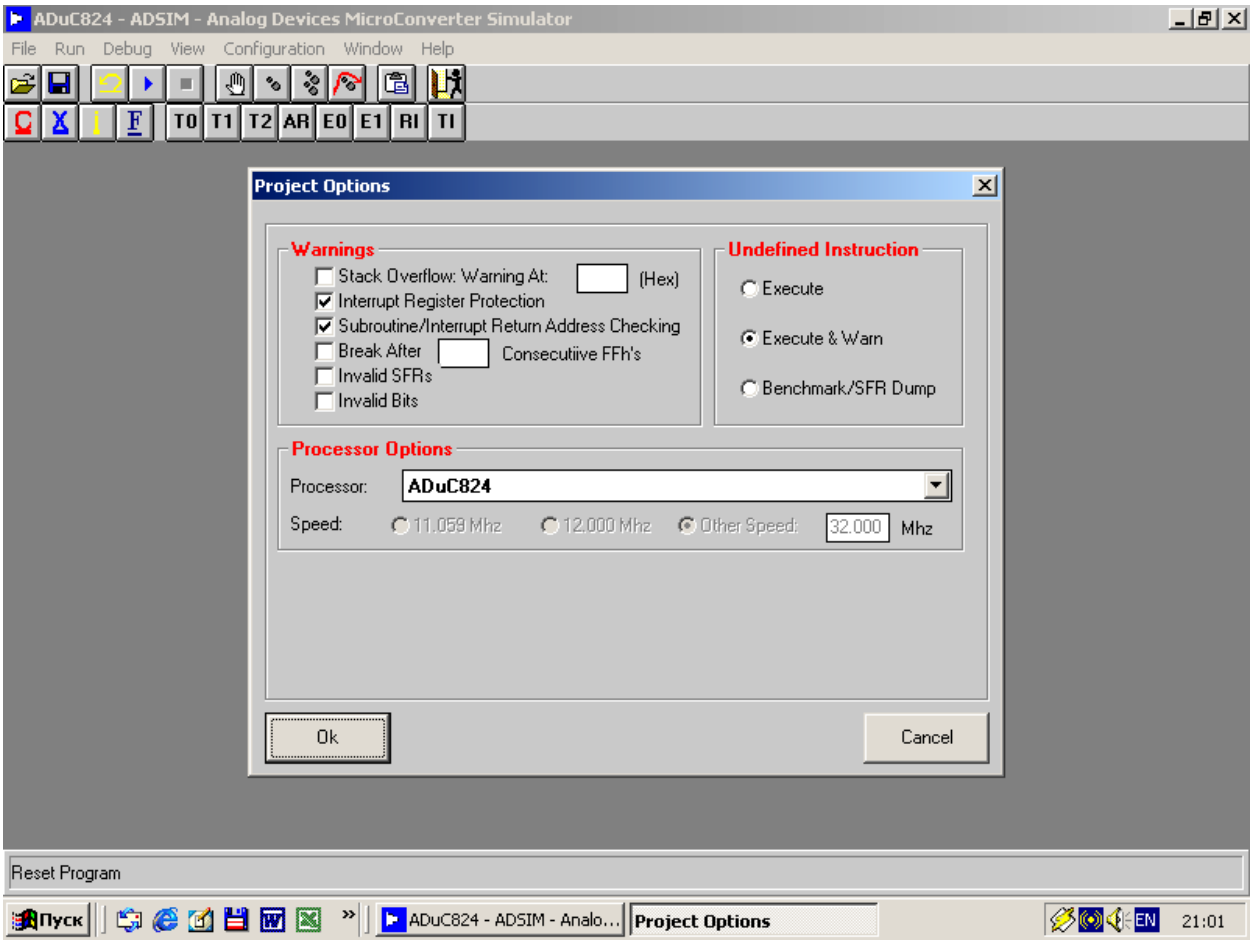


Рис. 2.16. Настройка симулятора

Для подготовки к старту симуляции необходимо в опции меню «View» выбрать опцию «Program Disassembly». В открывшемся окне можно наблюдать таблицу, отображающую программную память симулируемого микроконвертора с индикацией в каждой строке адреса памяти, шестнадцатеричного содержимого ячейки с указанным адресом и мнемонического обозначения ассемблерной инструкции, шестнадцатеричный код которой соответствует этому содержимому (рис. 2.17).

Так как никакая целевая программа пока не загружена в симулятор, во всех ячейках содержится код, соответствующий «стертому» состоянию ячейки, – 0FFh. Этому коду соответствует ассемблерная мнемоника MOV R7,A. Помимо окна «Program Disassembly» в распоряжении пользователя имеется еще несколько окон симуляции, которые открываются из опции меню «View» и из подопций «Memory View» и «SFR View». Из подопции «Memory View» можно открыть окна, отображающие содержимое внешней и внутренней регистровой памяти микроконвертора (XRAM и IRAM), а также Flash/EE-памяти данных. При открытии любого из этих окон карта соответствующего типа памяти устройства отображается в виде таблицы. При однократном щелчке левой клавиши мыши на любом элементе таблицы открывается окно «Change Byte Value», из которого пользователь может модифицировать значение байта данных,

содержащегося в этом элементе (ячейке) памяти. В окне «Flash/EE Data Memory» дополнительно отображаются регистры специальных функций, обслуживающие массив Flach/EE-памяти данных ADuC824. Они также доступны для модификации пользователем.

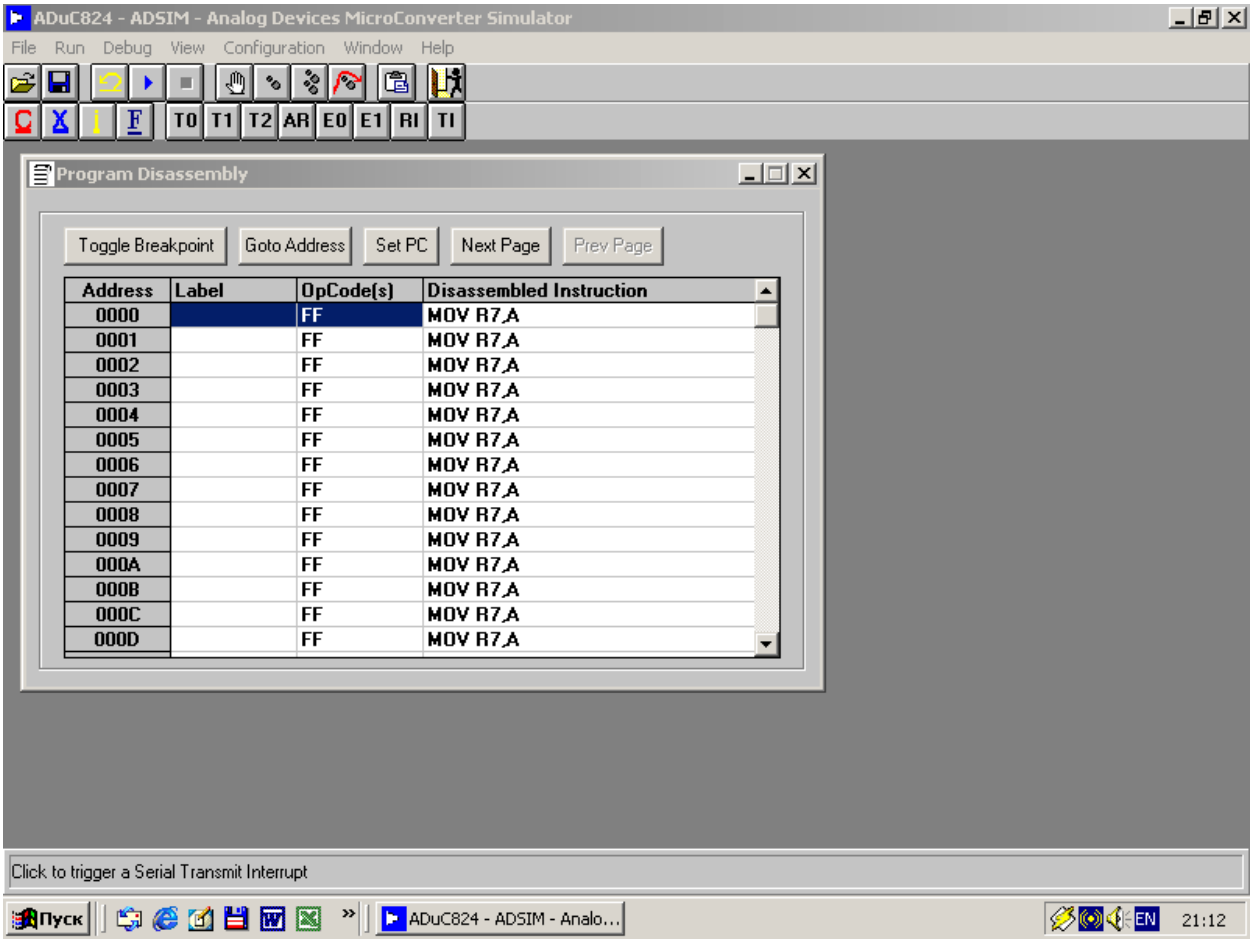


Рис. 2.17. Просмотр памяти МК

Из подопции «SFR View» можно открыть окна, позволяющие контролировать регистры специальных функций микроконвертора, обслуживающие различные аппаратные модули ADuC824. Окно «8051 Core» дает возможность наблюдать и модифицировать регистры специальных функций процессорного ядра, совместимого с семейством микроконтроллеров 8051. Окно «Analog SFRs» отображает группу специальных регистров, обслуживающих модули основного и дополнительного АЦП, включая регистры калибровки, а также текущую конфигурацию обоих модулей АЦП и температуру, соответствующую показаниям модуля дополнительного АЦП в случае, если последний подключен к температурному датчику. Окно «On Chip Monitors» отображает специальные регистры контроля сторожевого таймера и монитора источника питания микроконвертора. Окно «I2C/SPI I/O SFRs» отображает группу специальных регистров контроля интерфейсов I2C/SPI, индицирует текущий режим интерфейса I2C микроконвертора, а также имеет поле отображения данных, принимаемых через линии интерфейса I2C/SPI от

внешних устройств. Окно «Time Interval Counter SFRs» является окном контроля аппаратного модуля счетчика временного интервала (TIC) ADuC824. В окне «PLL SFR» можно путем модификации соответствующего регистра специальных функций выбирать тактовую частоту вычислительного ядра ADuC824. Окно «Timer SFRs» позволяет контролировать все таймеры ADuC824 путем прямой модификации их содержимого и модификации специальных регистров, которые обслуживают таймеры.

Окно «Uart Terminal Window» отображает текущее состояние последовательного порта UART эмулируемого микроконвертора (инициализирован порт целевой программой или нет), индицирует скорость передачи данных, на которую в данный момент настроен порт, а также имеет поле отображения данных, принимаемых через порт от внешних устройств. В случае, если UART был инициализирован программой корректно, при однократном щелчке правой клавишей мыши на кнопке «ASCII» открывается окно «Enter the HEX value to send», в поле метки «Value(HEX)» которого можно установить в шестнадцатеричном виде значение байта данных. Этот байт при нажатии на кнопку «OK» будет передан через UART во внешнее устройство.

Из других подопций опции «View» наибольший интерес для пользователя представляют окна «I/O Ports» и «Program Analysis». Окно «I/O Ports» позволяет наблюдать и побитно модифицировать состояние всех линий ввода-вывода всех портов ADuC824. Окно включает поля «Output Latch» и «Input», которые отображают соответственно побитное содержимое регистров портов и внешних уровней на ножках портов микроконвертора. Окно «Program Analysis» предоставляет пользователю информацию о ходе процесса симуляции программы, о чем будет подробнее рассказано ниже.

Подопция «Keypad 4×5» позволяет симулировать внешнюю клавиатуру, подключенную к любым линиям ввода–вывода портов микроконвертора. При выборе этой подопции открывается окно «4×5 Keypad Simulator», в поле которого размещена клавиатурная матрица 4×5. Подробнее об использовании этого окна будет рассказано при описании опции меню «Configuration».

Перед началом симуляции рекомендуется открыть несколько наиболее необходимых для анализа выполнения целевой программы окон из опции «View» и разместить их в пределах главного окна симулятора, например, как показано на рис. 2.18. Это даст возможность более детально контролировать процесс выполнения программы и текущее состояние ядра и периферии симулируемого микроконвертора.

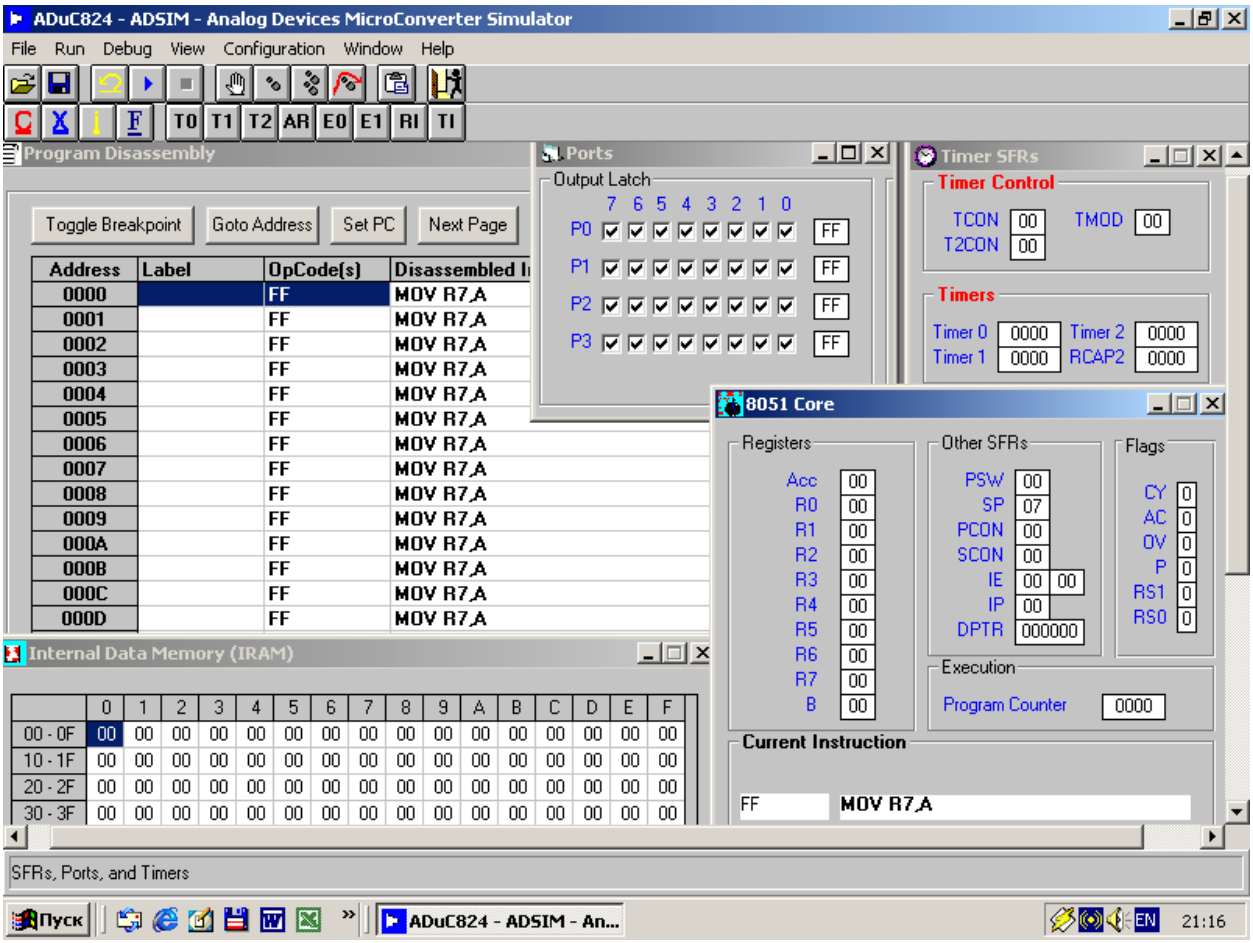


Рис. 2.18. Размещение окно перед началом симуляции

Для загрузки файла с целевым кодом в симулятор необходимо в опции меню «File» выбрать подопцию «Open HEX File». В открывшемся окне «Open» следует выбрать объектный файл с расширением .hex, ранее полученный в результате трансляции исходного текста целевой программы, и нажать на кнопку «OK». В результате окно «Program Disassembly» окажется модифицированным, в нем появятся коды операции и дизассемблированные команды, как показано на рис. 2.19. Заголовок окна «ADSim» должен теперь включать строку, указывающую на загруженный объектный файл и путь к нему, как это показано на рисунке. Далее следует в опции меню «File» выбрать подопцию «Import Map File» и в открывшемся окне «Open» выбрать файл с тем же именем, что и объектный, но с расширением .lst. Это приведет к модификации экрана симулятора в виде появления имен функций и меток в программе вместо адресов этих функций и меток.

Далее в опции меню «Run» нужно выбрать подопцию «Reset Program» или однократно щелкнуть левой клавишей «мыши» на кнопке «Reset» на панели инструментов. После запроса и получения подтверждения действия курсор в виде закрашенной строки, указывающий на текущий адрес нахождения управления в программе, переместится в позицию адреса 0000 программной памяти, то есть произойдет симуляция сброса микроконвертора.

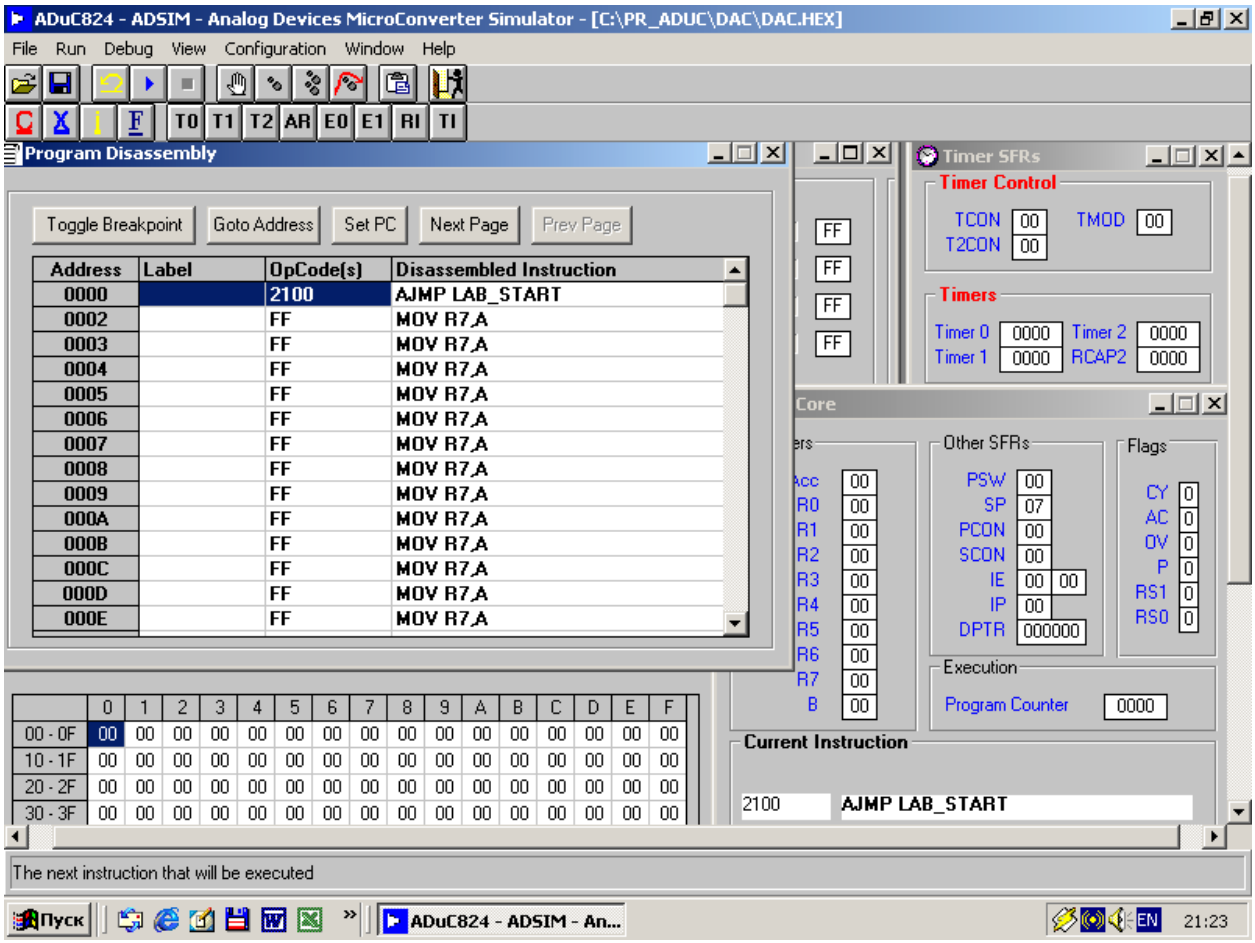


Рис. 2.19. Дизассемблирование программы

Симуляция целевой программы осуществляется командами, находящимися в опциях меню «Run» и «Debug». Часть этих команд дублируется кнопками на панели инструментов. Команда «Execute Program» запускает выполнение программы в непрерывном режиме, остановить которое можно командой «Stop». По команде «Single Step» производится выполнение одной, помеченной курсором команды с учетом переходов и вызовов подпрограмм. По команде «Procedure Step» также производится выполнение одной команды, но в случае, если это команда вызова подпрограммы, подпрограмма выполняются скрытно, передач управления внутри подпрограммы в процессе ее выполнения пользователь не видит. По команде «Skip Step» производится пропуск очередной команды без ее выполнения и без каких-либо изменений в эмулируемом микроконверторе. Результаты выполнения всех этих команд удобно наблюдать с помощью окна «Program Analysis». В этом окне можно измерять в относительных и абсолютных единицах, как долго длится выполнение того или иного фрагмента выполняемой целевой программы. В процессе симуляции в поле метки «Instructions» поля «Execution» окна «Program Analysis» производится отображение количества выполненных инструкций с момента последнего нажатия на кнопку «Reset Inst. Count», а в полях меток «Cycles» и «CPU Time» отображается соответственно количество прошедших машинных циклов вычислительного

ядра и прошедшее машинное (системное) время в часах, минутах и секундах с момента последнего нажатия на кнопку «Reset Cycles». В поле метки «Initialized To» поля «Stack» отображается значение, записанное целевой программой в регистр-указатель стека при инициализации стека, если таковая инициализация в программе имела место. В поле метки «Max Stack Size» отображается текущее значение содержимого регистра-указателя вершины стека. Поле «Execution History» отображает «историю» эмуляции – последовательность всех выполненных эмулятором команд целевой программы в порядке их выполнения. Поле «Unresolved Function Calls» отображает последовательность всех вызванных подпрограмм целевой программы в порядке следования их вызовов. Вид окна «Program Analysis», открытого во время симуляции целевой программы, показан на рис. 2.20.

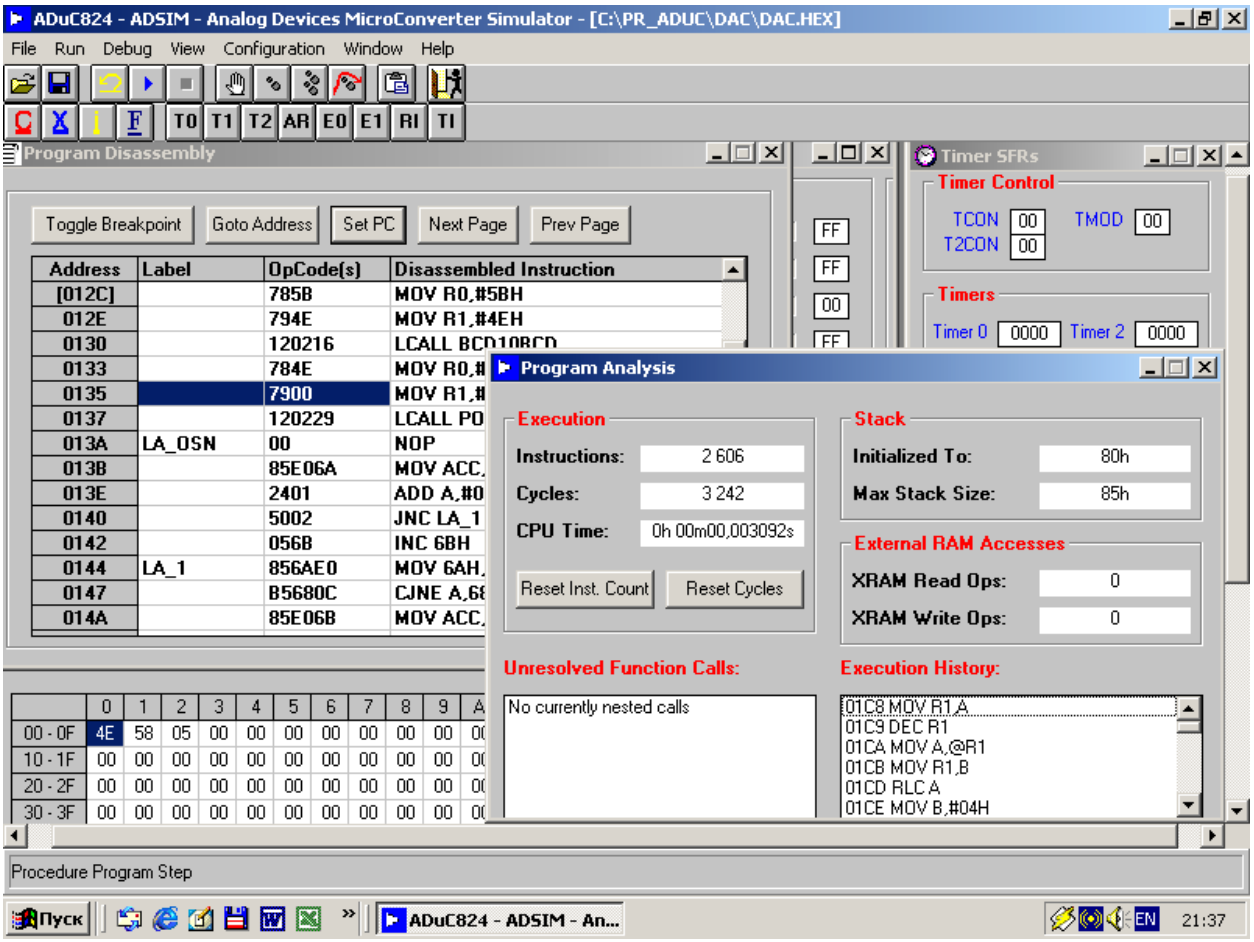


Рис. 2.20. Окно «Program Analysis» во время симуляции

Для программной симуляции запросов прерываний служат кнопки на инструментальной панели ADSim. Однократный щелчок левой клавиши мыши на кнопке генерации запроса прерывания вызовет на следующем шаге симуляции передачу управления на вектор этого прерывания в программной памяти при условии, что оно разрешено. Таким способом можно генерировать прерывания от системных таймеров 0, 1, 2 (кнопки «T0», «T1», «T2», «AR»),

внешние прерывания 0 и 1 («E0, «E1»), и прерывания по приему и передаче через UART («RI», «TI»).

Очистку программной памяти, а также внутреннего и внешнего ОЗУ микроконвертора можно произвести командами соответственно «Clear Program Memory», «Clear Internal Ram» и «Clear External Ram» опции меню «Debug». Эти команды дублируются кнопками на инструментальной панели. Команда «Clear Flash RAM» производит сброс (запись во все ячейки код 0FFh) Flash/EE-памяти данных ADuC824.

Помимо поля отображения программной памяти в окне «Program Disassembly» имеется еще несколько кнопок, о назначении которых стоит рассказать подробнее. При нажатии на кнопку «Goto Address» открывается окно «Set New Disassembly Address». В поле метки «Value (Hex)» этого окна можно ввести значение адреса программной памяти, по которому требуется передать управление. При нажатии на кнопку «Ok» курсор переместится на этот адрес, но управление по нему не будет передано до тех пор, пока не будет нажата кнопка установки программного счетчика «Set PC». Если после нажатия этой кнопки дать команду на выполнение программы, выполнение продолжится уже с нового адреса. Если кнопка «Set PC» после перемещения на новый адрес не нажималась, выполнение продолжится с прежнего адреса.

В окне «Program Disassembly» с учетом полного хода полосы прокрутки доступна для просмотра не вся программная память, а только одна ее условная страница, включающая сто адресов. Кнопки «Next Page» и «Prev Page» служат для перемещения по таблице программной памяти между страницами. При этих перемещениях, по аналогии с кнопкой «Goto Address», передача управления по новому адресу произойдет только в том случае, если после перемещения будет нажата кнопка установки программного счетчика «Set PC». Кроме того, для указания адреса передачи управления строку с этим адресом необходимо предварительно пометить курсором.

Удобным средством для отладки целевых программ являются контрольные точки (точки останова). Работа с контрольными точками производится с использованием нескольких команд из опции меню «Debug». Команда «Toggle Breakpoint» этой опции устанавливает или снимает контрольную точку по адресу программной памяти, который в данный момент подсвечен курсором в окне «Program Disassembly». Эта команда дублируется кнопками на панелях инструментов главного окна симулятора и окна «Program Disassembly». В случае правильной установки контрольной точки значение адреса программной памяти в окне «Program Disassembly» заключается в квадратные скобки (на рис. 2.21).

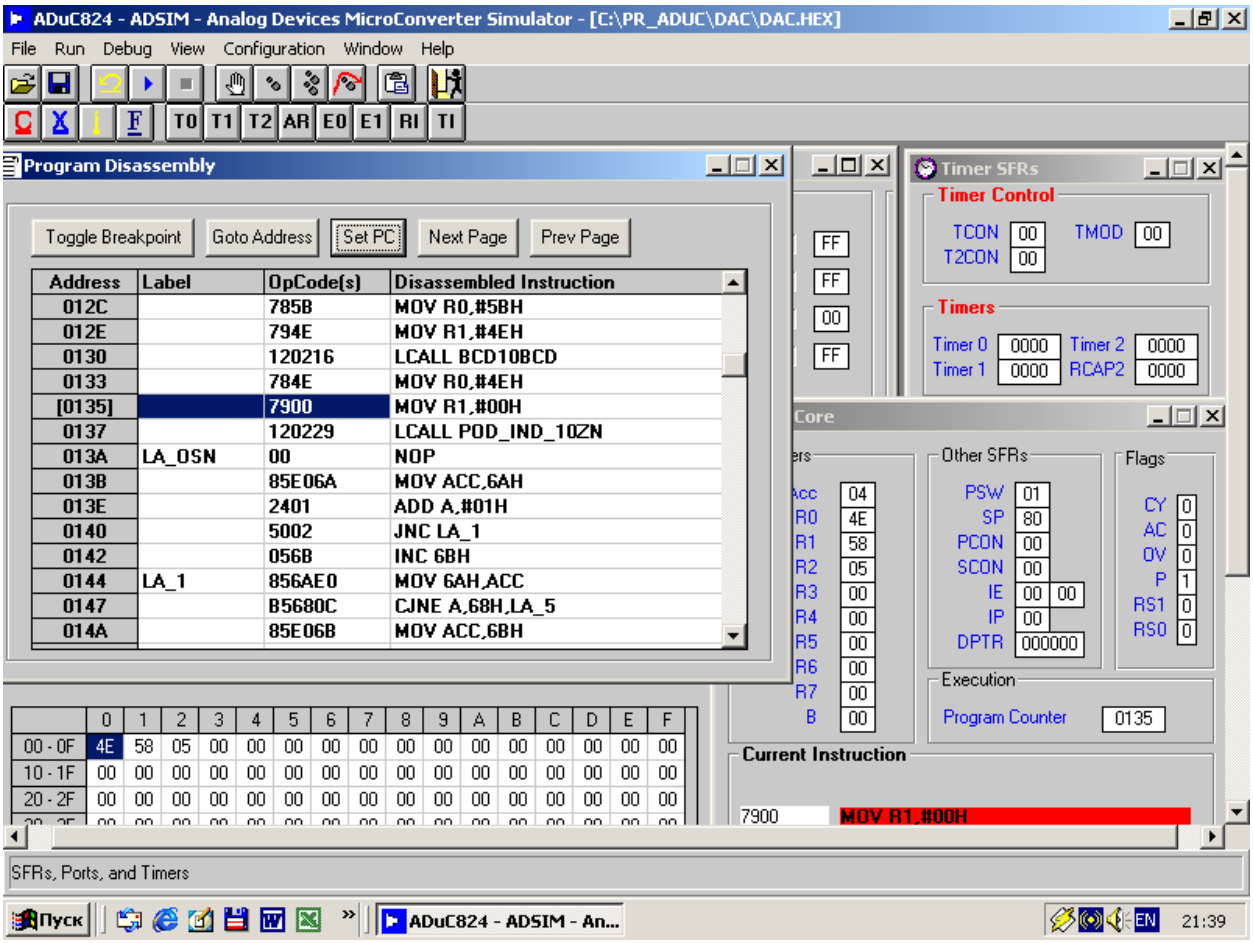


Рис. 2.21. Задание контрольной точки

Когда в процессе симуляции управление попадет на адрес, по которому была установлена контрольная точка, выполнение целевой программы на инструкции с этим адресом автоматически остановится. Команда «Clear All Breakpoints» используется для снятия всех ранее установленных в программе контрольных точек. Подопция «View/Edit Current Breakpoints» позволяет устанавливать или снимать контрольные точки по произвольному, а не только по текущему адресу. При выборе этой подопции открывается окно «Add/Remove Breakpoints», где в поле «Breakpoints» высвечивается список адресов, по которым в настоящий момент установлены контрольные точки. В окне «Add/Remove Breakpoints» имеется кнопка «Add Breakpoint», при нажатии на которую открывается окно «Set New Breakpoint». В поле метки «Value (Hex)» этого окна можно ввести значение адреса программной памяти, по которому требуется установить контрольную точку (рис. 2.22).

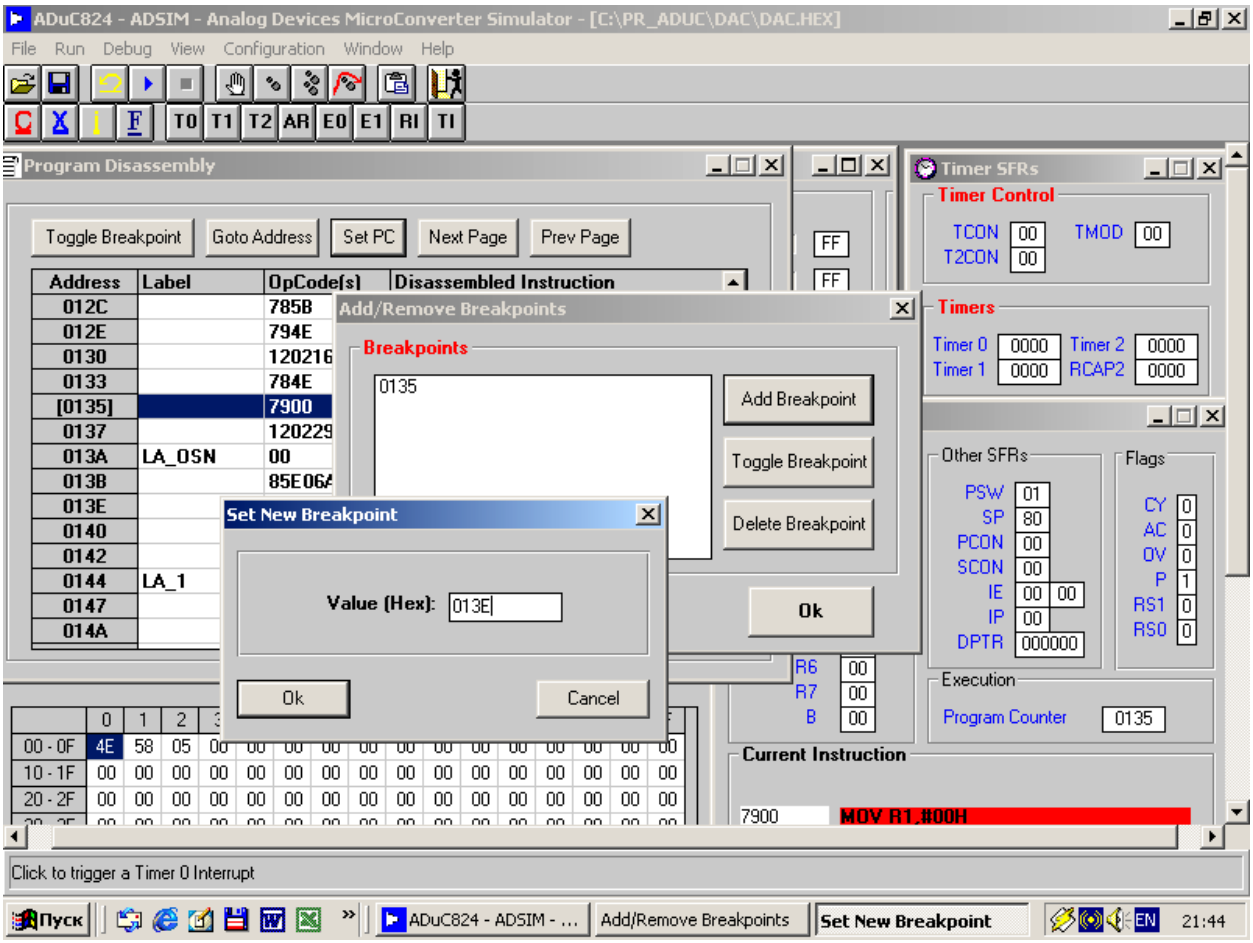


Рис. 2.22. Формирование контрольной точки

При нажатии на кнопку «Ok» контрольная точка будет установлена, а ее адрес добавится к списку в поле «Breakpoints» окна «Add/Remove Breakpoints». Путем нажатия на кнопку «Delete Breakpoints» можно снять контрольную точку по подсвеченному курсором адресу из списка поля «Breakpoints». При этом значение адреса точки из списка исчезнет. Нажатие на кнопку «Toggle Breakpoints» окна «Add/Remove Breakpoints» приведет к отключению контрольной точки по подсвеченному курсором адресу без ее снятия. При этом рядом со значением адреса в поле «Breakpoints» появится надпись «Disabled».

Результаты описанных манипуляций по установке и снятию контрольных точек можно отслеживать в окне «Program Disassembly», где все установленные на текущий момент контрольные точки будут автоматически помечаться заключением их адресов в программной памяти в квадратные скобки.

Опция меню «Configuration» предназначена, как уже упоминалось выше, для установки глобальных настроек целевого проекта (подопция «Project Options»), для установки настроек собственно симулятора (подопция «Simulator Configuration»), а также для эмуляции некоторых условий внешней среды микроконвертора при выполнении целевой программы.

Окно «Simulator Configuration» включает в себя несколько полей. В поле «Screen Update» можно устанавливать скорость симуляции. Установка скорости

осуществляется посредством задания в поле метки «Instructions per Execution Batch» значения количества программных инструкций, выполняемых за один такт симулятора. Необходимость в повышенной скорости, когда за один такт выполняется 100 – 1000 инструкций может возникнуть при симуляции программ с большим количеством циклов, опросов и т. п. В поле метки «Auto-Update Program Analysis Every___seconds» можно установить значение интервала времени в секундах, по истечении которого будут обновляться данные в окне «Program Analysis» в ходе симуляции выполнения программы.

В поле «Options» можно задать разрешение сброса всех контрольных точек при загрузке в симулятор объектного файла («Reset Breakpoints on HEX Load»), а также разрешение автоматического импорта имен меток и подпрограмм при загрузке объектного файла («Auto-Load MAP File on HEX Load»), однако, последняя установка будет работать, если только .lst-файл находится в одном каталоге с одноименным .hex-файлом.

В поле «Termilal Options» можно разрешить или запретить выбор окна «Uart Terminal Window» (опция меню «View»).

В окне «Keypad Configuration» производится настройка симулируемой клавиатурной матрицы, которая открывается командой «Keypad 4×5» опции меню «View». Матрица состоит из четырех горизонтальных линий («Line») и пяти вертикальных («Column»), которые образуют 20 пересечений и, соответственно, дает возможность симулировать 20 клавиш. В полях окна «Keypad Configuration»: «Line Selekt» и «Column Return Values» для каждой линии матрицы можно задать соединение с любой линией любого порта ввода-вывода микроконвертора, выбрав имя линии порта из выпадающего списка, или оставить линию матрицы неподключенной («Not Used»). По умолчанию, горизонтальные линии являются выходами (генераторами) сканирующих уровней, а вертикальные – входами (приемниками). В случае, если на некоторой горизонтальной линии матрицы установлен логический уровень, помеченный в поле «Selected When:», то при нажатии на любую клавишу матрицы, к которой подключена эта линия, на вертикальной линии, подключенной к этой клавише, появится логический уровень, помеченный в поле «Keypress Generates:».

Поясним работу с матрицей на следующем примере. Пусть наша симулируемая клавиатура будет состоять только из одной клавиши, включенной на пересечении линий ввода-вывода портов ADuC824 P1.0 (горизонтальная линия) и P0.0 (вертикальная линия). Остальные линии матрицы нами не используются. Сканирующий уровень на линии, подключенной к P0.0, зададим высокий. При нажатии на клавишу на линии, подключенной к P1.0, требуется автоматически генерировать также высокий уровень. Перечисленным условиям соответствуют настройки полей окна «Keypad Configuration» (на рис. 2.23).

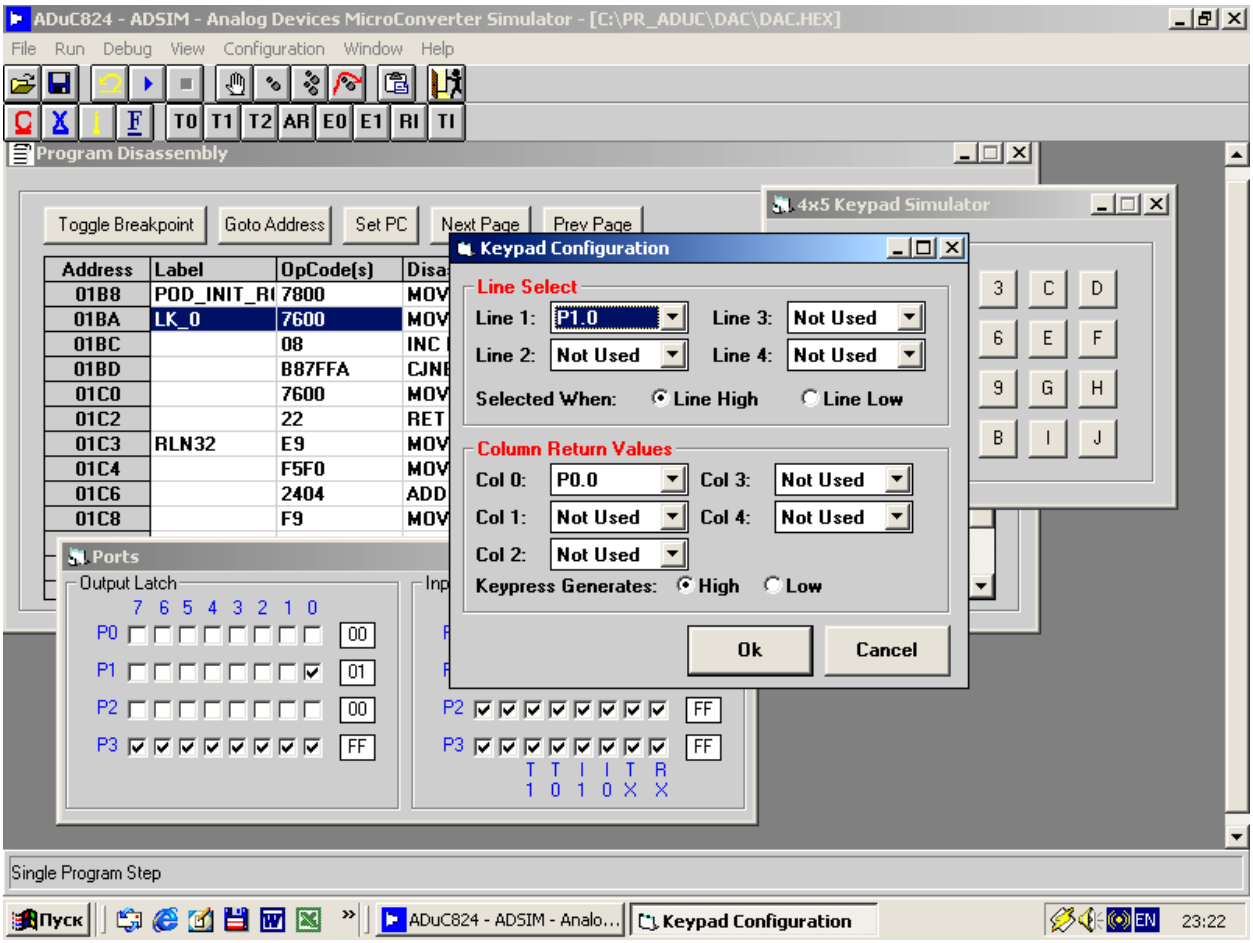


Рис. 2.23. Настройки для симуляции клавиатуры

Если во время симуляции целевой программы установить в поле «Output Latch» окна «Ports» P1.0 в высокий уровень, то при щелчке мышкой на клавише «1» клавиатурной матрицы «4×5 Keypad Simulator» P1.0 в поле «Input» окна «Ports» также перейдет в высокий уровень (рис. 2.24). При повторном щелчке мышкой на клавише «1» P1.0 вернется в низкий уровень, т. е. симулируемая клавиша имеет фиксацию.

При щелчке правой кнопкой мыши на любой клавише матрицы откроется окно, в поле которого можно задать этой клавише новое имя, например, «PR», взамен установленного по умолчанию (рис. 2.24).

Опция меню «File» помимо упоминавшихся выше команд также включает в себя команды сохранения и загрузки созданной пользователем среды симуляции. Среду симуляции образуют объектный файл и файл листинга с установленными контрольными точками. Командой «Save Current Simulation» среду можно сохранить в виде файла с расширением .sim. Команда «Open Simulation» загружает в симулятор ранее сохраненную среду из открывшегося окна дерева файлов и каталогов. Опция «File» включает в себя также команды сохранения и загрузки файла содержимого внешней оперативной памяти микроконвертора (*.xrm) «Save XRAM Memory File» и «Open XRAM Memory» соответственно.

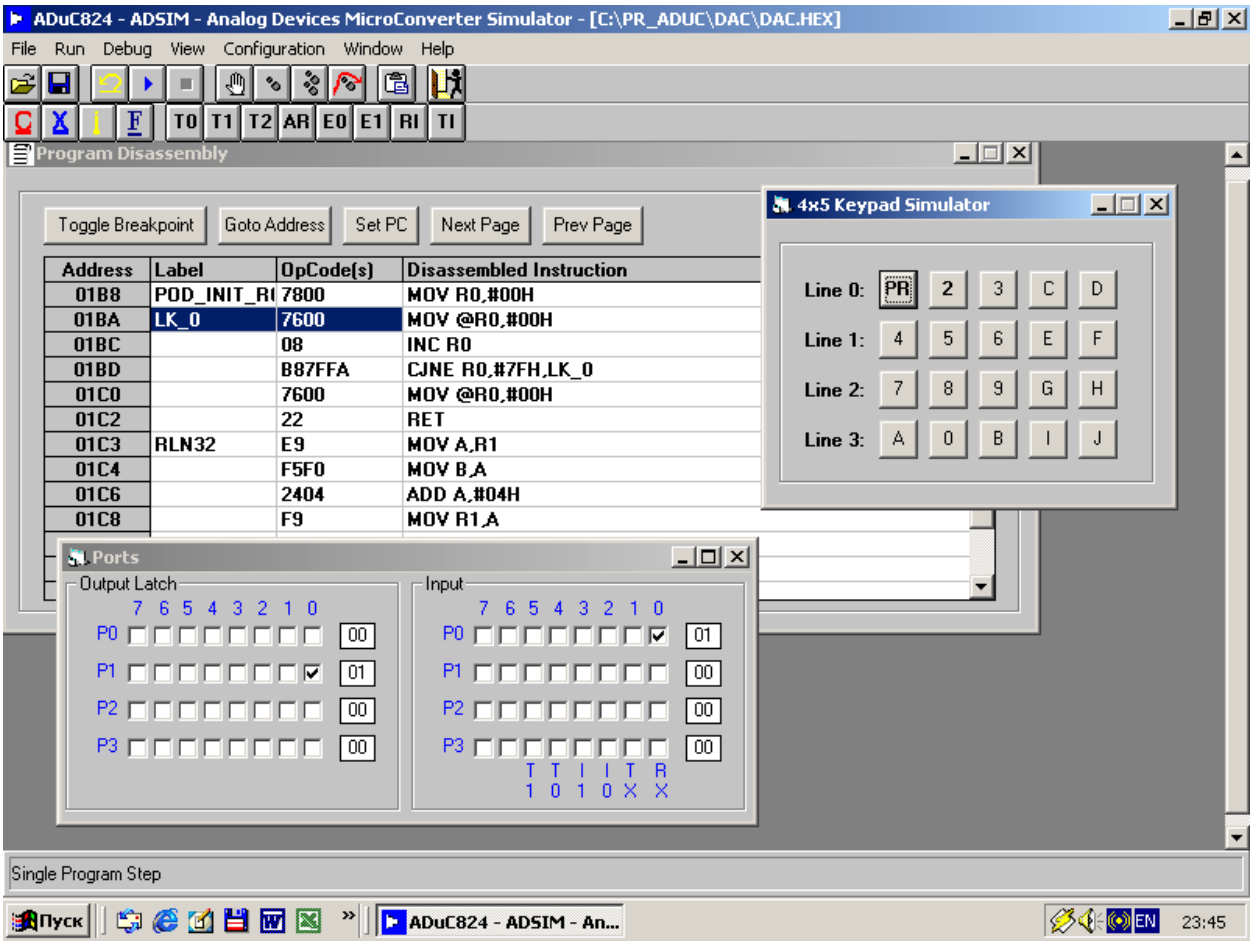


Рис. 2.24. Имитация нажатия клавиш

2.6. Программный анализатор АЦП WASP

WASP – это работающая под Windows программа, предназначенная для наблюдения и анализа с помощью компьютера результатов аналоговых измерений, производимых микроконвертором в реальном времени. Физически WASP поддерживает связь с ADuC824 через последовательный порт последнего и имеет встроенный режим автоматической загрузки в программную память ADuC824 собственной управляющей программы, производящей аналоговые измерения и передающей их результаты в последовательный порт. Программа, кроме того, позволяет через компьютер настраивать и конфигурировать аналоговые узлы микроконвертора посредством команд, передаваемых из ПК по COM-порту, а также осуществляет некоторую статистическую обработку результатов измерений.

Для корректного запуска WASP на демонстрационной плате следует разрешить режим последовательной загрузки, замкнув перемычку X1.

Для демонстрации работы WASP подадим на входы ADuC824 REFIN+ и REFIN– внешнее опорное напряжение 2,5 В от ИОНа AD780, например, как это

показано на рис. 3.2. Входы ADuC824 AIN1, AIN2 можно никуда не подключать. Затем подадим питание на эволюционную плату, предварительно подключив ее интерфейсным кабелем к COM-порту компьютера. Запуск WASP производится однократным щелчком левой клавиши мыши на значке «WASP», что вызывает запуск файла C:\ADuC\WASP\WASP.exe. В поле метки «MicroConverter Select» открывшегося окна следует выбрать из выпадающего списка требуемое устройство (ADuC824). Затем следует произвести загрузку в ADuC824 управляющей программы, щелкнув левой клавишей мыши на кнопке «Download». В случае, если все подготовительные операции проведены корректно, в окне должна появиться надпись, указывающая на факт обнаружения программой WASP микроконвертора (в данном случае – «ADuC824»), и шкала, графически отображающая процесс загрузки, как это показано на рис. 2.25. По окончании загрузки откроется окно «WASP – ADuC824», вид которого показан на рис. 2.26.

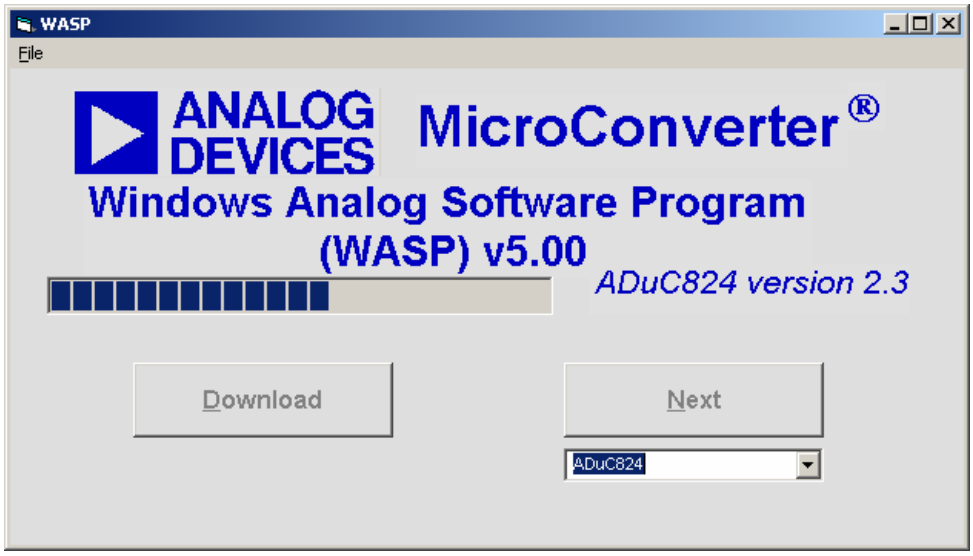


Рис. 2.25. Процесс загрузки программы в МК

Щелчок левой клавиши мыши на кнопке «Next» и запуск связанной с ней команды позволяет избежать процесса загрузки и сразу открыть окно «WASP – ADuC824», если управляющая программа поддержки WASP уже загружена в программную память ADuC824, а режим последовательной загрузки запрещен разомкнутой перемычкой X1. Наличие этой команды позволяет не расходовать попусту ресурс количества циклов стирания-записи микроконвертора. Перед ее подачей может потребоваться произвести сброс ADuC824 кнопкой сброса на эволюционной плате.

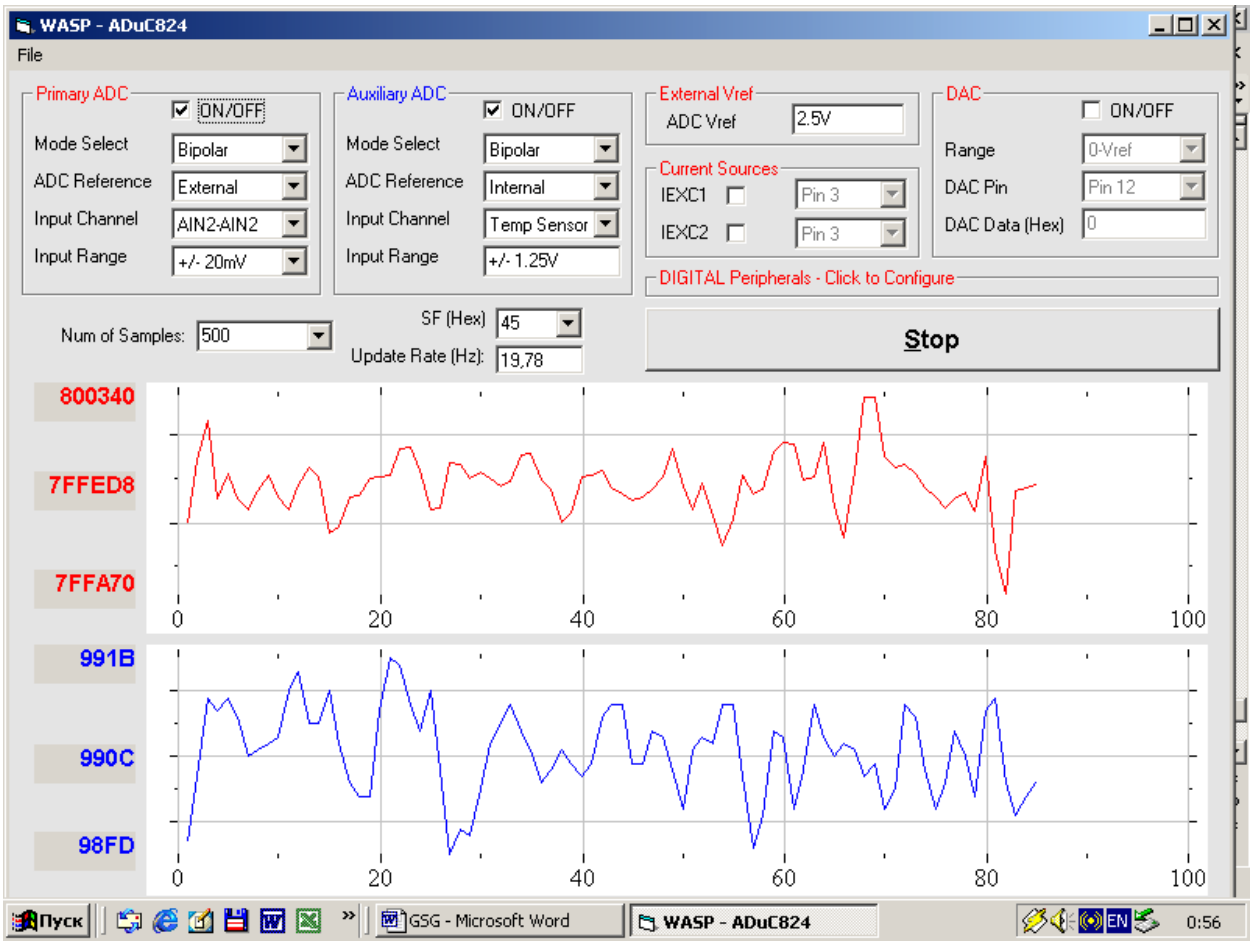


Рис. 2.26. Рабочее окно анализатора АЦП

Окно «WASP – ADuC824» включает в себя поля, позволяющие включать и настраивать все аналоговые аппаратные модули ADuC824. Как можно видеть из рис. 2.26, по умолчанию модуль основного АЦП включен, к нему подключены внутренне замкнутые входы AIN2-AIN2, преобразование будет осуществляться в биполярном режиме с пределами ±2,56 В и с использованием в качестве опорного напряжения 2,5 В от внешнего источника. Все остальные аналоговые модули по умолчанию отключены. В регистр специального назначения SF ADuC824 по умолчанию записано число 45h (модифицируется в поле метки «SF (Hex)»), что соответствует скорости преобразования 19,78 Гц (поле метки «Update Rate (Hz)»). При запуске преобразований для формирования выборки будет сделано 500 измерений (модифицируется в поле метки «Num of Samples»).

В поле «DIGITAL Peripherals-Click to Configure» доступно для просмотра и модификации содержимое некоторых специальных регистров цифровой периферии ADuC824.

Для примера включим помимо основного дополнительный модуль АЦП, который оставим подключенным по умолчанию к внутреннему температурному датчику ADuC824, а также установим пределы измерения основного модуля АЦП равными ±20 мВ.

Для запуска преобразований следует однократно щелкнуть левой клавишей мыши на кнопке «Run». Результаты преобразований для основного и дополнительного модулей АЦП будут отображаться в реальном времени по мере их поступления из ADuC824 в компьютер на соответствующих графических диаграммах окна «WASP – ADuC824», где по горизонтальным осям отложено количество измерений, а по вертикальным – результаты измерений в дискретах АЦП в шестнадцатеричной форме. Когда будет получена выборка из пятисот измерений, откроется окно «WASP – ADuC824 – ADC Noise Analysis», вид которого показан на рис. 2.27.

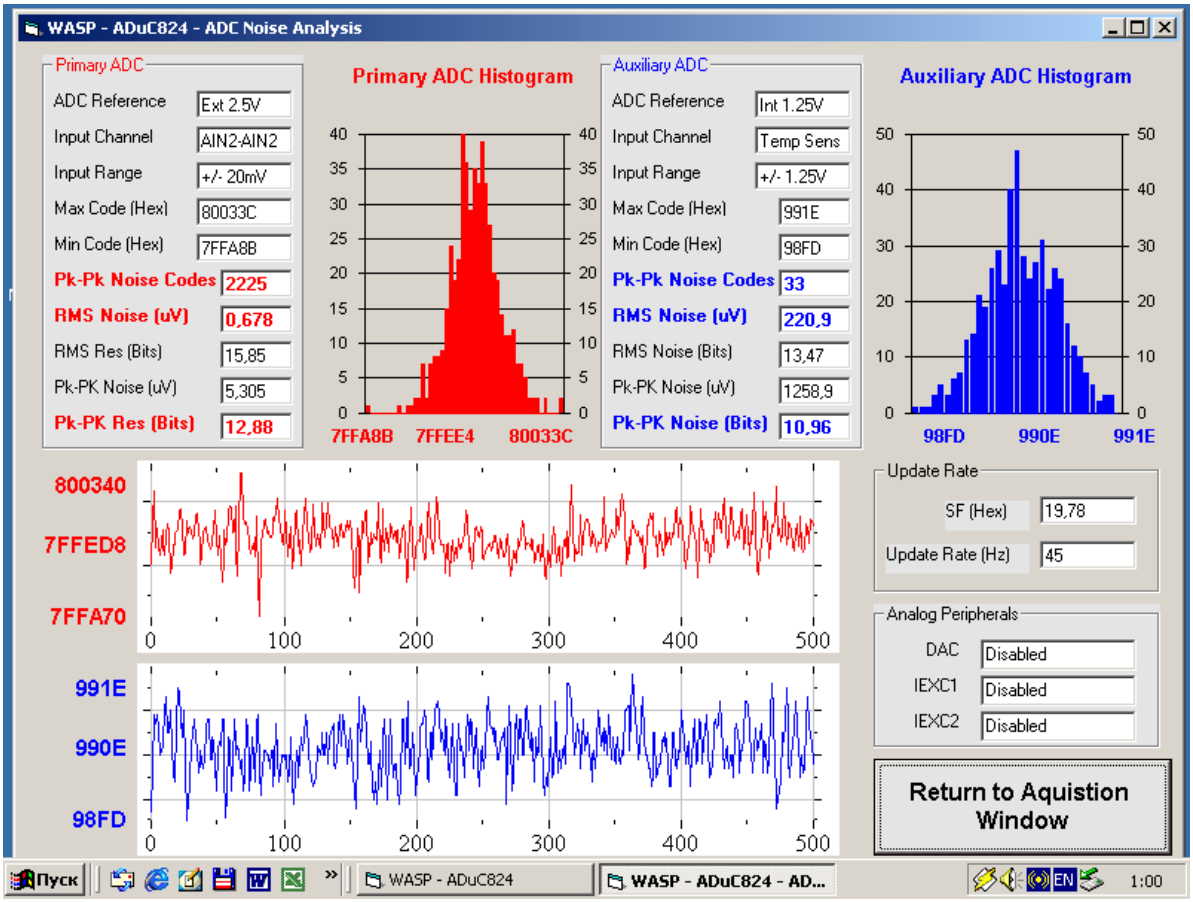


Рис. 2.27. Окно анализатора шума

По данным выборок результатов измерений основного и дополнительного модулей АЦП WASP построит в этом окне гистограммы, позволяющие оценить математическое ожидание (среднее по выборке) величины измеренного напряжения и его дисперсию (максимальное отклонение от среднего). Кроме того, будут вычислены и отображены в соответствующих полях максимальные и минимальные значения кодов измеряемых величин, интервалы «от пика до пика» в дискретах АЦП и микровольтах, а также вычисленное WASP реально достигнутое разрешение модулей АЦП. Как можно видеть из рисунка, код, возвращаемый модулем основного АЦП, колеблется около значения 800000h, что для биполярного режима измерений соответствует нулевому значению шкалы.

Это хорошо согласуется с текущей настройкой модуля АЦП, ведь измерения проводились на короткозамкнутой петле AIN2-AIN2. Код, возвращаемый модулем дополнительного АЦП, колеблется в данном случае около значения 990Eh. Зная среднее значение кода, можно вычислить температуру, которую кристалл ADuC824 имел при измерениях. Она равна (990Eh – 8000h) / 0FFh = 19h = 25 °C.

Любопытно проделать следующий эксперимент, демонстрирующий работу внутреннего температурного датчика микроконвертора. Перед запуском преобразований приложите кончик указательного пальца руки к поверхности корпуса микросхемы ADuC824 и удерживайте его в течение всего цикла преобразований. Возвращенное в результате среднее значение кода модуля дополнительного АЦП окажется существенно больше, чем в предыдущем случае из-за нагрева кристалла от пальца, а ломаная линия на графической диаграмме дополнительного АЦП в окне «WASP – ADuC824 – ADC Noise Analysis» будет отражать динамику нагрева, если мысленно сгладить шумовые выбросы на ней. Теперь осторожно нанесите на поверхность корпуса микросхемы маленькую каплю спирта или ацетона и сразу вслед за этим запустите преобразования. Возвращенное в результате среднее значение кода окажется существенно меньше, чем в первом случае из-за охлаждения кристалла в результате испарения спирта с поверхности корпуса микросхемы.

2.7. Вопросы для самоконтроля

1. Каков состав комплекса MicroConverter QuickStart Development System?
2. Каково назначение и принципы работы кросс-ассемблера Metalink 8051?
3. Каковы принципы работы последовательного загрузчика WSD?
4. С какой скоростью может производиться загрузка данных в МК под управлением WSD?
5. Какие задачи решает отладчик DeBug?
6. Как с помощью светодиода, подключенного к порту МК, можно отлаживать программу?
7. Зачем нужна катушка индуктивности L1 на рис. 2.8?
8. Каково назначение конденсаторов C5, C6 на рис. 2.8?
9. Как при отладке в DeBug просмотреть содержимое встроенной оперативной памяти?
10. Как работать с точками остановки в DeBug?
11. Каково назначение и принципы работы симулятора ADsim?
12. Как произвести дизассемблирование программы?
13. Можно ли в симуляторе ADsim имитировать работу внешнего ОЗУ?
14. Как работать с имитатором клавиатуры в ADsim?
15. Каково назначение и принципы работы анализатора АЦП WASP?

3. Демонстрационные программы для ADuC824

В этой части учебного пособия приводятся исходные тексты и описания ассемблерных программ, обеспечивающих работу основных периферийных узлов ADuC824. Каждая программа отлажена и является полностью законченным продуктом, реально работающим в «железе». Несмотря на свой отчасти демонстрационный характер эти программы, тем не менее, могут быть полезны разработчикам ПО для ADuC824 и для других 51-совместимых микроконтроллеров, так как содержат программно-аппаратные интерфейсы цифровой и аналоговой периферии микроконвертора и несколько программных интерфейсов внешних устройств.

Исходные тексты большинства предлагаемых программ помимо авторского кода содержат директиву $INCLUDE подключения файла preobr.asm (листинг 3.1), содержащего подпрограммы преобразования формы представления чисел, взятые из [9]. Эти подпрограммы используются при обработке результатов измерений, индикации адресов памяти и т. д. В качестве подключаемых также используются файл мнемоник ADuC824 mod824 (листинг 3.2), поставляемый в составе пакета QuickStart Development System, и файл дополнительных мнемоник ADuC824 824.inc (листинг 3.3), написанный Редькиным П.П.

Листинг 3.1. Преобразование формы представления чисел

```
;-------------------------------------------------------------------------------
; Подпрограммы преобразования представления чисел, используемые для обработки
; и вывода на индикацию результатов измерений и вычислений.
; (Файл  preobr.asm)
;-------------------------------------------------------------------------------

;-------------------------------------------------------------------------------
;Подпрограмма поворота 32-разрядного числа влево на 1 разряд.
;Вход: R1 - адрес мл байта числа (далее старшие байты)
;      CY - задвигаемый в число бит.
;Выход: CY - младший бит повернутого числа
;Использует подпрограммы: RLCN32
;-------------------------------------------------------------------------------
RLN32:    MOV       A,R1              ;
          MOV       B,A               ;coxp R1
          ADD       A,#4              ;
          MOV       R1,A              ;
          DEC       R1                ;
          MOV       A,@R1             ;
          MOV       R1,B              ;
          RLC       A                 ;CY = ст бит числа
;-------------------------------------------------------------------------------
;Подпрограмма логического сдвига 32-разрядного числа влево на 1 разряд через
;перенос.
;Вход: R1 - адрес мл байта числа (далее старшие байты)
;      CY - задвигаемый в число бит.
; Выход: CY - выдвинутый из числа бит.
;-------------------------------------------------------------------------------
RLCN32:   MOV       B,#4              ;задаем кол-во байт числа
RLCN_1:   MOV       A,@R1             ;---------------------------------
```

```
          RLC       A                   ; сдвиг байта на 1 бит с учетом CY
          MOV       @R1,A               ;---------------------------------
          INC       R1                  ;переход к более старшему байту
          DJNZ      B,RLCN_1            ;повторить для следующ байта
          MOV       B,PSW               ;сохр флаги
          MOV       A,R1                ;
          CLR       C                   ;
          SUBB      A,#4                ;
          MOV       R1,A                ;восст  R1
          MOV       PSW,B               ;восст флаги
          RET

;-------------------------------------------------------------------------------
;Подпрограмма преобразования 32-разрядного двоичного целого числа без знака в
;5-байтное целое беззнаковое упакованное двоично-десятичное число.
;      R0 - адрес буфера для результата, 5-байтного упакованного BCD числа.
;      В старш половине @R0 - ед миллиардов, в младш половине @R0+5  - единицы.
;
;      R1 - адрес 32-разрядного буфера с двоичным целым числом без знака.
;      В @R1 - младший байт, в @R1+4 - старш байт.
;Использует подпрограммы: RLN32
;-------------------------------------------------------------------------------
B32BCD:   MOV       A,#5                ;
B32BCD_1: MOV       @R0,#0              ;
          INC       R0                  ;
          DJNZ      ACC,B32BCD_1        ;обнулили результат
          DEC       R0                  ;
          MOV       A,#4                ;задаем кол-во байт двоичного числа
          RL        A                   ;
          RL        A                   ;
          RL        A                   ;
B32BCD_2: PUSH      ACC                 ;
          MOV       B,#5                ;
          CLR       C                   ;
B32BCD_3: MOV       A,@R0               ;
          ADDC      A,@R0               ;
          DA        A                   ;
          MOV       @R0,A               ;
          DEC       R0                  ;
          DJNZ      B,B32BCD_3          ;
          MOV       A,R0                ;
          ADD       A,#5                ;
          MOV       R0,A                ;
          LCALL     RLN32               ;повернули исходное число на 1 бит влево
          MOV       A,@R0               ;
          ADDC      A,#0                ;
          MOV       @R0,A               ;
          POP       ACC                 ;
          DJNZ      ACC,B32BCD_2        ;повторить для всех битов исходного числа
          MOV       A,R0                ;
          INC       A                   ;
          SUBB      A,#5                ;
          MOV       R0,A                ;восстановили R0
          RET

;-------------------------------------------------------------------------------
;Подпрограмма преобразования 5-байтового упакованного двоично-десятичн числа
;в 10-байтовое неупакованное двоично-десятичное число.
;R0 должен указывать на ст байт упакованного двоично-десятичного числа
;R1 должен указывать на ст байт (ед миллиардов) неупакованного двоичн-дес числа
;-------------------------------------------------------------------------------
BCD10BCD:    MOV        R2,#0               ;нач уст ст цикла
BCD10BCD_1:  MOV        A,@R0               ;---------------------
             SWAP       A                   ; выделяем ст тетраду
             ANL        A,#00001111b        ;---------------------
             MOV        @R1,A               ;

             MOV        A,@R0               ;---------------------
```

```
            ANL       A,#00001111b        ; выделяем мл тетраду
            INC       R1                  ;
            MOV       @R1,A               ;

            INC       R0                  ;
            INC       R1                  ;
            INC       R2                  ;
            CJNE      R2,#5,BCD10BCD_1    ;
            RET

;-------------------------------------------------------------------------------
;Подпрограмма преобразования 16-разрядного двоичного целого числа без знака в
;3-байтное целое беззнаковое упакованное двоично-десятичное число.
;      Вход:  R5R4 - 16-разрядное двоичное число (ст байт, мл байт).
;      Выход: R1R2R3 - двоично-десятичные цифры (мл половина R1 - дес тыс,
;             ст половина R2 - тыс,... мл половина R3 - ед.)
;-------------------------------------------------------------------------------
B16BCDD:    MOV       R1,#0               ;-------------------------
            MOV       R2,#0               ; нач обнуление результата
            MOV       R3,#0               ;-------------------------
            MOV       B,#16               ;
            SJMP      B16BCDD_2           ;пропуст первое (лишнее) умнож на 2
B16BCDD_1:  MOV       A,R3                ;
            ADD       A,R3                ;
            DA        A                   ;
            MOV       R3,A                ;
            MOV       A,R2                ;
            ADDC      A,R2                ;
            DA        A                   ;
            MOV       R2,A                ;
            MOV       A,R1                ;
            ADDC      A,R1                ;
            MOV       R1,A          ;R1R2R3 умнож на 2 по правилам 10-й системы
B16BCDD_2:  MOV       A,R4                ;
            RLC       A                   ;
            MOV       R4,A                ;
            MOV       A,R5                ;
            RLC       A                   ;
            MOV       R5,A                ;число сдвинуто влево
            MOV       A,R3                ;
            ADDC      A,#0                ;
            MOV       R3,A             ;добавили старш разряд к ед рез-та R1R2R3
            DJNZ      B,B16BCDD_1         ;зацикливание
            RET

;-------------------------------------------------------------------------------
;Подпрограмма преобразования 3-байтового упакованного двоично-десятичн числа
;в 6-байтовое неупакованное двоично-десятичное число.
;Упакованное двоично-десятичного число нужно предварительно поместить в R1R2R3
;(мл половина R1 - дес тыс, ст половина R2 - тыс,... мл половина R3 - ед.).
; Неупакованное двоичн-дес число возвращается в РОНах
; IND_DESTIS_R, IND_TIS_R, IND_SOT_R, IND_DES_R, IND_ED_R,
;-------------------------------------------------------------------------------
BCD5BCD:    MOV       A,R1                ;----------------------
            ANL       A,#00001111b        ; выделяем мл тетраду R1
            MOV       IND_DESTIS_R,A      ;получ дес тыс
            MOV       A,R2                ;----------------------
            SWAP      A                   ;
            ANL       A,#00001111b        ; выделяем ст тетраду R2
            MOV       IND_TIS_R,A         ;получ тыс
            MOV       A,R2                ;----------------------
            ANL       A,#00001111b        ; выделяем мл тетраду R2
            MOV       IND_SOT_R,A         ;получ сот
            MOV       A,R3                ;----------------------
            SWAP      A                   ;
            ANL       A,#00001111b        ; выделяем ст тетраду R3
            MOV       IND_DES_R,A         ;получ дес
            MOV       A,R3                ;----------------------
```

```
            ANL         A,#00001111b          ; выделяем мл тетраду R3
            MOV         IND_ED_R,A            ;получ ед
            RET
```

Листинг 3.2. Файл мнемоник ADuC824

```
;REV.  1.0    1 Dec 2000
;ADuC824   Apps, Analog Devices Inc.
P0        DATA  080H  ;PORT 0
SP        DATA  081H  ;STACK POINTER
DPL       DATA  082H  ;DATA POINTER - LOW BYTE
DPH       DATA  083H  ;DATA POINTER - HIGH BYTE
DPP       DATA  084H  ;DATA POINTER - PAGE BYTE
PCON      DATA  087H  ;POWER CONTROL
TCON      DATA  088H  ;TIMER CONTROL
TMOD      DATA  089H  ;TIMER MODE
TL0       DATA  08AH  ;TIMER 0 - LOW BYTE
TL1       DATA  08BH  ;TIMER 1 - LOW BYTE
TH0       DATA  08CH  ;TIMER 0 - HIGH BYTE
TH1       DATA  08DH  ;TIMER 1 - HIGH BYTE
P1        DATA  090H  ;PORT 1
SCON      DATA  098H  ;SERIAL PORT CONTROL
SBUF      DATA  099H  ;SERIAL PORT BUFFER
I2CDAT    DATA  09AH  ;I2C DATA BUFFER
I2CADD    DATA  09BH  ;I2C ADDRESS
P2        DATA  0A0H  ;PORT 2
TIMECON   DATA  0A1H  ;TIME COUNTER CONTROL REGISTER
HTHSEC    DATA  0A2H  ;1/128 OF A SECOND COUNTER
SEC       DATA  0A3H  ;SECONDS COUNTER
MIN       DATA  0A4H  ;MINUTES COUNTER
HOUR      DATA  0A5H  ;HOURS COUNTER
INTVAL    DATA  0A6H  ;TIMER INTERVAL
IE        DATA  0A8H  ;INTERRUPT ENABLE 1
IEIP2     DATA  0A9H  ;INTERRUPT ENABLE/PRIORITY 2
P3        DATA  0B0H  ;PORT 3
IP        DATA  0B8H  ;INTERRUPT PRIORITY 1
ECON      DATA  0B9H  ;FLASH CONTROL
EDATA1    DATA  0BCH  ;FLASH DATA1
EDATA2    DATA  0BDH  ;FLASH DATA2
EDATA3    DATA  0BEH  ;FLASH DATA3
EDATA4    DATA  0BFH  ;FLASH DATA4
WDCON     DATA  0C0H  ;WATCHDOG TIMER CONTROL
CHIPID    DATA  0C2H  ;CHIP ID REGISTER
EADRL     DATA  0C6H  ;EEPROM ADDRESS LOW
T2CON     DATA  0C8H  ;TIMER 2 CONTROL
RCAP2L    DATA  0CAH  ;TIMER 2 CAPTURE REGISTER - LOW BYTE
RCAP2H    DATA  0CBH  ;TIMER 2 CAPTURE REGISTER - HIGH BYTE
TL2       DATA  0CCH  ;TIMER 2 - LOW BYTE
TH2       DATA  0CDH  ;TIMER 2 - HIGH BYTE
PSW       DATA  0D0H  ;PROGRAM STATUS WORD
ADCMODE   DATA  0D1H  ;ADC MODE REGISTER
ADC0CON   DATA  0D2H  ;MAIN ADC CONFIG REGISTER
ADC1CON   DATA  0D3H  ;AUX ADC CONFIG REGISTER
SF        DATA  0D4H  ;DECIMATION FACTOR
ICON      DATA  0D5H  ;CURRENT SOURCE CONTROL REGISTER
PLLCON    DATA  0D7H  ;CRYSTAL PLL CONTROL REGISTER
ADCSTAT   DATA  0D8H  ;ADC STATUS REGISTER
ADC0L     DATA  0D9H  ;MAIN ADC DATA REGISTER
ADC0M     DATA  0DAH  ;MAIN ADC DATA REGISTER
ADC0H     DATA  0DBH  ;MAIN ADC DATA REGISTER
ADC1L     DATA  0DCH  ;AUX ADC DATA REGISTER
ADC1H     DATA  0DDH  ;AUX ADC DATA REGISTER
PSMCON    DATA  0DFH  ;POWER SUPPLY MONITOR
ACC       DATA  0E0H  ;ACCUMULATOR
OF0L      DATA  0E1H  ;MAIN ADC OFFSET REGISTER
OF0M      DATA  0E2H  ;MAIN ADC OFFSET REGISTER
OF0H      DATA  0E3H  ;MAIN ADC OFFSET REGISTER
OF1L      DATA  0E4H  ;AUX ADC OFFSET REGISTER
```

```
OF1H     DATA  0E5H  ;AUX ADC OFFSET REGISTER
I2CCON   DATA  0E8H  ;I2C CONTROL
GN0L     DATA  0E9H  ;MAIN ADC GAIN REGISTER
GN0M     DATA  0EAH  ;MAIN ADC GAIN REGISTER
GN0H     DATA  0EBH  ;MAIN ADC GAIN REGISTER
GN1L     DATA  0ECH  ;AUX ADC GAIN REGISTER
GN1H     DATA  0EDH  ;AUX ADC GAIN REGISTER
B        DATA  0F0H  ;MULTIPLICATION REGISTER
SPIDAT   DATA  0F7H  ;SPI DATA REGISTER
SPICON   DATA  0F8H  ;SPI CONTROL REGISTER
DACL     DATA  0FBH  ;DAC LOW BYTE
DACH     DATA  0FCH  ;DAC HIGH BYTE
DACCON   DATA  0FDH  ;DAC CONTROL REGISTER
IT0      BIT   088H  ;TCON.0 - EXT. INTERRUPT 0 TYPE
IE0      BIT   089H  ;TCON.1 - EXT. INTERRUPT 0 EDGE FLAG
IT1      BIT   08AH  ;TCON.2 - EXT. INTERRUPT 1 TYPE
IE1      BIT   08BH  ;TCON.3 - EXT. INTERRUPT 1 EDGE FLAG
TR0      BIT   08CH  ;TCON.4 - TIMER 0 ON/OFF CONTROL
TF0      BIT   08DH  ;TCON.5 - TIMER 0 OVERFLOW FLAG
TR1      BIT   08EH  ;TCON.6 - TIMER 1 ON/OFF CONTROL
TF1      BIT   08FH  ;TCON.7 - TIMER 1 OVERFLOW FLAG
T2       BIT   090H  ;P1.0 - TIMER 2 TRIGGER INPUT
T2EX     BIT   091H  ;P1.1 - TIMER 2 COUNT INPUT
RI       BIT   098H  ;SCON.0 - RECEIVE INTERRUPT FLAG
TI       BIT   099H  ;SCON.1 - TRANSMIT INTERRUPT FLAG
RB8      BIT   09AH  ;SCON.2 - RECEIVE BIT 8
TB8      BIT   09BH  ;SCON.3 - TRANSMIT BIT 8
REN      BIT   09CH  ;SCON.4 - RECEIVE ENABLE
SM2      BIT   09DH  ;SCON.5 - SERIAL MODE CONTROL BIT 2
SM1      BIT   09EH  ;SCON.6 - SERIAL MODE CONTROL BIT 1
SM0      BIT   09FH  ;SCON.7 - SERIAL MODE CONTROL BIT 0
EX0      BIT   0A8H  ;IE.0 - EXTERNAL INTERRUPT 0 ENABLE
ET0      BIT   0A9H  ;IE.1 - TIMER 0 INTERRUPT ENABLE
EX1      BIT   0AAH  ;IE.2 - EXTERNAL INTERRUPT 1 ENABLE
ET1      BIT   0ABH  ;IE.3 - TIMER 1 INTERRUPT ENABLE
ES       BIT   0ACH  ;IE.4 - SERIAL PORT INTERRUPT ENABLE
ET2      BIT   0ADH  ;IE.5 - TIMER 2 INTERRUPT ENABLE
EADC     BIT   0AEH  ;IE.6 - ENABLE ADC INTURRUPT
EA       BIT   0AFH  ;IE.7 - GLOBAL INTERRUPT ENABLE
RXD      BIT   0B0H  ;P3.0 - SERIAL PORT RECEIVE INPUT
TXD      BIT   0B1H  ;P3.1 - SERIAL PORT TRANSMIT OUTPUT
INT0     BIT   0B2H  ;P3.2 - EXTERNAL INTERRUPT 0 INPUT
INT1     BIT   0B3H  ;P3.3 - EXTERNAL INTERRUPT 1 INPUT
T0       BIT   0B4H  ;P3.4 - TIMER 0 COUNT INPUT
T1       BIT   0B5H  ;P3.5 - TIMER 1 COUNT INPUT
WR       BIT   0B6H  ;P3.6 - WRITE CONTROL FOR EXT. MEMORY
RD       BIT   0B7H  ;P3.7 - READ CONTROL FOR EXT. MEMORY
PX0      BIT   0B8H  ;IP.0 - EXTERNAL INTERRUPT 0 PRIORITY
PT0      BIT   0B9H  ;IP.1 - TIMER 0 PRIORITY
PX1      BIT   0BAH  ;IP.2 - EXTERNAL INTERRUPT 1 PRIORITY
PT1      BIT   0BBH  ;IP.3 - TIMER 1 PRIORITY
PS       BIT   0BCH  ;IP.4 - SERIAL PORT PRIORITY
PT2      BIT   0BDH  ;IP.5 - TIMER 2 PRIORITY
PADC     BIT   0BEH  ;IP.6 - ADC PRIORITY
WDWR     BIT   0C0H  ;WDCON.0 - WATCHDOG WRITE ENABLE BIT
WDE      BIT   0C1H  ;WDCON.1 - WATCHDOG ENABLE
WDS      BIT   0C2H  ;WDCON.2 - WATCHDOG STATUS
WDIR     BIT   0C3H  ;WDCON.3 - WATCHDOG INTERRUPT RESPONSE BIT
PRE0     BIT   0C4H  ;WDCON.4 - WATCHDOG TIMEOUT SELECTION BIT0
PRE1     BIT   0C5H  ;WDCON.5 - WATCHDOG TIMEOUT SELECTION BIT1
PRE2     BIT   0C6H  ;WDCON.6 - WATCHDOG TIMEOUT SELECTION BIT2
PRE3     BIT   0C7H  ;WDCON.7 - WATCHDOG TIMEOUT SELECTION BIT3
CAP2     BIT   0C8H  ;T2CON.0 - CAPTURE OR RELOAD SELECT
CNT2     BIT   0C9H  ;T2CON.1 - TIMER OR COUNTER SELECT
TR2      BIT   0CAH  ;T2CON.2 - TIMER 2 ON/OFF CONTROL
EXEN2    BIT   0CBH  ;T2CON.3 - TIMER 2 EXTERNAL ENABLE FLAG
TCLK     BIT   0CCH  ;T2CON.4 - TRANSMIT CLOCK SELECT
RCLK     BIT   0CDH  ;T2CON.5 - RECEIVE CLOCK SELECTT
```

```
EXF2     BIT   0CEH  ;T2CON.6 - EXTERNAL TRANSITION FLAG
TF2      BIT   0CFH  ;T2CON.7 - TIMER 2 OVERFLOW FLAG
P        BIT   0D0H  ;PSW.0 - ACCUMULATOR PARITY FLAG
F1       BIT   0D1H  ;PSW.1 - FLAG 1
OV       BIT   0D2H  ;PSW.2 - OVERFLOW FLAG
RS0      BIT   0D3H  ;PSW.3 - REGISTER BANK SELECT 0
RS1      BIT   0D4H  ;PSW.4 - REGISTER BANK SELECT 1
F0       BIT   0D5H  ;PSW.5 - FLAG 0
AC       BIT   0D6H  ;PSW.6 - AUXILIARY CARRY FLAG
CY       BIT   0D7H  ;PSW.7 - CARRY FLAG
ERR1     BIT   0DAH  ;ADSTAT.2 - AUX ADC ERROR BIT
ERR0     BIT   0DBH  ;ADSTAT.3 - MAIN ADC ERROR BIT
NOXREF   BIT   0DCH  ;ADSTAT.4 - NO EXTERNAL REFERENCE BIT
CAL      BIT   0DDH  ;ADSTAT.5 - CALIBRATION BIT
RDY1     BIT   0DEH  ;ADSTAT.6 - READY BIT FOR AUX ADC
RDY0     BIT   0DFH  ;ADSTAT.7 - READY BIT FOR MAIN ADC
I2CI     BIT   0E8H  ;I2CCON.0 - I2C INTERRUPT FLAG
I2CTX    BIT   0E9H  ;I2CCON.1 - I2C DIRECTION TRANSFER
I2CRS    BIT   0EAH  ;I2CCON.2 - I2C RESET
I2CM     BIT   0EBH  ;I2CCON.3 - I2C MASTER MODE SELECT
MDI      BIT   0ECH  ;I2CCON.4 - I2C MASTER DATA IN BIT
MCO      BIT   0EDH  ;I2CCON.5 - I2C MASTER CLOCK
MDE      BIT   0EEH  ;I2CCON.6 - I2C MASTER DATA OUT ENABLE BIT
MDO      BIT   0EFH  ;I2CCON.7 - I2C MASTER MODE SDATA OUTPUT
SPR0     BIT   0F8H  ;SPICON.0 - SPI BITRATE SELECT BIT0
SPR1     BIT   0F9H  ;SPICON.1 - SPI BITRATE SELECT BIT1
CPHA     BIT   0FAH  ;SPICON.2 - SPI CLOCK PHASE SELECT
CPOL     BIT   0FBH  ;SPICON.3 - SPI CLOCK POLARITY SELECT
SPIM     BIT   0FCH  ;SPICON.4 - SPI MASTER/SLAVE MODE SELECT
SPE      BIT   0FDH  ;SPICON.5 - SPI INTERFACE ENABLE
WCOL     BIT   0FEH  ;SPICON.6 - SPI WRITE COLLISION ERROR FLAG
ISPI     BIT   0FFH  ;SPICON.7 - SPI INTERRUPT BIT
```

Листинг 3.3. Дополнительный файл мнемоник

```
;-------------------------------------------------------------------------
;Файл мнемоник дополнительных описаний ADuC824    824.inc
;-------------------------------------------------------------------------
           P0_0         BIT     080h       ;----------------------------
           P0_1         BIT     081h       ;
           P0_2         BIT     082h       ;  биты  P0
           P0_3         BIT     083h       ;
           P0_4         BIT     084h       ;
           P0_5         BIT     085h       ;
           P0_6         BIT     086h       ;
           P0_7         BIT     087h       ;----------------------------

           P1_0         BIT     090h       ;----------------------------
           P1_1         BIT     091h       ;
           P1_2         BIT     092h       ;  биты  P1
           P1_3         BIT     093h       ;
           P1_4         BIT     094h       ;
           P1_5         BIT     095h       ;
           P1_6         BIT     096h       ;
           P1_7         BIT     097h       ;----------------------------

           P2_0         BIT     0A0h       ;----------------------------
           P2_1         BIT     0A1h       ;
           P2_2         BIT     0A2h       ;  биты  P2
           P2_3         BIT     0A3h       ;
           P2_4         BIT     0A4h       ;
           P2_5         BIT     0A5h       ;
           P2_6         BIT     0A6h       ;
           P2_7         BIT     0A7h       ;----------------------------

           P3_0         BIT     0B0h       ;----------------------------
           P3_1         BIT     0B1h       ;
           P3_2         BIT     0B2h       ;  биты  P3
```

```
P3_3        BIT     0B3h        ;
P3_4        BIT     0B4h        ;
P3_5        BIT     0B5h        ;
P3_6        BIT     0B6h        ;
P3_7        BIT     0B7h        ;----------------------------

ACC_0       BIT     0E0h        ;----------------------------
ACC_1       BIT     0E1h        ;  биты
ACC_2       BIT     0E2h        ;
ACC_3       BIT     0E3h        ;  аккумулятора
ACC_4       BIT     0E4h        ;
ACC_5       BIT     0E5h        ;
ACC_6       BIT     0E6h        ;
ACC_7       BIT     0E7h        ;----------------------------
```

Для всех приведенных ниже исходных текстов, содержащихся в файлах с расширением .asm, имеются соответствующие им таблицы прошивок программной памяти микроконвертора, содержащиеся в одноименных файлах с расширением .hex.

При создании своих собственных проектов с использованием приведенных здесь материалов читателям следует обращать внимание на соответствие путей к подключаемым файлам, указанным в ассемблерных директивах $INCLUDE, реальному местоположению этих файлов на диске.

3.1. Интерфейс кнопок управления

При разработке проектов на базе различных микроконтроллеров и, в частности, на базе ADuC824 в большинстве случаев в составе устройства необходимо иметь какие-то органы управления, чаще всего, кнопки. При этом представляется оптимальным преодолеть связанные с применением кнопок недостатки, в частности, «дребезг» механических контактов, не аппаратно, а программно, в рамках встроенного программного интерфейса обслуживания кнопок, что позволит минимизировать аппаратные средства интерфейса (рис. 3.1). Некоторая громоздкость алгоритма компенсируется его гибкостью и универсальностью.

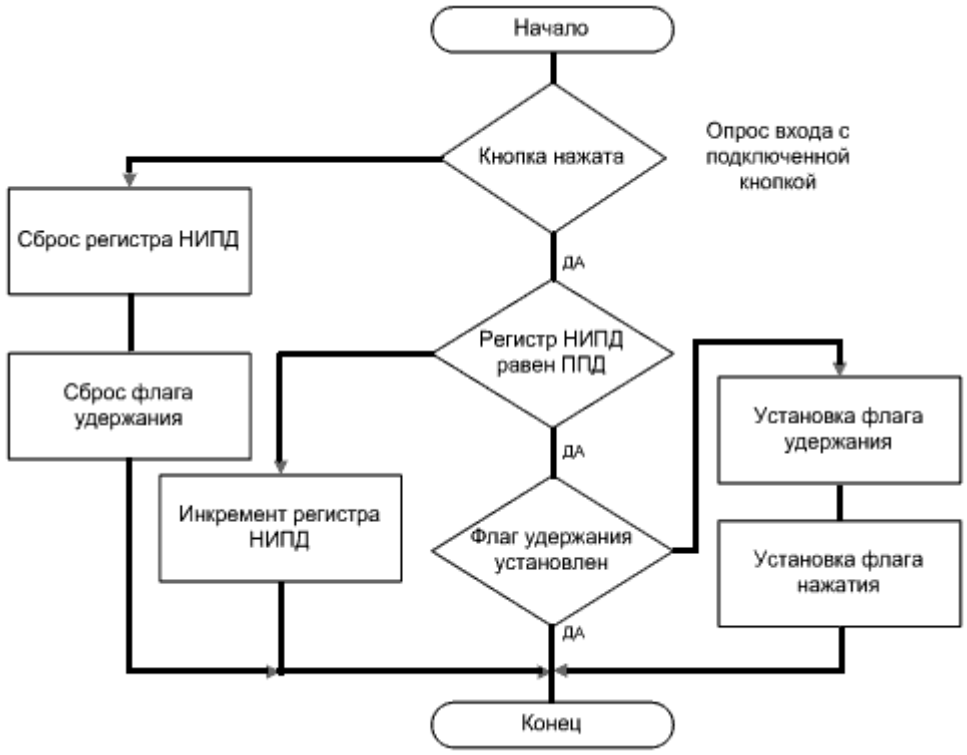


Рис. 3.1. Алгоритм обработки сигнала кнопки
(НИПД – накопление интервала подавления «дребезга»,
ППД – порог подавления «дребезга»)

Листинг 3.4. Программа обработки сигнала кнопки

```
;-------------------------------------------------------------------------------
;Подключаемый программный модуль обработки нажатий на кнопки для ADuC824.
;Обрабатывает нажатия на 6 кнопок: "0" - "5".
;При опросе кнопок производится косвенная адресация обслуживающих их регистров.
;Модуль содержит подпрограммы опроса кнопок "0" - "5" и подпрограммы обработки
;нажатия на кнопку и ненажатия на кнопку.
;-------------------------------------------------------------------------------
;-------------------------------------------------------------------------
;Подпрограмма опроса кнопки 0.
;-------------------------------------------------------------------------
Pod_OPR_KNOP0:
          JNB          _IN_KNOP0,Jj_0       ;
          ACALL        Pod_NNAG_KNOP        ;действия в случае ненажатой кнопки
          RET                               ;
Jj_0:     ACALL        Pod_NAG_KNOP         ;действия в случае нажатой кнопки
          RET

;-------------------------------------------------------------------------
;Подпрограмма опроса кнопки 1.
;-------------------------------------------------------------------------
Pod_OPR_KNOP1:
          JNB          _IN_KNOP1,Jj_1       ;
          ACALL        Pod_NNAG_KNOP        ;действия в случае ненажатой кнопки
          RET                               ;
Jj_1:     ACALL        Pod_NAG_KNOP         ;действия в случае нажатой кнопки
          RET

;-------------------------------------------------------------------------
;Подпрограмма опроса кнопки 2.
;-------------------------------------------------------------------------
Pod_OPR_KNOP2:
          JNB          _IN_KNOP2,Jj_2       ;
```

```
          ACALL       Pod_NNAG_KNOP        ;действия в случае ненажатой кнопки
          RET                              ;
Jj_2:     ACALL       Pod_NAG_KNOP         ;действия в случае нажатой кнопки
          RET

;-------------------------------------------------------------------------
;Подпрограмма опроса кнопки 3.
;-------------------------------------------------------------------------
Pod_OPR_KNOP3:
          JNB         _IN_KNOP3,Jj_3       ;
          ACALL       Pod_NNAG_KNOP        ;действия в случае ненажатой кнопки
          RET                              ;
Jj_3:     ACALL       Pod_NAG_KNOP         ;действия в случае нажатой кнопки
          RET

;-------------------------------------------------------------------------
;Подпрограмма опроса кнопки 4.
;-------------------------------------------------------------------------
Pod_OPR_KNOP4:
          JNB         _IN_KNOP4,Jj_4       ;
          ACALL       Pod_NNAG_KNOP        ;действия в случае ненажатой кнопки
          RET                              ;
Jj_4:     ACALL       Pod_NAG_KNOP         ;действия в случае нажатой кнопки
          RET

;-------------------------------------------------------------------------
;Подпрограмма опроса кнопки 5.
;-------------------------------------------------------------------------
Pod_OPR_KNOP5:
          JNB         _IN_KNOP5,Jj_5       ;
          ACALL       Pod_NNAG_KNOP        ;действия в случае ненажатой кнопки
          RET                              ;
Jj_5:     ACALL       Pod_NAG_KNOP         ;действия в случае нажатой кнопки
          RET

;-------------------------------------------------------------------------
;Подпрограмма производимых действий в случае ненажатой кнопки.
;-------------------------------------------------------------------------
Pod_NNAG_KNOP:
          MOV         @R1,#0               ;обнуление регистра накопления кнопки
          MOV         ACC,@R0              ;------------------------------
          CLR         ACC_0                ; сброс флага удержания кнопки
          MOV         @R0,ACC              ;------------------------------
          RET

;-------------------------------------------------------------------------
;Подпрограмма производимых действий в случае нажатой кнопки.
;-------------------------------------------------------------------------
Pod_NAG_KNOP:
          CJNE        @R1,#POROG_K,Jj_30   ;не накоплено ли знач подавл дреб ?
          MOV         ACC,@R0              ;накоплено
          JB          ACC_0,Jj_20          ;флаг удержания установлен ?
          SETB        ACC_0                ;нет, уст флаг удержания
          SETB        ACC_1                ;     уст флаг нажатия
          MOV         @R0,ACC              ;
Jj_20:    RET                              ;да
Jj_30:    INC         @R1                  ;не накоплено, копить дальше
          RET
```

Программа позволяет определить моменты нажатия и отпускания одной кнопки управления и при этом производит фильтрацию импульсов «дребезга» контактов. Для одной кнопки требуются следующие ресурсы микроконтроллера: один регистр общего назначения, назовем его регистром накопления интервала подавления «дребезга», и два флага (бита общего назначения), назовем их

флагом нажатия на кнопку и флагом удержания кнопки. Кроме этого, подпрограмма использует некоторую байтовую константу, назовем ее порогом подавления «дребезга». Как видно из блок-схемы, при каждом вызове подпрограммы при условии, что кнопка не нажата, регистр накопления будет обнуляться, а флаг удержания будет сбрасываться, то есть будет происходить постоянная установка цифрового программного фильтра «дребезга» в начальное состояние. Вызовы подпрограммы должны происходить циклически, для чего следует поместить инструкцию вызова либо в основной цикл управляющей программы микроконтроллера, либо, что более предпочтительно, в блок обработки периодических прерываний от какого-нибудь системного таймера. Когда кнопку нажимают, то при каждом вызове подпрограммы регистр накопления начинает инкрементироваться, причем содержимое регистра начнет реально увеличиваться только после окончания «дребезга», когда регистр перестанет сбрасываться из-за пауз между импульсами «дребезга». Когда процесс инкрементирования приобретет устойчивый характер, то через некоторое число вызовов подпрограммы содержимое регистра накопления станет равным порогу подавления «дребезга» и произойдет установка флагов нажатия и удержания кнопки. Флаг удержания будет оставаться в единичном состоянии столько времени, сколько удерживается кнопка, и сбросится при первом же импульсе «дребезга» в момент отпускания кнопки. Этот флаг можно использовать в целевой программе, например, для реализации каких-то дополнительных функций данной кнопки в пользовательском интерфейсе разрабатываемого устройства.

Флаг нажатия подпрограммой не сбрасывается. Его сброс возлагается на тот фрагмент пользовательской программы, который опрашивает состояние этого флага, т. е. реализует обработку обнаружения нажатия на данную кнопку.

Подбирая значение константы порога подавления «дребезга» в своей программе, пользователь может, во-первых, настраивать интерфейс под конкретный тип кнопок, во-вторых, под конкретную периодичность вызовов подпрограмм обслуживания кнопок и, наконец, устанавливать требуемое время реакции своего интерфейса на нажатия кнопок. С учетом этих соображений значение константы подавления «дребезга» может лежать в диапазоне от нескольких единиц до нескольких десятков.

Исходные тексты подпрограмм, написанных на ассемблере Metalink 8051 и реализующих описанный выше алгоритм обслуживания кнопок, содержатся в файле knop.asm. Этот файл в дальнейшем будет указываться как подключаемый в исходных текстах пользовательских программ для ADuC824, о которых будет рассказано в следующих главах. В файле содержатся подпрограммы обслуживания шести кнопок, каждая из которых подключена к отдельной линии ввода-вывода микроконвертора, предварительно сконфигурированной как вход. Эти линии в подпрограммах имеют имена _IN_KNOP0 – _IN_KNOP5. Для оптимизации кода каждая ветвь управления в подпрограммах Pod_OPR_KNOP0 – Pod_OPR_KNOP5 реализована подпрограммами более низкого уровня вложенности вызова, одинаковыми для всех кнопок: Pod_NNAG_KNOP – последова-

тельность действий, производимая алгоритмом в случае ненажатой кнопки, а Pod_NAG_KNOP – последовательность действий, производимая в случае нажатой кнопки. При этом предполагается, что кнопки являются нормально разомкнутыми и включены между входами ADuC824 и общим проводом устройства, а входы микроконвертора «подтянуты» к «плюсу» источника питания внешними резисторами. Перед тем, как вызвать подпрограмму обслуживания какой-нибудь кнопки, следует в основной программе косвенно адресовать регистр накопления этой кнопки с помощью регистра-указателя R1, а регистр, содержащий флаги нажатия и удержания этой кнопки, – с помощью регистра-указателя R0. Флаг удержания – бит 0 в регистре флагов, флаг нажатия – бит 1. Константа подавления «дребезга» в подпрограмме имеет имя POROG_K.

3.2. Интерфейс с ЖКИ

Наряду с органами управления в устройстве на базе микроконтроллера должны быть какие-то средства индикации, позволяющие пользователю получать информацию о режимах работы устройства, входных и выходных сигналах и т. п. В зависимости от сложности самого устройства отображаемая при его работе информация также может быть достаточно сложной, что предъявляет к применяемым индикаторам определенные требования по объему одновременно отображаемых данных. С другой стороны, интерфейс взаимодействия индикатора с устройством должен быть по возможности простым, чтобы минимально задействовать аппаратные и программные ресурсы микроконтроллера. Кроме того, для многих пользовательских приложений имеет значение мощность, потребляемая индикатором от источника питания. В настоящее время к наиболее распространенным индикаторам, адаптированным к использованию в составе устройств на микроконтроллерах, можно отнести символьные (буквенно-цифровые) жидкокристаллические модули со встроенным контроллером управления, совместимым с контроллером HD44780 фирмы HITACHI. Они поставляются на рынок несколькими фирмами-производителями (Data Vision, Bolymin, Povertip, SII, Optrex, EDT, Picture, Winstar, МЭЛТ и т. д.) и в значительной мере отвечают упомянутым выше требованиям. Эти индикаторы имеют малое энергопотребление, простой интерфейс (для взаимодействия с устройством им требуется от шести до одиннадцати линий, не считая питания и общего провода) и предоставляют очень широкие возможности в плане отображения информации. В предлагаемых далее демонстрационных и практических конструкциях использовался именно такой индикатор – WH1602A-NGG-CP фирмы Winstar (русифицированный, 16 символов в строке, 2 строки). Таблица кодов символов (фонтов) для этого (и других русифицированных) ЖКИ приводится в Приложении 3. Электрические параметры, временные диаграммы сигналов управления, набор команд индикаторов с HD44780-совместитимыми

контроллерами нет необходимости здесь приводить, так как они подробно описаны в литературе [10], [11].

Обмен данными устройства с индикатором возможен по восьми- или по четырехпроводной шине данных. В первом случае обмен происходит быстрее, но и аппаратных ресурсов у микроконтроллера задействовано больше на четыре линии ввода-вывода. Предлагаемое далее пользовательское программное обеспечение реализует обмен ADuC824 с ЖКИ только по четырехпроводной шине данных.

При взаимодействии ЖКИ с микроконтроллером возможен обмен данными, т. е. передача их в обе стороны, и возможна также односторонняя передача данных – только из микроконтроллера в ЖКИ. В первом случае пользовательское программное обеспечение будет более сложным, так как микроконтроллеру придется помимо передачи данных задавать режим (направление) передачи, а также опрашивать ЖКИ на предмет его готовности к приему и принимать данные от него. Во втором случае кроме упрощения программного обеспечения возможно несколько упростить и аппаратный интерфейс. Логический уровень на входе ЖКИ, определяющий его режим (запись/чтение), можно аппаратно установить для режима записи, а выход микроконтроллера, предназначенный для программной генерации этого сигнала, освободить. Готовность ЖКИ к приему данных микроконтроллер может вообще не проверять, а вместо этого программно генерировать временные задержки заведомо больших интервалов времени, необходимых ЖКИ на подготовку к приему. Платой за аппаратное и программное упрощение в этом случае будет более медленная работа интерфейса.

Исходные тексты подпрограмм, реализующих интерфейс обмена с ЖКИ по четырехпроводной шине данных с опросом состояния ЖКИ, содержатся в файле lcd_opr.asm (листинг 3.5). Этот файл в дальнейшем будет указываться как подключаемый в исходных текстах демонстрационных пользовательских программ для ADuC824. В файле содержатся следующие подпрограммы: подпрограмма инициализации ЖКИ после сброса при включении питания Pod_INIT_LCD, подпрограмма очистки экрана ЖКИ Pod_CLEAR_LCD, подпрограмма передачи в ЖКИ одной команды Pod_PER_COM_LCD, подпрограмма записи в ОЗУ ЖКИ одного байта данных (индикации на экране одного символа) Pod_PER_DAT_LCD, а также несколько вспомогательных подпрограмм более низкого уровня вложенности.

Листинг 3.5. Программа работы с ЖКИ

```
;-------------------------------------------------------------------------------
;Подключаемый программный модуль обслуживания ЖКИ с контроллером для ADuC824.
;
; Данные передаются по 4-битной шине, состояние ЖКИ опрашивается,
; подпрограмма опроса возвращает управление, когда ЖКИ готов к приему данных.
;-------------------------------------------------------------------------------

;-------------------------------------------------------------------------
;Подпрограмма инициализации ЖКИ.
;-------------------------------------------------------------------------
Pod_INIT_LCD:
```

```
          MOV         COM_IND_R,#00101000b   ;4-х битная шина, 2 стр, 5х7
          ACALL       Pod_PER_COM_LCD        ;
          MOV         COM_IND_R,#00000001b   ;очист диспл, курсор в нач полож
          ACALL       Pod_PER_COM_LCD        ;
          MOV         COM_IND_R,#00000110b   ;дисплей не сдвиг, курсор сдвиг
          ACALL       Pod_PER_COM_LCD        ;
          MOV         COM_IND_R,#00001100b   ;вкл дисплей, откл курсор
          ACALL       Pod_PER_COM_LCD        ;
          RET

;-------------------------------------------------------------------------
;Подпрограмма очистки ЖКИ.
;-------------------------------------------------------------------------
Pod_CLEAR_LCD:
          MOV         COM_IND_R,#00000001b   ;очист диспл, курсор в нач полож
          ACALL       Pod_PER_COM_LCD        ;
          RET

;-------------------------------------------------------------------------
;Подпрограмма побитного копирования старшей тетрады аккумулятора в старшую
;тетраду порта индикатора PORT_IND.
;-------------------------------------------------------------------------
Pod_TETR:
          MOV         C,ACC_4
          MOV         PORT_IND_4,C
          MOV         C,ACC_5
          MOV         PORT_IND_5,C
          MOV         C,ACC_6
          MOV         PORT_IND_6,C
          MOV         C,ACC_7
          MOV         PORT_IND_7,C
          RET

;-------------------------------------------------------------------------
;Подпрограмма генерации одного положительного импульса на линии E ЖКИ.
;-------------------------------------------------------------------------
Pod_IMP_E:
          SETB        E                      ;фронт импульса
          NOP                                ;
          CLR         E                      ;спад импульса
          RET

;-------------------------------------------------------------------------
;Подпрограмма загрузки в ЖКИ байта команды. Загружаемый байт должен быть
;предварительно помещен в регистр COM_IND_R (он же ADR_IND_R).
;COM_IND_R не портится.
;-------------------------------------------------------------------------
Pod_PER_COM_LCD:
          CLR         RS                     ;сброс RS - будет перед команда
          MOV         ACC,COM_IND_R          ;
          ACALL       Pod_TETR               ;
          ACALL       Pod_IMP_E              ;имп E - загр ст тетрады
          SWAP        A                      ;
          ACALL       Pod_TETR               ;
          ACALL       Pod_IMP_E              ;имп E - загр мл тетрады
          ACALL       Pod_OPROS_LCD          ;опрос ЖКИ
          RET

;-------------------------------------------------------------------------
;Подпрограмма загрузки в ЖКИ байта данных (индикация одного символа).
;Передаваемый байт должен быть предварительно помещен в регистр DATA_IND_R,
;адрес ОЗУ ЖКИ должен быть предварительно помещен в регистр ADR_IND_R
;(он же COM_IND_R). ADR_IND_R, DATA_IND_R не портятся.
;-------------------------------------------------------------------------
Pod_PER_DAT_LCD:
          CLR         RS                     ;сброс RS - будет перед команда
          MOV         ACC,ADR_IND_R          ;
          ACALL       Pod_TETR               ;
          SETB        PORT_IND_7             ;будет передаваться адрес ОЗУ ЖКИ
```

```
          NOP                           ;даем установиться данным
          ACALL     Pod_IMP_E           ;имп E - загр ст тетрады
          SWAP      A                   ;
          ACALL     Pod_TETR            ;
          ACALL     Pod_IMP_E           ;имп E - загр мл тетрады
          ACALL     Pod_OPROS_LCD       ;опрос ЖКИ
          MOV       ACC,DATA_IND_R      ;----------------------------
          CLR       C                   ;  блок определения,
          SUBB      A,#10               ;
          JNC       Aa_0                ;  не цифра ли будет
          MOV       ACC,DATA_IND_R      ;
          ADD       A,#30h              ;  индицироваться
          AJMP      Aa_10               ;
Aa_0:     MOV       ACC,DATA_IND_R      ;----------------------------
Aa_10:    SETB      RS                  ;уст RS - будут перед данные
          ACALL     Pod_TETR            ;
          ACALL     Pod_IMP_E           ;имп E - загр ст тетрады
          SWAP      A                   ;
          ACALL     Pod_TETR            ;
          ACALL     Pod_IMP_E           ;имп E - загр мл тетрады
          ACALL     Pod_OPROS_LCD       ;опрос ЖКИ
          RET

;-------------------------------------------------------------------------
;Подпрограмма опроса ЖКИ на предмет готовности принимать команды и данные.
;Подпрограмма возвращает управление, как только ЖКИ освободится.
;-------------------------------------------------------------------------
Pod_OPROS_LCD:
          CLR       RS                   ;сброс RS - будет читаться команда
          MOV       PORT_IND,#11110000b  ;сделать шину данных PORT_IND_4 -
                                         ;PORT_IND_7 входами
Aa_40:    SETB      RW                   ;уст режим чтения
          NOP                            ;даем установиться сигналам
          NOP                            ;
          SETB      E                    ;фронт импульса E
          NOP                            ;
          NOP                            ;
          JNB       PORT_IND_7,Aa_50     ;
          CLR       E                    ;спад импульса  E
          NOP                            ;
          NOP                            ;
          ACALL     Pod_IMP_E            ;имп E (продолжить чтение)
          AJMP      Aa_40                ;
Aa_50:    CLR       E                    ;спад импульса  E
          NOP                            ;
          NOP                            ;
          ACALL     Pod_IMP_E            ;имп E (закончить чтение)
          CLR       RW                   ;уст режим записи
          MOV       PORT_IND,#00000000b  ;сделать шину данных PORT_IND_4 -
                                         ;PORT_IND_7 выходами
          RET
```

3.3. Модуль основного АЦП

Теперь, когда все необходимые средства общения ADuC824 с пользователем определены и разработаны, можно, наконец, заняться исследованием главной «изюминки» микроконвертора – модуля 24-разрядного АЦП (ADC0). Прежде всего, представляется целесообразным проверить соответствие его заявленных характеристик реальным характеристикам. С этой целью вниманию читателей предлагается программа, исходный текст которой приведен в файле adc0.asm (листинг 3.6). Для проведения исследований необходимо на отладоч-

ной плате собрать макет, принципиальная схема которого изображена на рис. 3.2, соединив между собой отдельные узлы через разъемы проводниками. Проводники, соединяющие ИОН DA1 AD780 и прецизионный резистивный делитель R7, R8 с входами АЦП, должны быть минимальной длины. Разъемы на этих проводниках на схеме условно не показаны.

Листинг 3.6. Программа работы с АЦП0

```
;-------------------------------------------------------------------------------
;Демонстрационная программа использования модуля АЦП0 ADuC824.
;
;Данные в ЖКИ передаются по 4-битной шине, состояние ЖКИ опрашивается.
;
;Результат АЦП модуля АЦП0 индицируется на ЖКИ в двоичном виде начиная со СЗР
;старшего байта и заканчивая МЗР младшего байта (старш и средни байт в верхней
;строке, младш - в нижней строке).
;При нажатии на кнопку 0 производится внутренняя калибровка нуля.
;При этом по адресу 74 ОЗУ ЖКИ индицируется номер нажатой кнопки (0).
;При нажатии на кнопку 1 производится внутренняя калибровка верхнего предела.
;При этом по адресу 75 ОЗУ ЖКИ индицируется номер нажатой кнопки (1).
;При нажатии на кнопку 2 производится системная калибровка нуля.
;При этом по адресу 76 ОЗУ ЖКИ индицируется номер нажатой кнопки (2).
;При нажатии на кнопку 3 производится системная калибровка верхнего предела.
;При этом по адресу 77 ОЗУ ЖКИ индицируется номер нажатой кнопки (3).
;При нажатии на кнопку 4 производится переход в режим циклических преобразов.
;При этом по адресу 78 ОЗУ ЖКИ индицируется номер нажатой кнопки (4).
;При нажатии на кнопку 5 производится очистка ЖКИ.
;
;Используются прерывания от АЦП.
;-------------------------------------------------------------------------------
          $INCLUDE  (C:\ADuC\mod824)
          $INCLUDE  (C:\PR_ADUC\ADC0\824.inc)
;-------------------------------------------------------------------------------
;Описание битов, регистров и констант
;-------------------------------------------------------------------------------
;Порты и линии ввода-вывода
          PORT_KNOP    EQU      P0      ;порт кнопок

          PORT_IND     EQU      P2      ;порт индикации

          PORT_IND_0   EQU      P2_0    ;-----------------------------
          PORT_IND_1   EQU      P2_1    ;
          PORT_IND_2   EQU      P2_2    ; выходы индикации
          PORT_IND_3   EQU      P2_3    ;
          PORT_IND_4   EQU      P2_4    ;
          PORT_IND_5   EQU      P2_5    ;
          PORT_IND_6   EQU      P2_6    ;
          PORT_IND_7   EQU      P2_7    ;-----------------------------

          RW           EQU      PORT_IND_1 ;--------------------------
          RS           EQU      PORT_IND_2 ; линии управления ЖКИ
          E            EQU      PORT_IND_3 ;--------------------------

          _IN_KNOP0    EQU      P0_0    ;-----------------------------
          _IN_KNOP1    EQU      P0_1    ;
          _IN_KNOP2    EQU      P0_2    ;  входы кнопок
          _IN_KNOP3    EQU      P0_3    ;
          _IN_KNOP4    EQU      P0_4    ;
          _IN_KNOP5    EQU      P0_5    ;
          _IN_KNOP6    EQU      P0_6    ;
          _IN_KNOP7    EQU      P0_7    ;-----------------------------

;РОНы обслуживания ЖКИ и кнопок
          ADR_IND_R    DATA     030h    ;-----------------------------
          COM_IND_R    DATA     030h    ; РОНы обслуживания ЖКИ
          DATA_IND_R   DATA     031h    ;-----------------------------
```

```
          KNOP0_R      DATA     032h    ;----------------------------
          KNOP1_R      DATA     033h    ;
          KNOP2_R      DATA     034h    ; РОНы, содержащие
          KNOP3_R      DATA     035h    ;
          KNOP4_R      DATA     036h    ; флаги нажатия и удержания
          KNOP5_R      DATA     037h    ;
          KNOP6_R      DATA     038h    ; каждой кнопки
          KNOP7_R      DATA     039h    ;----------------------------

          NAKOPL0_R    DATA     03Ah    ;----------------------------
          NAKOPL1_R    DATA     03Bh    ;
          NAKOPL2_R    DATA     03Ch    ; РОНы накопления
          NAKOPL3_R    DATA     03Dh    ;
          NAKOPL4_R    DATA     03Eh    ; значения подавления дребезга
          NAKOPL5_R    DATA     03Fh    ;
          NAKOPL6_R    DATA     040h    ; каждой кнопки
          NAKOPL7_R    DATA     041h    ;----------------------------

           BYTE_R       DATA     04Fh    ;вспомогательный РОН для вывода
                                         ;байта на индикатор

          ADC0L_R      DATA     068h    ;-------------------------------------
          ADC0M_R      DATA     069h    ;РОНы хранения результ преобр АЦП0
          ADC0H_R      DATA     06Ah    ;-------------------------------------

;Константы
          NACH_ADR     EQU      000h    ;начальный адрес обнуления РОНов
          KON_ADR      EQU      07Fh    ;конечный адрес обнуления РОНов
          POROG_K      EQU      10      ;порог подавления дребезга кнопок

;Начало исполняемого кода-------------------------------------------------------
                ORG 0h
           AJMP       Lab_START            ;идти на начало осн программы

                ORG 033h
           AJMP       Lab_ADC0             ;идти на нач блока обр прер от АЦП

                     ORG 05Fh
;Начало блока обработки прерывания по окончании преобразования АЦП0
;Прерывания от АЦП и глобально должны быть разрешены.
Lab_ADC0:  CLR        EA                   ;---------------------------------
           PUSH       PSW                  ; глоб запрет прер и сохр контекста
           PUSH       ACC                  ;---------------------------------

           JB         RDY0,L_A0            ;уточняем источник прерывания
            AJMP       Lab_RETI             ;прерывание не от АЦП0, идти на вых

L_A0:      JNB        CAL,L_A1             ;уточняем, была калибр или измерение
           CLR        RDY0                 ;разрешаем дальнейшие преобразования
            AJMP       Lab_RETI             ;была калибровка, идти на выход

L_A1:      MOV        ADC0H_R,ADC0H        ;---------------------------------
            MOV        ADC0M_R,ADC0M        ; было измерение, копируем его рез
            MOV        ADC0L_R,ADC0L        ;---------------------------------

            CLR        RDY0                 ;разрешаем дальнейшие преобразования

;Блок возврата из прерываний---------------------------------------------------
Lab_RETI:  POP        ACC                  ;---------------------------------
           POP        PSW                  ; восст контекста и глоб разр прер
           SETB       EA                   ;---------------------------------
           RETI                            ;возврат из блока обраб прерываний

;Начало осн программы-----------------------------------------------------------
                ORG 100h
Lab_START: MOV        SP,#080h             ;определить указатель стека
           MOV        PLLCON,#00000000b    ;уст макс частоту ядра (12,58 МГц)
```

```
           NOP                          ;
           LCALL      Pod_INIT_RSN      ;иниц РСН
           LCALL      Pod_INIT_RON      ;иниц РОН

           LCALL      Pod_INIT_LCD      ;иниц ЖКИ

           LCALL      Pod_CLEAR_LCD     ;стирание ЖКИ

            SETB       EA               ;глоб разрешение прерываний
;Начало основного цикла-----------------------------------------------------
La_OSN:    NOP                          ;метка возврата в осн цикле

;Блок, предшествующий вызову подпрограммы опроса кнопки 0.
           MOV        R0,#KNOP0_R       ;
           MOV        R1,#NAKOPL0_R     ;
           LCALL      Pod_OPR_KNOP0     ;
;Блок, следующий за вызовом подпрограммы опроса кнопки (уст факта ее наж)
           MOV        ACC,@R0           ;
           JNB        ACC_1,La_1        ;

           CLR        ACC_1             ;кн была нажата, сброс флага нажат кн
           MOV        @R0,ACC           ;

            MOV        ADCMODE,#00100100b   ;АЦП0 вкл, реж внутр калибровки нуля

           MOV        DATA_IND_R,#0     ;-------------------------------------
           MOV        ADR_IND_R,#74     ; индикация  номера нажатой кнопки
           LCALL      Pod_PER_DAT_LCD   ;-------------------------------------

           LJMP       La_OSN            ;закрыть основной цикл

;Блок, предшествующий вызову подпрограммы опроса кнопки 1.
La_1:      MOV        R0,#KNOP1_R       ;
           MOV        R1,#NAKOPL1_R     ;
           LCALL      Pod_OPR_KNOP1     ;
;Блок, следующий за вызовом подпрограммы опроса кнопки (уст факта ее наж)
           MOV        ACC,@R0           ;
           JNB        ACC_1,La_2        ;

           CLR        ACC_1             ;кн была нажата, сброс флага нажат кн
           MOV        @R0,ACC           ;

          MOV      ADCMODE,#00100101b   ;АЦП0 вкл, реж внутр калибровки верхн пр

           MOV        DATA_IND_R,#1     ;-------------------------------------
           MOV        ADR_IND_R,#75     ; индикация  номера нажатой кнопки
           LCALL      Pod_PER_DAT_LCD   ;-------------------------------------

           LJMP       La_OSN            ;закрыть основной цикл

;Блок, предшествующий вызову подпрограммы опроса кнопки 2.
La_2:      MOV        R0,#KNOP2_R       ;
           MOV        R1,#NAKOPL2_R     ;
           LCALL      Pod_OPR_KNOP2     ;
;Блок, следующий за вызовом подпрограммы опроса кнопки (уст факта ее наж)
           MOV        ACC,@R0           ;
           JNB        ACC_1,La_3        ;

           CLR        ACC_1             ;кн была нажата, сброс флага нажат кн
           MOV        @R0,ACC           ;

            MOV        ADCMODE,#00100110b   ;АЦП0 вкл, реж сист калибровки нуля

           MOV        DATA_IND_R,#2     ;-------------------------------------
           MOV        ADR_IND_R,#76     ; индикация  номера нажатой кнопки
           LCALL      Pod_PER_DAT_LCD   ;-------------------------------------

           LJMP       La_OSN            ;закрыть основной цикл
```

```
;Блок, предшествующий вызову подпрограммы опроса кнопки 3.
La_3:       MOV         R0,#KNOP3_R          ;
            MOV         R1,#NAKOPL3_R        ;
            LCALL       Pod_OPR_KNOP3        ;
;Блок, следующий за вызовом подпрограммы опроса кнопки (уст факта ее наж)
            MOV         ACC,@R0              ;
            JNB         ACC_1,La_4           ;

            CLR         ACC_1                ;кн была нажата, сброс флага нажат кн
            MOV         @R0,ACC              ;

           MOV       ADCMODE,#00100111b   ;АЦП0 вкл, реж сист калибровки верхн пр

            MOV         DATA_IND_R,#3        ;-------------------------------------
            MOV         ADR_IND_R,#77        ; индикация  номера нажатой кнопки
            LCALL       Pod_PER_DAT_LCD      ;-------------------------------------

            LJMP        La_OSN               ;закрыть основной цикл

;Блок, предшествующий вызову подпрограммы опроса кнопки 4.
La_4:       MOV         R0,#KNOP4_R          ;
            MOV         R1,#NAKOPL4_R        ;
            LCALL       Pod_OPR_KNOP4        ;
;Блок, следующий за вызовом подпрограммы опроса кнопки (уст факта ее наж)
            MOV         ACC,@R0              ;
            JNB         ACC_1,La_5           ;

            CLR         ACC_1                ;кн была нажата, сброс флага нажат кн
            MOV         @R0,ACC              ;

            MOV         ADCMODE,#00100011b   ;АЦП0 вкл, реж циклич преобразований

            MOV         DATA_IND_R,#4        ;-------------------------------------
            MOV         ADR_IND_R,#78        ; индикация  номера нажатой кнопки
            LCALL       Pod_PER_DAT_LCD      ;-------------------------------------

            LJMP        La_OSN               ;закрыть основной цикл

;Блок, предшествующий вызову подпрограммы опроса кнопки 5.
La_5:       MOV         R0,#KNOP5_R          ;
            MOV         R1,#NAKOPL5_R        ;
            LCALL       Pod_OPR_KNOP5        ;
;Блок, следующий за вызовом подпрограммы опроса кнопки (уст факта ее наж)
            MOV         ACC,@R0              ;
            JNB         ACC_1,La_6           ;

            CLR         ACC_1                ;кн была нажата, сброс флага нажат кн
            MOV         @R0,ACC              ;

            LCALL       Pod_CLEAR_LCD        ;стирание ЖКИ

            LJMP        La_OSN               ;закрыть основной цикл

;Блок обработки результатов и вывода их на индикацию
La_6:       CLR         EA                   ;глоб запрещение прерываний

            MOV         BYTE_R,ADC0H_R       ;-------------------------------------
            MOV         R2,#0                ;  индикация ADC0H_R с адр 0 ОЗУ ЖКИ
            LCALL       Pod_IND_BYTE         ;-------------------------------------
            MOV         BYTE_R,ADC0M_R       ;-------------------------------------
            MOV         R2,#8                ;  индикация ADC0M_R с адр 8 ОЗУ ЖКИ
            LCALL       Pod_IND_BYTE         ;-------------------------------------
            MOV         BYTE_R,ADC0L_R       ;-------------------------------------
            MOV         R2,#64               ;  индикация ADC0L_R с адр 64 ОЗУ ЖКИ
            LCALL       Pod_IND_BYTE         ;-------------------------------------

            SETB        EA                   ;глоб разрешение прерываний
```

```
            LJMP       La_OSN               ;закрыть основной цикл

;Подпрограммы-----------------------------------------------------------------

;-------------------------------------------------------------------------------
;Подпрограмма инициализации РСН.
;-------------------------------------------------------------------------------
Pod_INIT_RSN:
          MOV        PORT_KNOP,#11111111b ;сделать вх все линии порта кн
          MOV        PORT_IND,#00000000b  ;сделать вых все линии порта индик
;Блок настройки АЦП
           MOV        ADCSTAT,#00000000b   ;сбросить все флаги АЦП
           MOV        ADCMODE,#00000000b   ;АЦП0, АЦП1 выкл, реж "снято питание"
           MOV        ADC0CON,#01001111b   ;АЦП0: внешн ИОН, входы AIN1-AIN2,
                                           ;униполярный режим, диапазон +/-2,56В
          MOV        SF,#045h             ;частота обновл выхода 20 Гц
          MOV        ICON,#00000000b      ;отключены все источники тока
           MOV        IE,#01000000b        ;разрешены только прерыв от АЦП
          RET

;-------------------------------------------------------------------------------
;Подпрограмма инициализации РОНов. Обнуляются все РОНы с адресами
;от NACH_ADR до KON_ADR при помощи косвенной адресации.
;-------------------------------------------------------------------------------
Pod_INIT_RON:
          MOV        R0,#NACH_ADR         ;установка начального адреса
Lk_0:     MOV        @R0,#0               ;обнуление очередного РОНа
          INC        R0                   ;переход к следующему адресу
          CJNE       R0,#KON_ADR,Lk_0     ;не достигли ли последнего адреса ?
          MOV        @R0,#0               ;обнуление последнего РОНа
          RET                             ;да, выход

;-------------------------------------------------------------------------
;Подпрограмма вывода на ЖКИ содержимого байта в двоичном коде (побитно) в
;виде восьми знакомест. Первым выводится СЗР байта. Байт предварительно
;должен быть помещен в РОН BYTE_R. В R2 предварительно следует поместить
;адрес ОЗУ ЖКИ, с которого начнется вывод на индикацию.
;-------------------------------------------------------------------------
Pod_IND_BYTE:
           MOV       R1,#0                ;нач уст счетчика битов в байте
```

```
Lj_0:      MOV       ACC,BYTE_R           ;копируем байт в акк
           JB        ACC_7,Lj_1           ;
           MOV       DATA_IND_R,#0        ;
           AJMP      Lj_2                 ;
Lj_1:       MOV       DATA_IND_R,#1        ;
Lj_2:      MOV       ADR_IND_R,R2         ;
           RL        A                    ;
            MOV       BYTE_R,ACC           ;сдвиг байта влево циклический
           ACALL     Pod_PER_DAT_LCD      ;
           INC       R1                   ;
           INC       R2                   ;
            CJNE      R1,#8,Lj_0           ;
           RET

;Подключение модулей опроса кнопок и вывода данных на ЖКИ (с опросом ЖКИ)
          $INCLUDE   (C:\PR_ADUC\ADC0\knop.asm)
          $INCLUDE   (C:\PR_ADUC\ADC0\lcd_opr.asm)

;Конец исполняемого кода
          END
```

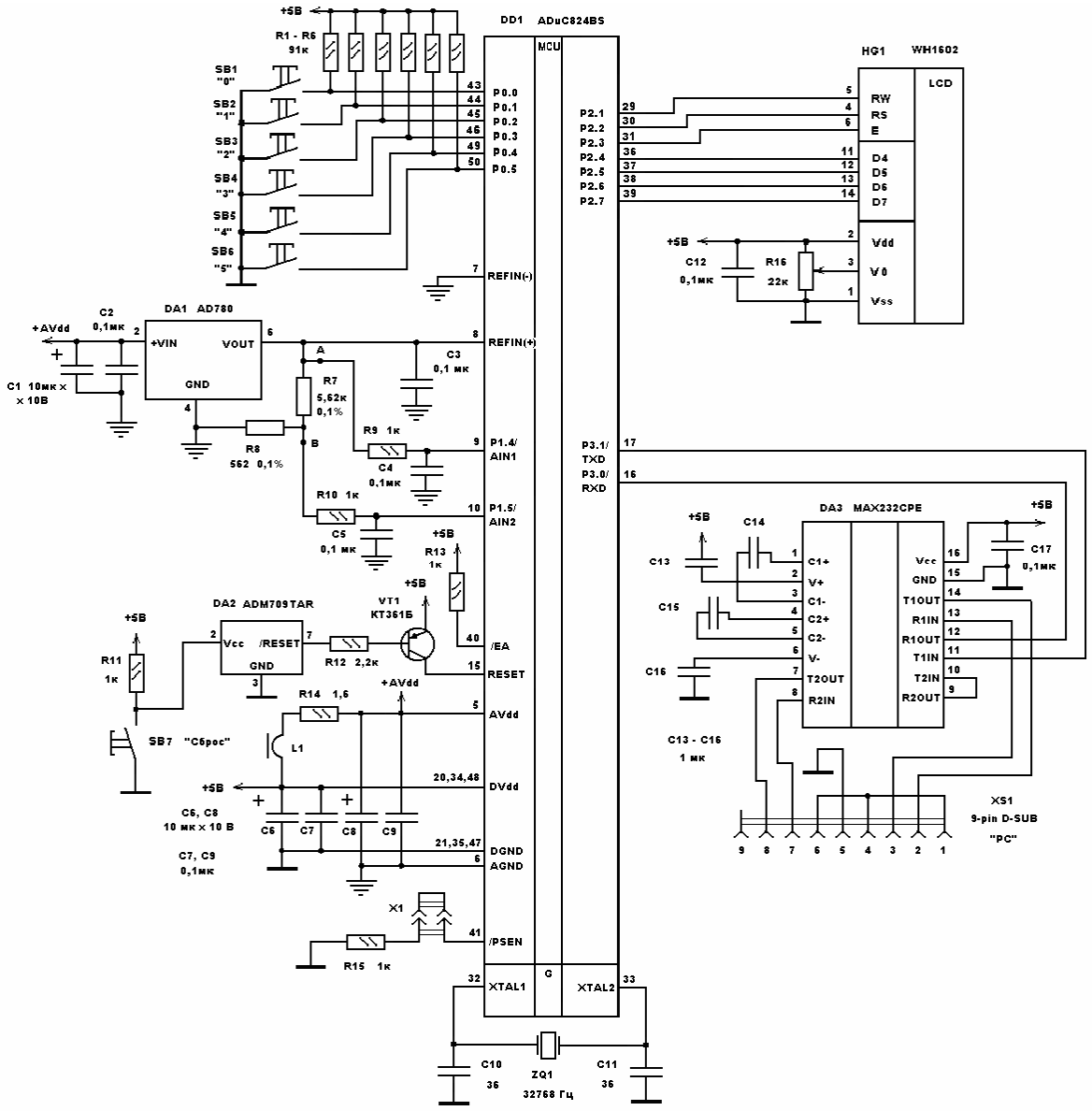


Рис. 3.2. Схема макета для исследования АЦП0

Программа позволяет производить циклические преобразования (измерения) и все виды калибровок для модуля ADC0 в выбранном канале (AIN1-AIN2), в выбранных диапазоне (±2,56В), режиме (униполярном) и частоте обновления выходных данных (19,79 Гц) с использованием в качестве ИОН АЦП внешнего источника (DA1). Все приведенные параметры настройки модуля ADC0 могут быть легко изменены путем коррекции байтовых констант в подпрограмме Pod_INIT_RSN. Результат каждого преобразования индицируется на ЖКИ в двоичном коде в виде 24-разрядного слова данных начиная со СЗР старшего байта (адрес 0 ОЗУ ЖКИ) и заканчивая МЗР младшего байта (адрес 71 ОЗУ ЖКИ). Обновление показаний ЖКИ производится с выбранной частотой обновления данных на выходе АЦП. При нажатии на кнопку «0» производится внутренняя калибровка «нуля», при этом по адресу 74 ОЗУ ЖКИ индицируется номер нажатой кнопки («0»). При нажатии на кнопку «1» производится внутренняя калибровка верхнего предела шкалы, при этом по адресу 75 ОЗУ

ЖКИ индицируется номер нажатой кнопки («1»). При нажатии на кнопку «2» производится системная калибровка «нуля», при этом по адресу 76 ОЗУ ЖКИ индицируется номер нажатой кнопки («2»). Перед проведением системной калибровки «нуля» на входы выбранного канала необходимо подать системное напряжение «нуля». При нажатии на кнопку «3» производится системная калибровка верхнего предела шкалы, при этом по адресу 77 ОЗУ ЖКИ индицируется номер нажатой кнопки («3»). Перед проведением системной калибровки верхнего предела шкалы на входы выбранного канала необходимо подать системное напряжение верхнего предела шкалы. После проведения любой из калибровок ADC0 переходит в режим «снятое питание» и процесс преобразований (и обновление показаний ЖКИ) приостанавливается. Запустить процесс циклических преобразований можно путем нажатия на кнопку «4», при этом по адресу 78 ОЗУ ЖКИ индицируется номер нажатой кнопки («4»). При нажатии на кнопку «5» производится очистка ЖКИ.

После включения питания ЖКИ будет отображать 24-разрядное слово, состоящее из одних нулей. Запустим преобразования, нажав на кнопку «4». На ЖКИ отобразится некоторое ненулевое 24-разрядное слово, значение которого будет соответствовать входному измеряемому напряжению. Младшие 8–9 разрядов будут меняться от преобразования к преобразованию, ограничивая, таким образом, реальное разрешение АЦП 15–16 разрядами. Проведение внутренних калибровок «нуля» и верхнего предела шкалы изменит абсолютное значение результата измерений (приведет результат к выбранным пределам), но не улучшит реального разрешения – 8–9 младших разрядов будут по-прежнему меняться. Перед проведением системной калибровки «нуля» разорвем проводники в точках А и В (между резисторами R9, R10 и резистором R7, на котором измеряется напряжение) и соединим друг с другом проводники, идущие от оставшихся свободными выводов R9, R10, подав, таким образом, на входы AIN1–AIN2 напряжение системного «нуля». Запустив после этого процесс циклических преобразований, можно наблюдать на экране ЖКИ 24-разрядное слово данных, соответствующее напряжению системного «нуля». Старшие 16 разрядов в нем нулевые, а младшие 8 меняются от преобразования к преобразованию. Теперь нажмем на кнопку «2», запустив системную калибровку «нуля», а затем снова запустим процесс циклических преобразований. После этого меняться будут только 6 младших разрядов, что дает возможность количественно оценить результат проведения калибровки. Если считать, что меняющиеся младшие разряды отражают влияние собственных шумов АЦП, то число неизменных нулевых старших разрядов как раз и будет являться реальным разрешением данного АЦП. Другими словами, АЦП интерпретирует внешнее короткое замыкание своих дифференциальных входов как нулевое входное напряжение с точностью до 6 младших разрядов. Это соответствует заявленному производителем в спецификации на ADuC824 значению реального разрешения ADC0, приведенному в табл. 1.1 (18 разрядов при частоте преобразования 20 Гц и диапазоне ±2,56 В).

Затем подадим на входы АЦП напряжение с резистора R7, восстановив разорванные ранее соединения. В 24-разрядном слове результата теперь будут меняться только 7 младших разрядов, что дает значение реального разрешения 17 разрядов. Очевидно, в данном случае расхождение между заявленным и реальным разрешением имеет место из-за собственных шумов AD780 и резисторов R7, R8 и (или) внешних наводок на измерительные цепи. Возможно, результаты окажутся лучше, если измерительные цепи и цепи опорного напряжения экранировать, хотя в спецификации на микроконвертор о необходимости такой экранировки ничего не сказано.

Для того, чтобы установить в программе adc0.asm диапазон входных сигналов модуля АЦП0 равным ±20 мВ, следует в подпрограмме Pod_INIT_RSN поместить в специальный регистр ADC0CON байтовую константу #01001000b. После трансляции получившегося исходного текста и загрузки сгенерированного файла adc0.hex в ADuC824 произведем оценку реального разрешения АЦП для диапазона ±20 мВ, проделав часть описанной выше последовательности операций. Можно видеть, что до проведения системной калибровки «нуля» при наличии замыкания (напряжения системного «нуля») на входе в показаниях ЖКИ будут оставаться неизменными и нулевыми 10-11 старших разрядов результата. После проведения системной калибровки «нуля» неизменными и нулевыми окажутся 12–13 старших разрядов. Как и в предыдущем случае, полученный результат вполне соответствует заявленному производителем в спецификации на ADuC824 значению реального разрешения ADC0, приведенному в табл. 1.1 (13 разрядов при частоте преобразования 20 Гц и диапазоне ±20 мВ).

Программа, исходный текст которой приведен в файле adc0_a.asm (листинг 3.7), функционально идентична предыдущему примеру, но производит вывод результата преобразований на индикацию в десятичном коде (в виде десятизначного числа). С помощью программы adc0_a.asm, располагая цифровым вольтметром с достаточной точностью и разрешением, можно количественно оценить такие параметры ADC0, как интегральную нелинейность и ошибку смещения. (Для этой цели вполне годится и программа из предыдущего примера, но десятичными числами при ручных расчетах оперировать удобнее, чем двоичными). Считывание результатов преобразований с ЖКИ следует производить после проведения внутренних калибровок «нуля» и верхнего предела шкалы и системной калибровки «нуля», как это описано в предыдущем примере. Для достижения наилучшей возможной точности преобразования частота обновления данных на выходе модуля АЦП выбрана минимальной (5 Гц). Для оценки интегральной нелинейности и смещения надо считать показания ЖКИ для нескольких значений входного напряжения, одновременно измерив вольтметром эти значения, а затем по полученным точкам построить график зависимости результата преобразования в дискретах от входного напряжения в вольтах. Величина отклонения получившейся кривой от прямой линии, проведенной из точки с координатами (0,0) в точку с координатами (2,56В, 224 = 16 777 216 дискрет) для каждой точки кривой, измеренная в дискретах, является зна-

чением интегральной нелинейности для этой точки. Необходимо отметить, что описанная методика вычисления интегральной нелинейности является лишь одной из нескольких возможных [12], причем в спецификации производителя на ADuC824 (табл. 1.1) не указано, по какой именно методике вычислялось приведенное там значение интегральной нелинейности, как не указан и конкретный метод линеаризации характеристики преобразования при вычислении приведенного в таблице значения ошибки смещения. В спецификации только указано, что после проведения системной калибровки «нуля» ошибка смещения будет полностью скомпенсирована, т. е. станет равной нулю.

Листинг 3.7. Использование АЦП0 с десятичной индикацией

```
;-------------------------------------------------------------------------------
;Демонстрационная программа использования модуля АЦП0 ADuC824.
;
;Данные в ЖКИ передаются по 4-битной шине, состояние ЖКИ опрашивается.
;
;Результат АЦП модуля АЦП0 индицируется на ЖКИ в десятичном виде.
;При нажатии на кнопку 0 производится внутренняя калибровка нуля.
;При этом по адресу 74 ОЗУ ЖКИ индицируется номер нажатой кнопки (0).
;При нажатии на кнопку 1 производится внутренняя калибровка верхнего предела.
;При этом по адресу 75 ОЗУ ЖКИ индицируется номер нажатой кнопки (1).
;При нажатии на кнопку 2 производится системная калибровка нуля.
;При этом по адресу 76 ОЗУ ЖКИ индицируется номер нажатой кнопки (2).
;При нажатии на кнопку 3 производится системная калибровка верхнего предела.
;При этом по адресу 77 ОЗУ ЖКИ индицируется номер нажатой кнопки (3).
;При нажатии на кнопку 4 производится переход в режим циклических преобразов.
;При этом по адресу 78 ОЗУ ЖКИ индицируется номер нажатой кнопки (4).
;При нажатии на кнопку 5 производится очистка ЖКИ.
;
;Используются прерывания от АЦП.
;-------------------------------------------------------------------------------
          $INCLUDE  (C:\ADuC\mod824)
          $INCLUDE  (C:\PR_ADUC\ADC0_a\824.inc)
;-------------------------------------------------------------------------------
;Описание битов, регистров и констант
;-------------------------------------------------------------------------------
;Порты и линии ввода-вывода
          PORT_KNOP    EQU      P0      ;порт кнопок

          PORT_IND     EQU      P2      ;порт индикации

          PORT_IND_0   EQU      P2_0    ;-----------------------------
          PORT_IND_1   EQU      P2_1    ;
          PORT_IND_2   EQU      P2_2    ; выходы индикации
          PORT_IND_3   EQU      P2_3    ;
          PORT_IND_4   EQU      P2_4    ;
          PORT_IND_5   EQU      P2_5    ;
          PORT_IND_6   EQU      P2_6    ;
          PORT_IND_7   EQU      P2_7    ;-----------------------------

          RW           EQU      PORT_IND_1 ;--------------------------
          RS           EQU      PORT_IND_2 ; линии управления ЖКИ
          E            EQU      PORT_IND_3 ;--------------------------

          _IN_KNOP0    EQU      P0_0    ;-----------------------------
          _IN_KNOP1    EQU      P0_1    ;
          _IN_KNOP2    EQU      P0_2    ;  входы кнопок
          _IN_KNOP3    EQU      P0_3    ;
          _IN_KNOP4    EQU      P0_4    ;
          _IN_KNOP5    EQU      P0_5    ;
          _IN_KNOP6    EQU      P0_6    ;
          _IN_KNOP7    EQU      P0_7    ;-----------------------------
```

```
;РОНы обслуживания ЖКИ и кнопок
          ADR_IND_R    DATA     030h    ;----------------------------
          COM_IND_R    DATA     030h    ; РОНы обслуживания ЖКИ
          DATA_IND_R   DATA     031h    ;----------------------------

          KNOP0_R      DATA     032h    ;----------------------------
          KNOP1_R      DATA     033h    ;
          KNOP2_R      DATA     034h    ; РОНы, содержащие
          KNOP3_R      DATA     035h    ;
          KNOP4_R      DATA     036h    ; флаги нажатия и удержания
          KNOP5_R      DATA     037h    ;
          KNOP6_R      DATA     038h    ; каждой кнопки
          KNOP7_R      DATA     039h    ;----------------------------

          NAKOPL0_R    DATA     03Ah    ;----------------------------
          NAKOPL1_R    DATA     03Bh    ;
          NAKOPL2_R    DATA     03Ch    ; РОНы накопления
          NAKOPL3_R    DATA     03Dh    ;
          NAKOPL4_R    DATA     03Eh    ; значения подавления дребезга
          NAKOPL5_R    DATA     03Fh    ;
          NAKOPL6_R    DATA     040h    ; каждой кнопки
          NAKOPL7_R    DATA     041h    ;----------------------------

;РОНы обслуживания подпрограмм преобразования формы представления чисел
          IND_MILL_R   DATA     04Eh    ;РОН миллиардов дес числа
          IND_SOTMIL_R DATA     04Fh    ;РОН сотен миллионов дес числа
          IND_DESMIL_R DATA     050h    ;РОН десятков миллионов дес числа
          IND_MIL_R    DATA     051h    ;РОН миллионов дес числа
          IND_SOTTIS_R DATA     052h    ;РОН сотен тысяч дес числа
          IND_DESTIS_R DATA     053h    ;РОН десятков тысяч дес числа
          IND_TIS_R    DATA     054h    ;РОН тысяч дес числа
          IND_SOT_R    DATA     055h    ;РОН сотен дес числа
          IND_DES_R    DATA     056h    ;РОН десятков дес числа
          IND_ED_R     DATA     057h    ;РОН единиц дес числа

          BYTE_0_R     DATA     05Bh    ;байт 0 упакованного двоичн-дес числа
          BYTE_1_R     DATA     05Ch    ;байт 1 упакованного двоичн-дес числа
          BYTE_2_R     DATA     05Dh    ;байт 2 упакованного двоичн-дес числа
          BYTE_3_R     DATA     05Eh    ;байт 3 упакованного двоичн-дес числа
          BYTE_4_R     DATA     05Fh    ;байт 4 упакованного двоичн-дес числа

          ADC0L_R      DATA     068h    ;-----------------------------------
          ADC0M_R      DATA     069h    ;РОНы хранения результата преобр АЦП0
          ADC0H_R      DATA     06Ah    ;-----------------------------------

;Константы
          NACH_ADR     EQU      000h    ;начальный адрес обнуления РОНов
          KON_ADR      EQU      07Fh    ;конечный адрес обнуления РОНов
          POROG_K      EQU      10      ;порог подавления дребезга кнопок

;Начало исполняемого кода-------------------------------------------------------
               ORG 0h
           AJMP       Lab_START           ;идти на начало осн программы

               ORG 033h
           AJMP       Lab_ADC0            ;идти на нач блока обр прер от АЦП

                    ORG 05Fh
;Начало блока обработки прерывания по окончании преобразования АЦП0
;Прерывания от АЦП и глобально должны быть разрешены.
Lab_ADC0:  CLR        EA                  ;-----------------------------------
           PUSH       PSW                 ; глоб запрет прер и сохр контекста
           PUSH       ACC                 ;-----------------------------------

           JB         RDY0,L_A0           ;уточняем источник прерывания
            AJMP       Lab_RETI            ;прерывание не от АЦП0, идти на вых

L_A0:      JNB        CAL,L_A1            ;уточняем, была калибр или измерение
```

```
            CLR         RDY0                ;разрешаем дальнейшие преобразования
             AJMP        Lab_RETI            ;была калибровка, идти на выход

L_A1:       MOV         ADC0H_R,ADC0H       ;--------------------------------
             MOV         ADC0M_R,ADC0M       ; было измерение, копируем его рез
             MOV         ADC0L_R,ADC0L       ;--------------------------------

             CLR         RDY0                ;разрешаем дальнейшие преобразования

;Блок возврата из прерываний-----------------------------------------------------
Lab_RETI:   POP         ACC                 ;--------------------------------
            POP         PSW                 ; восст контекста и глоб разр прер
            SETB        EA                  ;--------------------------------
            RETI                            ;возврат из блока обраб прерываний

;Начало осн программы------------------------------------------------------------
                 ORG 100h
Lab_START:  MOV         SP,#080h            ;определить указатель стека
            MOV         PLLCON,#00000000b   ;уст макс частоту ядра (12,58 МГц)
            NOP                             ;
            LCALL       Pod_INIT_RSN        ;иниц РСН
            LCALL       Pod_INIT_RON        ;иниц РОН

            LCALL       Pod_INIT_LCD        ;иниц ЖКИ

            LCALL       Pod_CLEAR_LCD       ;стирание ЖКИ

             SETB        EA                  ;глоб разрешение прерываний
;Начало основного цикла----------------------------------------------------------
La_OSN:     NOP                             ;метка возврата в осн цикле

;Блок, предшествующий вызову подпрограммы опроса кнопки 0.
            MOV         R0,#KNOP0_R         ;
            MOV         R1,#NAKOPL0_R       ;
            LCALL       Pod_OPR_KNOP0       ;
;Блок, следующий за вызовом подпрограммы опроса кнопки (уст факта ее наж)
            MOV         ACC,@R0             ;
            JNB         ACC_1,La_1          ;

            CLR         ACC_1               ;кн была нажата, сброс флага нажат кн
            MOV         @R0,ACC             ;

             MOV         ADCMODE,#00100100b   ;АЦП0 вкл, реж внутр калибровки нуля

            MOV         DATA_IND_R,#0       ;-------------------------------------
            MOV         ADR_IND_R,#74       ; индикация  номера нажатой кнопки
            LCALL       Pod_PER_DAT_LCD     ;-------------------------------------

            LJMP        La_OSN              ;закрыть основной цикл

;Блок, предшествующий вызову подпрограммы опроса кнопки 1.
La_1:       MOV         R0,#KNOP1_R         ;
            MOV         R1,#NAKOPL1_R       ;
            LCALL       Pod_OPR_KNOP1       ;
;Блок, следующий за вызовом подпрограммы опроса кнопки (уст факта ее наж)
            MOV         ACC,@R0             ;
            JNB         ACC_1,La_2          ;

            CLR         ACC_1               ;кн была нажата, сброс флага нажат кн
            MOV         @R0,ACC             ;

           MOV      ADCMODE,#00100101b   ;АЦП0 вкл, реж внутр калибровки верхн пр

            MOV         DATA_IND_R,#1       ;-------------------------------------
            MOV         ADR_IND_R,#75       ; индикация  номера нажатой кнопки
            LCALL       Pod_PER_DAT_LCD     ;-------------------------------------

            LJMP        La_OSN              ;закрыть основной цикл
```

```
;Блок, предшествующий вызову подпрограммы опроса кнопки 2.
La_2:       MOV         R0,#KNOP2_R          ;
            MOV         R1,#NAKOPL2_R        ;
            LCALL       Pod_OPR_KNOP2        ;
;Блок, следующий за вызовом подпрограммы опроса кнопки (уст факта ее наж)
            MOV         ACC,@R0              ;
            JNB         ACC_1,La_3           ;

            CLR         ACC_1                ;кн была нажата, сброс флага нажат кн
            MOV         @R0,ACC              ;

             MOV         ADCMODE,#00100110b   ;АЦП0 вкл, реж сист калибровки нуля

            MOV         DATA_IND_R,#2        ;-------------------------------------
            MOV         ADR_IND_R,#76        ; индикация  номера нажатой кнопки
            LCALL       Pod_PER_DAT_LCD      ;-------------------------------------

            LJMP        La_OSN               ;закрыть основной цикл

;Блок, предшествующий вызову подпрограммы опроса кнопки 3.
La_3:       MOV         R0,#KNOP3_R          ;
            MOV         R1,#NAKOPL3_R        ;
            LCALL       Pod_OPR_KNOP3        ;
;Блок, следующий за вызовом подпрограммы опроса кнопки (уст факта ее наж)
            MOV         ACC,@R0              ;
            JNB         ACC_1,La_4           ;

            CLR         ACC_1                ;кн была нажата, сброс флага нажат кн
            MOV         @R0,ACC              ;

           MOV        ADCMODE,#00100111b   ;АЦП0 вкл, реж сист калибровки верхн пр

            MOV         DATA_IND_R,#3        ;-------------------------------------
            MOV         ADR_IND_R,#77        ; индикация  номера нажатой кнопки
            LCALL       Pod_PER_DAT_LCD      ;-------------------------------------

            LJMP        La_OSN               ;закрыть основной цикл

;Блок, предшествующий вызову подпрограммы опроса кнопки 4.
La_4:       MOV         R0,#KNOP4_R          ;
            MOV         R1,#NAKOPL4_R        ;
            LCALL       Pod_OPR_KNOP4        ;
;Блок, следующий за вызовом подпрограммы опроса кнопки (уст факта ее наж)
            MOV         ACC,@R0              ;
            JNB         ACC_1,La_5           ;

            CLR         ACC_1                ;кн была нажата, сброс флага нажат кн
            MOV         @R0,ACC              ;

             MOV         ADCMODE,#00100011b   ;АЦП0 вкл, реж циклич преобразований

             MOV         DATA_IND_R,#4        ;-------------------------------------
            MOV         ADR_IND_R,#78        ; индикация  номера нажатой кнопки
            LCALL       Pod_PER_DAT_LCD      ;-------------------------------------

            LJMP        La_OSN               ;закрыть основной цикл

;Блок, предшествующий вызову подпрограммы опроса кнопки 5.
La_5:       MOV         R0,#KNOP5_R          ;
            MOV         R1,#NAKOPL5_R        ;
            LCALL       Pod_OPR_KNOP5        ;
;Блок, следующий за вызовом подпрограммы опроса кнопки (уст факта ее наж)
            MOV         ACC,@R0              ;
            JNB         ACC_1,La_6           ;

            CLR         ACC_1                ;кн была нажата, сброс флага нажат кн
            MOV         @R0,ACC              ;
```

```
            LCALL       Pod_CLEAR_LCD        ;стирание ЖКИ

            LJMP        La_OSN               ;закрыть основной цикл

;Блок обработки результатов и вывода их на индикацию
La_6:       CLR         EA                   ;глоб запрещение прерываний

            MOV         R0,#BYTE_0_R         ;указание на РОНы для преобразования
            MOV         R1,#ADC0L_R          ;
            MOV         ADC0H_R+1,#0         ;обнуление ст незначащего РОНа
            LCALL       B32BCD               ;преобр числа дискр из дв в дв-дес

            MOV         R0,#BYTE_0_R         ;указание на РОНы для преобразования
            MOV         R1,#IND_MILL_R       ;
            LCALL       BCD10BCD            ;пр числа дискр из уп дв-дес в неуп дес

            MOV         R0,#IND_MILL_R       ; индицировать с РОНа IND_MILL_R
            MOV         R1,#0                ; индицировать с адр 0 ЖКИ
            LCALL       Pod_IND_10ZN         ;индикация числа дискрет

            SETB        EA                   ;глоб разрешение прерываний

            LJMP        La_OSN               ;закрыть основной цикл

;Подпрограммы-----------------------------------------------------------------

;-------------------------------------------------------------------------------
;Подпрограмма инициализации РСН.
;-------------------------------------------------------------------------------
Pod_INIT_RSN:
           MOV         PORT_KNOP,#11111111b ;сделать вх все линии порта кн
           MOV         PORT_IND,#00000000b  ;сделать вых все линии порта индик
;Блок настройки АЦП
            MOV         ADCSTAT,#00000000b   ;сбросить все флаги АЦП
           MOV         ADCMODE,#00000000b   ;АЦП0, АЦП1 выкл, реж "снято питание"
            MOV         ADC0CON,#01001111b   ;АЦП0: внешн ИОН, входы AIN1-AIN2,
                                             ;униполярный режим, диапазон +/-2,56В
           MOV         SF,#0FFh             ;частота обновл выхода 5 Гц
            MOV         ICON,#00000000b      ;отключены все источники тока
            MOV         IE,#01000000b        ;разрешены только прерыв от АЦП
           RET

;-------------------------------------------------------------------------------
;Подпрограмма инициализации РОНов. Обнуляются все РОНы с адресами
;от NACH_ADR до KON_ADR при помощи косвенной адресации.
;-------------------------------------------------------------------------------
Pod_INIT_RON:
           MOV         R0,#NACH_ADR         ;установка начального адреса
Lk_0:      MOV         @R0,#0               ;обнуление очередного РОНа
           INC         R0                   ;переход к следующему адресу
           CJNE        R0,#KON_ADR,Lk_0     ;не достигли ли последнего адреса ?
           MOV         @R0,#0               ;обнуление последнего РОНа
           RET                              ;да, выход

;-------------------------------------------------------------------------------
;Подпрограмма вывода на ЖКИ содержимого буфера из 10 РОНов в виде 10 знакомест.
;R0 должен указывать на РОН в буфере, содержимое которого индицируется первым.
;R1 должен содержать нач адрес знакоместа в ОЗУ ЖКИ.
;-------------------------------------------------------------------------------
Pod_IND_10ZN:
            MOV         R2,#0                ;нач уст ст цикла
Ii_0:       MOV         DATA_IND_R,@R0       ;
            MOV         ADR_IND_R,R1         ;
            LCALL       Pod_PER_DAT_LCD      ; индикация очередного символа
            INC         R0                   ;
            INC         R1                   ;
            INC         R2                   ;
```

```
            CJNE        R2,#10,Ii_0          ;
            RET

;Подключение модулей опроса кнопок, вывода данных на ЖКИ (с опросом ЖКИ)
;и преобразования представлений чисел
           $INCLUDE   (C:\PR_ADUC\ADC0_a\knop.asm)
           $INCLUDE   (C:\PR_ADUC\ADC0_a\lcd_opr.asm)
            $INCLUDE   (C:\PR_ADUC\ADC0_a\preobr.asm)

;Конец исполняемого кода
           END
```

Экстраполировав построенную по точкам кривую до пересечения с осями, получим реальное значение ошибки смещения АЦП, выраженное в дискретах и в вольтах.

Несколько значений входного напряжения АЦП (точек измерения) можно получить, заменив постоянный резистор R7 в схеме на рис. 3.2 на многооборотный переменный, причем, напряжение, подаваемое на RC-фильтр R9C4 входа AIN1, следует снимать с подвижного контакта этого резистора. Измерение напряжения необходимо производить цифровым вольтметром, имеющим разрешение не менее семи десятичных разрядов после запятой на пределе «единицы вольт» и погрешность на этом пределе не хуже одной-двух единиц младшего разряда, так как требуется измерять напряжение с точностью до (2,56 В/ 2^{18}) = 0,0000097 В.

3.4. Модуль дополнительного АЦП

Для проведения оценочных исследований модуля дополнительного АЦП вниманию читателей предлагается программа, исходный текст которой приведен в файле adc1.asm (листинг 3.8). При проведении исследований можно использовать макет, собранный для программы из предыдущего примера, однако, необходимо внести в него некоторые изменения, показанные на фрагменте принципиальной схемы, приведенном на рис. 3.3. Отличия заключаются в том, что входное напряжение здесь подается на несимметричный аналоговый вход AIN3 относительно общего провода (аналоговой «земли»).

Листинг 3.8. Использование дополнительного АЦП

```
;-------------------------------------------------------------------------
;Демонстрационная программа использования модуля АЦП1 ADuC824.
;
;Данные в ЖКИ передаются по 4-битной шине, состояние ЖКИ опрашивается.
;
;Результат АЦП модуля АЦП1 индицируется на ЖКИ в двоичном виде начиная со СЗР
;старшего байта и заканчивая МЗР младшего байта (в верхней строке).
;При нажатии на кнопку 0 производится внутренняя калибровка нуля.
;При этом по адресу 74 ОЗУ ЖКИ индицируется номер нажатой кнопки (0).
;При нажатии на кнопку 1 производится внутренняя калибровка верхнего предела.
;При этом по адресу 75 ОЗУ ЖКИ индицируется номер нажатой кнопки (1).
;При нажатии на кнопку 2 производится системная калибровка нуля.
;При этом по адресу 76 ОЗУ ЖКИ индицируется номер нажатой кнопки (2).
;При нажатии на кнопку 3 производится системная калибровка верхнего предела.
;При этом по адресу 77 ОЗУ ЖКИ индицируется номер нажатой кнопки (3).
;При нажатии на кнопку 4 производится переход в режим циклических преобразов.
;При этом по адресу 78 ОЗУ ЖКИ индицируется номер нажатой кнопки (4).
;При нажатии на кнопку 5 производится очистка ЖКИ.
```

```
;
;Используются прерывания от АЦП.
;-------------------------------------------------------------------------------
          $INCLUDE  (C:\ADuC\mod824)
          $INCLUDE  (C:\PR_ADUC\ADC1\824.inc)
;-------------------------------------------------------------------------------
;Описание битов, регистров и констант
;-------------------------------------------------------------------------------
;Порты и линии ввода-вывода
          PORT_KNOP    EQU      P0      ;порт кнопок

          PORT_IND     EQU      P2      ;порт индикации

          PORT_IND_0   EQU      P2_0    ;-----------------------------
          PORT_IND_1   EQU      P2_1    ;
          PORT_IND_2   EQU      P2_2    ; выходы индикации
          PORT_IND_3   EQU      P2_3    ;
          PORT_IND_4   EQU      P2_4    ;
          PORT_IND_5   EQU      P2_5    ;
          PORT_IND_6   EQU      P2_6    ;
          PORT_IND_7   EQU      P2_7    ;-----------------------------

          RW           EQU      PORT_IND_1 ;--------------------------
          RS           EQU      PORT_IND_2 ; линии управления ЖКИ
          E            EQU      PORT_IND_3 ;--------------------------

          _IN_KNOP0    EQU      P0_0    ;-----------------------------
          _IN_KNOP1    EQU      P0_1    ;
          _IN_KNOP2    EQU      P0_2    ;  входы кнопок
          _IN_KNOP3    EQU      P0_3    ;
          _IN_KNOP4    EQU      P0_4    ;
          _IN_KNOP5    EQU      P0_5    ;
          _IN_KNOP6    EQU      P0_6    ;
          _IN_KNOP7    EQU      P0_7    ;-----------------------------

;РОНы обслуживания ЖКИ и кнопок
          ADR_IND_R    DATA     030h    ;-----------------------------
          COM_IND_R    DATA     030h    ; РОНы обслуживания ЖКИ
          DATA_IND_R   DATA     031h    ;-----------------------------

          KNOP0_R      DATA     032h    ;-----------------------------
          KNOP1_R      DATA     033h    ;
          KNOP2_R      DATA     034h    ; РОНы, содержащие
          KNOP3_R      DATA     035h    ;
          KNOP4_R      DATA     036h    ; флаги нажатия и удержания
          KNOP5_R      DATA     037h    ;
          KNOP6_R      DATA     038h    ; каждой кнопки
          KNOP7_R      DATA     039h    ;-----------------------------

          NAKOPL0_R    DATA     03Ah    ;-----------------------------
          NAKOPL1_R    DATA     03Bh    ;
          NAKOPL2_R    DATA     03Ch    ; РОНы накопления
          NAKOPL3_R    DATA     03Dh    ;
          NAKOPL4_R    DATA     03Eh    ; значения подавления дребезга
          NAKOPL5_R    DATA     03Fh    ;
          NAKOPL6_R    DATA     040h    ; каждой кнопки
          NAKOPL7_R    DATA     041h    ;-----------------------------

           BYTE_R       DATA     04Fh    ;вспомогательный РОН для вывода
                                         ;байта на индикатор

          ADC1L_R      DATA     068h    ;-------------------------------------
          ADC1H_R      DATA     069h    ;РОНы хранения результата преобр АЦП1

;Константы
          NACH_ADR     EQU      000h    ;начальный адрес обнуления РОНов
          KON_ADR      EQU      07Fh    ;конечный адрес обнуления РОНов
          POROG_K      EQU      10      ;порог подавления дребезга кнопок
```

```
;Начало исполняемого кода-------------------------------------------------
                ORG 0h
           AJMP       Lab_START            ;идти на начало осн программы

                ORG 033h
           AJMP       Lab_ADC1             ;идти на нач блока обр прер от АЦП

                     ORG 05Fh
;Начало блока обработки прерывания по окончании преобразования АЦП1
;Прерывания от АЦП и глобально должны быть разрешены.
Lab_ADC1:  CLR        EA                   ;---------------------------------
           PUSH       PSW                  ; глоб запрет прер и сохр контекста
           PUSH       ACC                  ;---------------------------------

           JB         RDY1,L_A0            ;уточняем источник прерывания
            AJMP       Lab_RETI             ;прерывание не от АЦП1, идти на вых

L_A0:      JNB        CAL,L_A1             ;уточняем, была калибр или измерение
           CLR        RDY1                 ;разрешаем дальнейшие преобразования
            AJMP       Lab_RETI             ;была калибровка, идти на выход

L_A1:      MOV        ADC1H_R,ADC1H        ;---------------------------------
            MOV        ADC1L_R,ADC1L        ; было измерение, копируем его рез

            CLR        RDY1                 ;разрешаем дальнейшие преобразования

;Блок возврата из прерываний----------------------------------------------
Lab_RETI:  POP        ACC                  ;---------------------------------
           POP        PSW                  ; восст контекста и глоб разр прер
           SETB       EA                   ;---------------------------------
           RETI                            ;возврат из блока обраб прерываний

;Начало осн программы-----------------------------------------------------
                ORG 100h
Lab_START: MOV        SP,#080h             ;определить указатель стека
           MOV        PLLCON,#00000000b    ;уст макс частоту ядра (12,58 МГц)
           NOP                             ;
           LCALL      Pod_INIT_RSN         ;иниц РСН
           LCALL      Pod_INIT_RON         ;иниц РОН

           LCALL      Pod_INIT_LCD         ;иниц ЖКИ

           LCALL      Pod_CLEAR_LCD        ;стирание ЖКИ

            SETB       EA                   ;глоб разрешение прерываний
;Начало основного цикла---------------------------------------------------
La_OSN:    NOP                             ;метка возврата в осн цикле

;Блок, предшествующий вызову подпрограммы опроса кнопки 0.
           MOV        R0,#KNOP0_R          ;
           MOV        R1,#NAKOPL0_R        ;
           LCALL      Pod_OPR_KNOP0        ;
;Блок, следующий за вызовом подпрограммы опроса кнопки (уст факта ее наж)
           MOV        ACC,@R0              ;
           JNB        ACC_1,La_1           ;

           CLR        ACC_1                ;кн была нажата, сброс флага нажат кн
           MOV        @R0,ACC              ;

            MOV        ADCMODE,#00010100b   ;АЦП1 вкл, реж внутр калибровки нуля

           MOV        DATA_IND_R,#0        ;-------------------------------------
           MOV        ADR_IND_R,#74        ; индикация  номера нажатой кнопки
           LCALL      Pod_PER_DAT_LCD      ;-------------------------------------

           LJMP       La_OSN               ;закрыть основной цикл
```

```
;Блок, предшествующий вызову подпрограммы опроса кнопки 1.
La_1:       MOV         R0,#KNOP1_R          ;
            MOV         R1,#NAKOPL1_R        ;
            LCALL       Pod_OPR_KNOP1        ;
;Блок, следующий за вызовом подпрограммы опроса кнопки (уст факта ее наж)
            MOV         ACC,@R0              ;
            JNB         ACC_1,La_2           ;

            CLR         ACC_1                ;кн была нажата, сброс флага нажат кн
            MOV         @R0,ACC              ;

           MOV      ADCMODE,#00010101b   ;АЦП1 вкл, реж внутр калибровки верхн пр

            MOV         DATA_IND_R,#1        ;-------------------------------------
            MOV         ADR_IND_R,#75        ; индикация  номера нажатой кнопки
            LCALL       Pod_PER_DAT_LCD      ;-------------------------------------

            LJMP        La_OSN               ;закрыть основной цикл

;Блок, предшествующий вызову подпрограммы опроса кнопки 2.
La_2:       MOV         R0,#KNOP2_R          ;
            MOV         R1,#NAKOPL2_R        ;
            LCALL       Pod_OPR_KNOP2        ;
;Блок, следующий за вызовом подпрограммы опроса кнопки (уст факта ее наж)
            MOV         ACC,@R0              ;
            JNB         ACC_1,La_3           ;

            CLR         ACC_1                ;кн была нажата, сброс флага нажат кн
            MOV         @R0,ACC              ;

             MOV         ADCMODE,#00010110b   ;АЦП1 вкл, реж сист калибровки нуля

            MOV         DATA_IND_R,#2        ;-------------------------------------
            MOV         ADR_IND_R,#76        ; индикация  номера нажатой кнопки
            LCALL       Pod_PER_DAT_LCD      ;-------------------------------------

            LJMP        La_OSN               ;закрыть основной цикл

;Блок, предшествующий вызову подпрограммы опроса кнопки 3.
La_3:       MOV         R0,#KNOP3_R          ;
            MOV         R1,#NAKOPL3_R        ;
            LCALL       Pod_OPR_KNOP3        ;
;Блок, следующий за вызовом подпрограммы опроса кнопки (уст факта ее наж)
            MOV         ACC,@R0              ;
            JNB         ACC_1,La_4           ;

            CLR         ACC_1                ;кн была нажата, сброс флага нажат кн
            MOV         @R0,ACC              ;

           MOV       ADCMODE,#00010111b   ;АЦП1 вкл, реж сист калибровки верхн пр

             MOV         DATA_IND_R,#3        ;-------------------------------------
            MOV         ADR_IND_R,#77        ; индикация  номера нажатой кнопки
            LCALL       Pod_PER_DAT_LCD      ;-------------------------------------

            LJMP        La_OSN               ;закрыть основной цикл

;Блок, предшествующий вызову подпрограммы опроса кнопки 4.
La_4:       MOV         R0,#KNOP4_R          ;
            MOV         R1,#NAKOPL4_R        ;
            LCALL       Pod_OPR_KNOP4        ;
;Блок, следующий за вызовом подпрограммы опроса кнопки (уст факта ее наж)
            MOV         ACC,@R0              ;
            JNB         ACC_1,La_5           ;

            CLR         ACC_1                ;кн была нажата, сброс флага нажат кн
            MOV         @R0,ACC              ;
```

```
            MOV         ADCMODE,#00010011b   ;АЦП1 вкл, реж циклич преобразований

            MOV         DATA_IND_R,#4        ;-----------------------------------
            MOV         ADR_IND_R,#78        ; индикация  номера нажатой кнопки
            LCALL       Pod_PER_DAT_LCD      ;-----------------------------------

            LJMP        La_OSN               ;закрыть основной цикл

;Блок, предшествующий вызову подпрограммы опроса кнопки 5.
La_5:       MOV         R0,#KNOP5_R          ;
            MOV         R1,#NAKOPL5_R        ;
            LCALL       Pod_OPR_KNOP5        ;
;Блок, следующий за вызовом подпрограммы опроса кнопки (уст факта ее наж)
            MOV         ACC,@R0              ;
            JNB         ACC_1,La_6           ;

            CLR         ACC_1                ;кн была нажата, сброс флага нажат кн
            MOV         @R0,ACC              ;

            LCALL       Pod_CLEAR_LCD        ;стирание ЖКИ

            LJMP        La_OSN               ;закрыть основной цикл

;Блок обработки результатов и вывода их на индикацию
La_6:       CLR         EA                   ;глоб запрещение прерываний

            MOV         BYTE_R,ADC1H_R       ;-----------------------------------
            MOV         R2,#0                ;  индикация ADC1H_R с адр 0 ОЗУ ЖКИ
            LCALL       Pod_IND_BYTE         ;-----------------------------------
            MOV         BYTE_R,ADC1L_R       ;-----------------------------------
            MOV         R2,#8                ;  индикация ADC1L_R с адр 8 ОЗУ ЖКИ
            LCALL       Pod_IND_BYTE         ;-----------------------------------

            SETB        EA                   ;глоб разрешение прерываний
            LJMP        La_OSN               ;закрыть основной цикл

;Подпрограммы-----------------------------------------------------------------
;-----------------------------------------------------------------------------
;Подпрограмма инициализации PCH.
;-----------------------------------------------------------------------------
Pod_INIT_RSN:
          MOV         PORT_KNOP,#11111111b ;сделать вх все линии порта кн
          MOV         PORT_IND,#00000000b  ;сделать вых все линии порта индик
;Блок настройки АЦП
            MOV         ADCSTAT,#00000000b   ;сбросить все флаги АЦП
            MOV         ADCMODE,#00000000b   ;АЦП0, АЦП1 выкл, реж "снято питание"
            MOV         ADC1CON,#01001000b   ;АЦП1: внешн ИОН, вход AIN3,
                                             ;униполярный режим
            MOV         SF,#045h             ;частота обновл выхода 20 Гц
            MOV         ICON,#00000000b      ;отключены все источники тока
            MOV         IE,#01000000b        ;разрешены только прерыв от АЦП
            RET

;-----------------------------------------------------------------------------
;Подпрограмма инициализации РОНов. Обнуляются все РОНы с адресами
;от NACH_ADR до KON_ADR при помощи косвенной адресации.
;-----------------------------------------------------------------------------
Pod_INIT_RON:
          MOV         R0,#NACH_ADR         ;установка начального адреса
Lk_0:     MOV         @R0,#0               ;обнуление очередного РОНа
          INC         R0                   ;переход к следующему адресу
          CJNE        R0,#KON_ADR,Lk_0     ;не достигли ли последнего адреса ?
          MOV         @R0,#0               ;обнуление последнего РОНа
          RET                              ;да, выход

;-----------------------------------------------------------------------
;Подпрограмма вывода на ЖКИ содержимого байта в двоичном коде (побитно) в
;виде восьми знакомест. Первым выводится СЗР байта. Байт предварительно
```

```
;должен быть помещен в РОН BYTE_R. В R2 предварительно следует поместить
;адрес ОЗУ ЖКИ, с которого начнется вывод на индикацию.
;-------------------------------------------------------------------------
Pod_IND_BYTE:
            MOV       R1,#0                 ;нач уст счетчика битов в байте

Lj_0:       MOV       ACC,BYTE_R            ;копируем байт в акк
            JB        ACC_7,Lj_1            ;
            MOV       DATA_IND_R,#0         ;
            AJMP      Lj_2                  ;
Lj_1:        MOV       DATA_IND_R,#1         ;
Lj_2:       MOV       ADR_IND_R,R2          ;
            RL        A                     ;
             MOV       BYTE_R,ACC            ; сдвиг байта влево циклический
            ACALL     Pod_PER_DAT_LCD       ;
            INC       R1                    ;
            INC       R2                    ;
             CJNE      R1,#8,Lj_0            ;
            RET

;Подключение модулей опроса кнопок и вывода данных на ЖКИ (с опросом ЖКИ)
          $INCLUDE   (C:\PR_ADUC\ADC1\knop.asm)
          $INCLUDE   (C:\PR_ADUC\ADC1\lcd_opr.asm)

;Конец исполняемого кода
          END
```

Программа позволяет производить циклические преобразования (измерения) и все виды калибровок для модуля ADC1 в выбранном канале (AIN3), выбранных режиме (униполярном) и частоте обновления выходных данных (19,79 Гц) с использованием в качестве ИОН АЦП внешнего источника (DA1). Результат каждого преобразования индицируется на ЖКИ в двоичном коде в виде 16-разрядного слова данных начиная со СЗР старшего байта (адрес 0 ОЗУ ЖКИ) и заканчивая МЗР младшего байта (адрес 15 ОЗУ ЖКИ). Обновление показаний ЖКИ производится с выбранной частотой обновления данных на выходе АЦП. Все функции кнопок идентичны их функциям, описанным выше для программы adc0.asm.

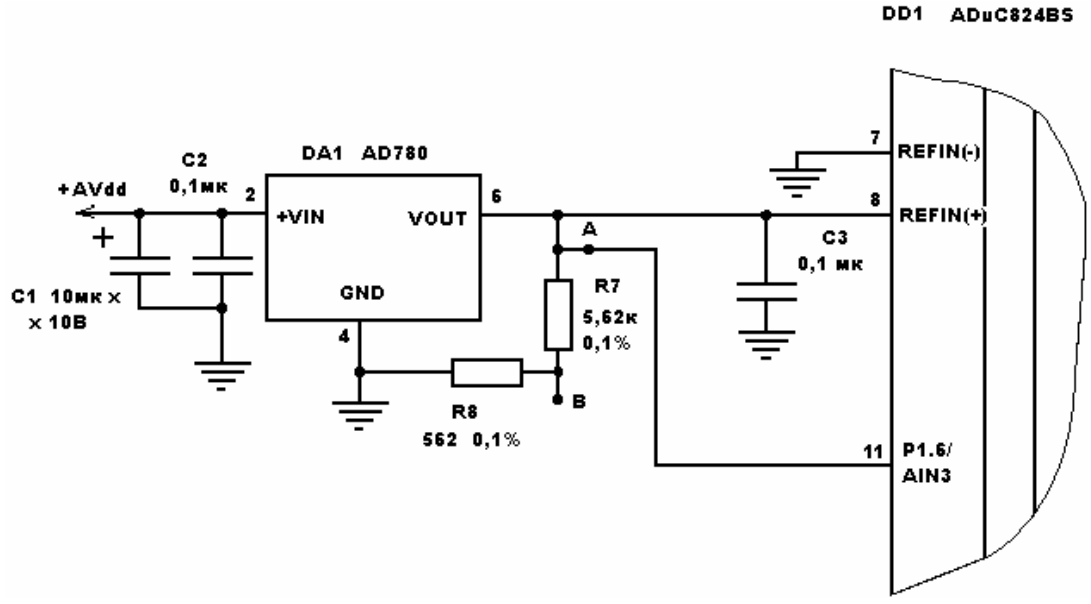


Рис. 3.3. Фрагмент схемы для исследования дополнительного АЦП

Проведенные с помощью программы adc1.asm исследования дали следующие результаты. При замкнутом на аналоговую «землю» входе AIN3 после сброса ADuC824 и запуска циклических преобразований до проведения системной и внутренней калибровок «нуля» 16-разрядное слово результата не равно нулю (имеются «единицы» в 4–5 младших разрядах) и меняются 1–2 младших разряда. Если после сброса ADuC824 первой производится системная калибровка «нуля», то после нее слово результата преобразований становится строго нулевым без меняющихся разрядов. Это полностью соответствует заявленному производителем в спецификации на ADuC824 значению реального разрешения ADC1, приведенному в табл. 1.1 (16 разрядов при частоте преобразования 20 Гц). В случае, если после сброса первой производится внутренняя калибровка «нуля», а после нее – системная калибровка «нуля», то добиться полностью нулевого результата запускаемых затем преобразований почему-то не удается. После проведения внутренней калибровки «нуля» ненулевое содержимое слова результата почти не меняется (что вполне объяснимо), а затем после проведения системной калибровки «нуля» оно уменьшается до наличия единиц в 2–3 младших разрядах и в нем колеблется 1 младший разряд. Таким образом, создается впечатление, что аппаратное подавление коэффициентов внутренней калибровки «нуля» коэффициентами системной калибровки «нуля» в специальных регистрах OF1H/OF1L происходит не полностью (часть битов в OF1H/OF1L остается неизменной). Подавление же заводских калибровочных коэффициентов, заносимых в OF1H/OF1L по умолчанию после сброса, при проведении любой из калибровок «нуля», очевидно, происходит нормально.

При проведении внутренней и системной калибровок верхнего предела наблюдается следующая картина (рассматриваем ситуацию, когда калибровка верхнего предела проводится после одной системной калибровки «нуля», как это рекомендуется в спецификации производителя ADuC824). После проведения внутренней калибровки верхнего предела с последующей контрольной подачей на вход AIN3 напряжения +2,5 В с выхода микросхемы ИОН AD780 (с вывода 6 DA1) и запуска циклических преобразований слово результата оказывается состоящим из одних единиц без меняющихся разрядов. Это идеальный результат. После проведения системной калибровки верхнего предела с предварительно поданным на вход AIN3 тем же самым напряжением +2,5 В с вывода 6 DA1 оказывается, что 2 младших разряда результата содержат «нули» и 1 младший разряд меняется. Повторная внутренняя калибровка верхнего предела снова дает идеальный результат «1111111111111111».

Программа, исходный текст которой приведен в файле adc1_a.asm (листинг 3.9), функционально идентична программе из предыдущего примера, но производит вывод результата преобразований на индикацию в десятичном коде (в виде десятизначного числа). Данный код может помочь при оценке интегральной нелинейности и ошибки смещения модуля АЦП1. Для достижения

наилучшей точности преобразования частота обновления данных на выходе модуля АЦП выбрана минимальной (5 Гц).

Листинг 3.9. Исследование дополнительного АЦП

```
;-------------------------------------------------------------------------------
;Демонстрационная программа использования модуля АЦП1 ADuC824.
;
;Данные в ЖКИ передаются по 4-битной шине, состояние ЖКИ опрашивается.
;
;Результат АЦП модуля АЦП1 индицируется на ЖКИ в десятичном виде.
;При нажатии на кнопку 0 производится внутренняя калибровка нуля.
;При этом по адресу 74 ОЗУ ЖКИ индицируется номер нажатой кнопки (0).
;При нажатии на кнопку 1 производится внутренняя калибровка верхнего предела.
;При этом по адресу 75 ОЗУ ЖКИ индицируется номер нажатой кнопки (1).
;При нажатии на кнопку 2 производится системная калибровка нуля.
;При этом по адресу 76 ОЗУ ЖКИ индицируется номер нажатой кнопки (2).
;При нажатии на кнопку 3 производится системная калибровка верхнего предела.
;При этом по адресу 77 ОЗУ ЖКИ индицируется номер нажатой кнопки (3).
;При нажатии на кнопку 4 производится переход в режим циклических преобразов.
;При этом по адресу 78 ОЗУ ЖКИ индицируется номер нажатой кнопки (4).
;При нажатии на кнопку 5 производится очистка ЖКИ.
;
; Используются прерывания от АЦП.
;-------------------------------------------------------------------------------
          $INCLUDE  (C:\ADuC\mod824)
          $INCLUDE  (C:\PR_ADUC\ADC1_a\824.inc)
;-------------------------------------------------------------------------------
;Описание битов, регистров и констант
;-------------------------------------------------------------------------------
;Порты и линии ввода-вывода
          PORT_KNOP    EQU      P0      ;порт кнопок

          PORT_IND     EQU      P2      ;порт индикации

          PORT_IND_0   EQU      P2_0    ;-----------------------------
          PORT_IND_1   EQU      P2_1    ;
          PORT_IND_2   EQU      P2_2    ; выходы индикации
          PORT_IND_3   EQU      P2_3    ;
          PORT_IND_4   EQU      P2_4    ;
          PORT_IND_5   EQU      P2_5    ;
          PORT_IND_6   EQU      P2_6    ;
          PORT_IND_7   EQU      P2_7    ;-----------------------------

          RW           EQU      PORT_IND_1 ;--------------------------
          RS           EQU      PORT_IND_2 ; линии управления ЖКИ
          E            EQU      PORT_IND_3 ;--------------------------

          _IN_KNOP0    EQU      P0_0    ;-----------------------------
          _IN_KNOP1    EQU      P0_1    ;
          _IN_KNOP2    EQU      P0_2    ;  входы кнопок
          _IN_KNOP3    EQU      P0_3    ;
          _IN_KNOP4    EQU      P0_4    ;
          _IN_KNOP5    EQU      P0_5    ;
          _IN_KNOP6    EQU      P0_6    ;
          _IN_KNOP7    EQU      P0_7    ;-----------------------------

;РОНы обслуживания ЖКИ и кнопок
          ADR_IND_R    DATA     030h    ;-----------------------------
          COM_IND_R    DATA     030h    ; РОНы обслуживания ЖКИ
          DATA_IND_R   DATA     031h    ;-----------------------------

          KNOP0_R      DATA     032h    ;-----------------------------
          KNOP1_R      DATA     033h    ;
          KNOP2_R      DATA     034h    ; РОНы, содержащие
          KNOP3_R      DATA     035h    ;
          KNOP4_R      DATA     036h    ; флаги нажатия и удержания
```

```
          KNOP5_R      DATA     037h    ;
          KNOP6_R      DATA     038h    ; каждой кнопки
          KNOP7_R      DATA     039h    ;----------------------------

          NAKOPL0_R    DATA     03Ah    ;----------------------------
          NAKOPL1_R    DATA     03Bh    ;
          NAKOPL2_R    DATA     03Ch    ; РОНы накопления
          NAKOPL3_R    DATA     03Dh    ;
          NAKOPL4_R    DATA     03Eh    ; значения подавления дребезга
          NAKOPL5_R    DATA     03Fh    ;
          NAKOPL6_R    DATA     040h    ; каждой кнопки
          NAKOPL7_R    DATA     041h    ;----------------------------

;РОНы обслуживания подпрограмм преобразования формы представления чисел
          IND_MILL_R   DATA     04Eh    ;РОН миллиардов дес числа
          IND_SOTMIL_R DATA     04Fh    ;РОН сотен миллионов дес числа
          IND_DESMIL_R DATA     050h    ;РОН десятков миллионов дес числа
          IND_MIL_R    DATA     051h    ;РОН миллионов дес числа
          IND_SOTTIS_R DATA     052h    ;РОН сотен тысяч дес числа
          IND_DESTIS_R DATA     053h    ;РОН десятков тысяч дес числа
          IND_TIS_R    DATA     054h    ;РОН тысяч дес числа
          IND_SOT_R    DATA     055h    ;РОН сотен дес числа
          IND_DES_R    DATA     056h    ;РОН десятков дес числа
          IND_ED_R     DATA     057h    ;РОН единиц дес числа

          BYTE_0_R     DATA     05Bh    ;байт 0 упакованного двоичн-дес числа
          BYTE_1_R     DATA     05Ch    ;байт 1 упакованного двоичн-дес числа
          BYTE_2_R     DATA     05Dh    ;байт 2 упакованного двоичн-дес числа
          BYTE_3_R     DATA     05Eh    ;байт 3 упакованного двоичн-дес числа
          BYTE_4_R     DATA     05Fh    ;байт 4 упакованного двоичн-дес числа

          ADC1L_R      DATA     068h    ;-----------------------------------
          ADC1H_R      DATA     069h    ;РОНы хранения результата преобр АЦП1

;Константы
          NACH_ADR     EQU      000h    ;начальный адрес обнуления РОНов
          KON_ADR      EQU      07Fh    ;конечный адрес обнуления РОНов
          POROG_K      EQU      10      ;порог подавления дребезга кнопок

;Начало исполняемого кода-------------------------------------------------------
               ORG 0h
           AJMP       Lab_START           ;идти на начало осн программы

               ORG 033h
           AJMP       Lab_ADC1            ;идти на нач блока обр прер от АЦП

                     ORG 05Fh
;Начало блока обработки прерывания по окончании преобразования АЦП1
;Прерывания от АЦП и глобально должны быть разрешены.
Lab_ADC1:  CLR        EA                  ;----------------------------------
           PUSH       PSW                 ; глоб запрет прер и сохр контекста
           PUSH       ACC                 ;----------------------------------

           JB         RDY1,L_A0           ;уточняем источник прерывания
            AJMP       Lab_RETI            ;прерывание не от АЦП1, идти на вых

L_A0:      JNB        CAL,L_A1            ;уточняем, была калибр или измерение
           CLR        RDY1                ;разрешаем дальнейшие преобразования
            AJMP       Lab_RETI            ;была калибровка, идти на выход

L_A1:      MOV        ADC1H_R,ADC1H       ;----------------------------------
            MOV        ADC1L_R,ADC1L       ; было измерение, копируем его рез

            CLR        RDY1                ;разрешаем дальнейшие преобразования

;Блок возврата из прерываний---------------------------------------------------
Lab_RETI:  POP        ACC                 ;----------------------------------
           POP        PSW                 ; восст контекста и глоб разр прер
```

```
           SETB       EA                    ;---------------------------------
           RETI                             ;возврат из блока обраб прерываний

;Начало осн программы-------------------------------------------------------
                ORG 100h
Lab_START: MOV        SP,#080h              ;определить указатель стека
           MOV        PLLCON,#00000000b     ;уст макс частоту ядра (12,58 МГц)
           NOP                              ;
           LCALL      Pod_INIT_RSN          ;иниц РСН
           LCALL      Pod_INIT_RON          ;иниц РОН

           LCALL      Pod_INIT_LCD          ;иниц ЖКИ

           LCALL      Pod_CLEAR_LCD         ;стирание ЖКИ

            SETB       EA                   ;глоб разрешение прерываний
;Начало основного цикла-----------------------------------------------------
La_OSN:    NOP                              ;метка возврата в осн цикле

;Блок, предшествующий вызову подпрограммы опроса кнопки 0.
           MOV        R0,#KNOP0_R           ;
           MOV        R1,#NAKOPL0_R         ;
           LCALL      Pod_OPR_KNOP0         ;
;Блок, следующий за вызовом подпрограммы опроса кнопки (уст факта ее наж)
           MOV        ACC,@R0               ;
           JNB        ACC_1,La_1            ;

           CLR        ACC_1                 ;кн была нажата, сброс флага нажат кн
           MOV        @R0,ACC               ;

            MOV        ADCMODE,#00010100b   ;АЦП1 вкл, реж внутр калибровки нуля

           MOV        DATA_IND_R,#0         ;-------------------------------------
           MOV        ADR_IND_R,#74         ; индикация  номера нажатой кнопки
           LCALL      Pod_PER_DAT_LCD       ;-------------------------------------

           LJMP       La_OSN                ;закрыть основной цикл

;Блок, предшествующий вызову подпрограммы опроса кнопки 1.
La_1:      MOV        R0,#KNOP1_R           ;
           MOV        R1,#NAKOPL1_R         ;
           LCALL      Pod_OPR_KNOP1         ;
;Блок, следующий за вызовом подпрограммы опроса кнопки (уст факта ее наж)
           MOV        ACC,@R0               ;
           JNB        ACC_1,La_2            ;

           CLR        ACC_1                 ;кн была нажата, сброс флага нажат кн
           MOV        @R0,ACC               ;

          MOV      ADCMODE,#00010101b   ;АЦП1 вкл, реж внутр калибровки верхн пр

           MOV        DATA_IND_R,#1         ;-------------------------------------
           MOV        ADR_IND_R,#75         ; индикация  номера нажатой кнопки
           LCALL      Pod_PER_DAT_LCD       ;-------------------------------------

           LJMP       La_OSN                ;закрыть основной цикл

;Блок, предшествующий вызову подпрограммы опроса кнопки 2.
La_2:      MOV        R0,#KNOP2_R           ;
           MOV        R1,#NAKOPL2_R         ;
           LCALL      Pod_OPR_KNOP2         ;
;Блок, следующий за вызовом подпрограммы опроса кнопки (уст факта ее наж)
           MOV        ACC,@R0               ;
           JNB        ACC_1,La_3            ;

           CLR        ACC_1                 ;кн была нажата, сброс флага нажат кн
           MOV        @R0,ACC               ;
```

```
            MOV         ADCMODE,#00010110b   ;АЦП1 вкл, реж сист калибровки нуля

            MOV         DATA_IND_R,#2        ;-------------------------------------
            MOV         ADR_IND_R,#76        ; индикация  номера нажатой кнопки
            LCALL       Pod_PER_DAT_LCD      ;-------------------------------------

            LJMP        La_OSN               ;закрыть основной цикл

;Блок, предшествующий вызову подпрограммы опроса кнопки 3.
La_3:       MOV         R0,#KNOP3_R          ;
            MOV         R1,#NAKOPL3_R        ;
            LCALL       Pod_OPR_KNOP3        ;
;Блок, следующий за вызовом подпрограммы опроса кнопки (уст факта ее наж)
            MOV         ACC,@R0              ;
            JNB         ACC_1,La_4           ;

            CLR         ACC_1                ;кн была нажата, сброс флага нажат кн
            MOV         @R0,ACC              ;

           MOV       ADCMODE,#00010111b   ;АЦП1 вкл, реж сист калибровки верхн пр

            MOV         DATA_IND_R,#3        ;-------------------------------------
            MOV         ADR_IND_R,#77        ; индикация  номера нажатой кнопки
            LCALL       Pod_PER_DAT_LCD      ;-------------------------------------

            LJMP        La_OSN               ;закрыть основной цикл

;Блок, предшествующий вызову подпрограммы опроса кнопки 4.
La_4:       MOV         R0,#KNOP4_R          ;
            MOV         R1,#NAKOPL4_R        ;
            LCALL       Pod_OPR_KNOP4        ;
;Блок, следующий за вызовом подпрограммы опроса кнопки (уст факта ее наж)
            MOV         ACC,@R0              ;
            JNB         ACC_1,La_5           ;

            CLR         ACC_1                ;кн была нажата, сброс флага нажат кн
            MOV         @R0,ACC              ;

            MOV         ADCMODE,#00010011b   ;АЦП1 вкл, реж циклич преобразований

            MOV         DATA_IND_R,#4        ;-------------------------------------
            MOV         ADR_IND_R,#78        ; индикация  номера нажатой кнопки
            LCALL       Pod_PER_DAT_LCD      ;-------------------------------------

            LJMP        La_OSN               ;закрыть основной цикл

;Блок, предшествующий вызову подпрограммы опроса кнопки 5.
La_5:       MOV         R0,#KNOP5_R          ;
            MOV         R1,#NAKOPL5_R        ;
            LCALL       Pod_OPR_KNOP5        ;
;Блок, следующий за вызовом подпрограммы опроса кнопки (уст факта ее наж)
            MOV         ACC,@R0              ;
            JNB         ACC_1,La_6           ;

            CLR         ACC_1                ;кн была нажата, сброс флага нажат кн
            MOV         @R0,ACC              ;

            LCALL       Pod_CLEAR_LCD        ;стирание ЖКИ

            LJMP        La_OSN               ;закрыть основной цикл

;Блок обработки результатов и вывода их на индикацию
La_6:       CLR         EA                   ;глоб запрещение прерываний

            MOV         R0,#BYTE_0_R         ;указание на РОНы для преобразования
            MOV         R1,#ADC1L_R          ;
            MOV         ADC1H_R+1,#0         ;обнуление ст незначащих РОНов
            MOV         ADC1H_R+2,#0         ;
```

```
           LCALL      B32BCD               ;преобр числа дискр из дв в дв-дес

           MOV        R0,#BYTE_0_R         ;указание на РОНы для преобразования
           MOV        R1,#IND_MILL_R       ;
           LCALL      BCD10BCD            ;пр числа дискр из уп дв-дес в неуп дес

           MOV        R0,#IND_MILL_R       ; индицировать с РОНа IND_MILL_R
           MOV        R1,#0                ; индицировать с адр 0 ЖКИ
           LCALL      Pod_IND_10ZN         ;индикация числа дискрет

           SETB       EA                   ;глоб разрешение прерываний

           LJMP       La_OSN               ;закрыть основной цикл

;Подпрограммы-----------------------------------------------------------------

;-------------------------------------------------------------------------------
;Подпрограмма инициализации РСН.
;-------------------------------------------------------------------------------
Pod_INIT_RSN:
          MOV        PORT_KNOP,#11111111b ;сделать вх все линии порта кн
          MOV        PORT_IND,#00000000b  ;сделать вых все линии порта индик
;Блок настройки АЦП
           MOV        ADCSTAT,#00000000b   ;сбросить все флаги АЦП
           MOV        ADCMODE,#00000000b   ;АЦП0, АЦП1 выкл, реж "снято питание"
           MOV        ADC1CON,#01001000b   ;АЦП1: внешн ИОН, вход AIN3,
                                           ;униполярный режим
          MOV        SF,#0FFh             ;частота обновл выхода 5 Гц
           MOV        ICON,#00000000b      ;отключены все источники тока
           MOV        IE,#01000000b        ;разрешены только прерыв от АЦП
          RET

;-------------------------------------------------------------------------------
;Подпрограмма инициализации РОНов. Обнуляются все РОНы с адресами
;от NACH_ADR до KON_ADR при помощи косвенной адресации.
;-------------------------------------------------------------------------------
Pod_INIT_RON:
          MOV        R0,#NACH_ADR         ;установка начального адреса
Lk_0:     MOV        @R0,#0               ;обнуление очередного РОНа
          INC        R0                   ;переход к следующему адресу
          CJNE       R0,#KON_ADR,Lk_0     ;не достигли ли последнего адреса ?
          MOV        @R0,#0               ;обнуление последнего РОНа
          RET                             ;да, выход

;-------------------------------------------------------------------------------
;Подпрограмма вывода на ЖКИ содержимого буфера из 10 РОНов в виде 10 знакомест.
;R0 должен указывать на РОН в буфере, содержимое которого индицируется первым.
;R1 должен содержать нач адрес знакоместа в ОЗУ ЖКИ.
;-------------------------------------------------------------------------------
Pod_IND_10ZN:
           MOV        R2,#0                ;нач уст ст цикла
Ii_0:      MOV        DATA_IND_R,@R0       ;
           MOV        ADR_IND_R,R1         ;
           LCALL      Pod_PER_DAT_LCD      ; индикация очередного символа
           INC        R0                   ;
           INC        R1                   ;
           INC        R2                   ;
           CJNE       R2,#10,Ii_0          ;
           RET

;Подключение модулей опроса кнопок, вывода данных на ЖКИ (с опросом ЖКИ)
;и преобразования представления чисел
          $INCLUDE   (C:\PR_ADUC\ADC1_a\knop.asm)
          $INCLUDE   (C:\PR_ADUC\ADC1_a\lcd_opr.asm)
          $INCLUDE   (C:\PR_ADUC\ADC1_a\preobr.asm)

;Конец исполняемого кода
          END
```

По итогам проведенных экспериментов хочется заметить что, даже несмотря на наличие не совсем понятных особенностей калибровки АЦП1, основной и дополнительный модули АЦП ADuC824, в целом, соответствуют параметрам, заявленным в их спецификациях, и оставляют приятное впечатление от своей работы. Что же касается упомянутых особенностей, то следует, во-первых, принять во внимание, что эксперименты проводились только с одним экземпляром ADuC824, во-вторых, выявленные «странности» в поведении модуля АЦП1 можно легко учесть в целевом программном обеспечении, не допустив, таким образом, потерь в точности и разрешающей способности преобразований.

В заключение, вниманию читателей предлагается программа, реализующая измерения температуры кристалла ADuC824 с помощью внутреннего температурного датчика. Исходный текст этой программы приведен в файле adc1_tem.asm (листинг 3.10), а для работы с ней требуется макет, где к ADuC824 подключен только один ЖКИ по схеме, показанной на рис. 3.2.

Листинг 3.10. Измерение температуры кристалла

```
;-------------------------------------------------------------------------------
;Демонстрационная программа использования модуля АЦП1 ADuC824 с подключенным
;внутренним температурным датчиком.
;Данные в ЖКИ передаются по 4-битной шине, состояние ЖКИ опрашивается.
;
;Результат АЦП модуля АЦП1 индицируется на ЖКИ в двоичном виде начиная со СЗР
;старшего байта и заканчивая МЗР младшего байта (в верхней строке) и
;в десятичном виде в градусах Цельсия (в нижней строке).
;
; Используются прерывания от АЦП.
;-------------------------------------------------------------------------------
          $INCLUDE  (C:\ADuC\mod824)
          $INCLUDE  (C:\PR_ADUC\ADC1_tem\824.inc)
;-------------------------------------------------------------------------------
;Описание битов, регистров и констант
;-------------------------------------------------------------------------------
;Порты и линии ввода-вывода
          PORT_IND     EQU      P2      ;порт индикации

          PORT_IND_0   EQU      P2_0    ;-----------------------------
          PORT_IND_1   EQU      P2_1    ;
          PORT_IND_2   EQU      P2_2    ; выходы индикации
          PORT_IND_3   EQU      P2_3    ;
          PORT_IND_4   EQU      P2_4    ;
          PORT_IND_5   EQU      P2_5    ;
          PORT_IND_6   EQU      P2_6    ;
          PORT_IND_7   EQU      P2_7    ;-----------------------------

          RW           EQU      PORT_IND_1 ;-------------------------
          RS           EQU      PORT_IND_2 ; линии управления ЖКИ
          E            EQU      PORT_IND_3 ;-------------------------
;РОНы обслуживания ЖКИ
          ADR_IND_R    DATA     030h    ;-----------------------------
          COM_IND_R    DATA     030h    ; РОНы обслуживания ЖКИ
          DATA_IND_R   DATA     031h    ;-----------------------------

;РОНы обслуживания подпрограмм преобразования формы представления чисел
          IND_DESTIS_R DATA     053h    ;РОН десятков тысяч дес числа
          IND_TIS_R    DATA     054h    ;РОН тысяч дес числа
          IND_SOT_R    DATA     055h    ;РОН сотен дес числа
          IND_DES_R    DATA     056h    ;РОН десятков дес числа
```

```
          IND_ED_R     DATA     057h    ;РОН единиц дес числа

          ADC1L_R      DATA     059h    ;-----------------------------------
          ADC1H_R      DATA     05Ah    ;РОНы хранения результата преобр АЦП1

          BYTE_R       DATA     060h    ;вспомогательный РОН для вывода
                                            ;байта на индикатор

;Константы
          NACH_ADR     EQU      000h    ;начальный адрес обнуления РОНов
          KON_ADR      EQU      07Fh    ;конечный адрес обнуления РОНов


;Начало исполняемого кода-------------------------------------------------------
               ORG 0h
           AJMP       Lab_START           ;идти на начало осн программы

               ORG 033h
           AJMP       Lab_ADC1            ;идти на нач блока обр прер от АЦП

               ORG 05Fh
;Начало блока обработки прерывания по окончании преобразования АЦП1
;Прерывания от АЦП и глобально должны быть разрешены.
Lab_ADC1:  CLR        EA                  ;-----------------------------------
           PUSH       PSW                 ; глоб запрет прер и сохр контекста
           PUSH       ACC                 ;-----------------------------------

           JB         RDY1,L_A0           ;уточняем источник прерывания
            AJMP       Lab_RETI            ;прерывание не от АЦП1, идти на вых

L_A0:      JNB        CAL,L_A1            ;уточняем, была калибр или измерение
           CLR        RDY1                ;разрешаем дальнейшие преобразования
            AJMP       Lab_RETI            ;была калибровка, идти на выход

L_A1:      MOV        ADC1H_R,ADC1H       ;-----------------------------------
           MOV        ADC1L_R,ADC1L       ; было измерение, копируем его рез

           CLR        RDY1                ;разрешаем дальнейшие преобразования

;Блок возврата из прерываний---------------------------------------------------
Lab_RETI:  POP        ACC                 ;-----------------------------------
           POP        PSW                 ; восст контекста и глоб разр прер
           SETB       EA                  ;-----------------------------------
           RETI                           ;возврат из блока обраб прерываний

;Начало осн программы-----------------------------------------------------------
               ORG 100h
Lab_START: MOV        SP,#080h            ;определить указатель стека
           MOV        PLLCON,#00000000b   ;уст макс частоту ядра (12,58 МГц)
           NOP                            ;
           LCALL      Pod_INIT_RSN        ;иниц РСН
           LCALL      Pod_INIT_RON        ;иниц РОН

           LCALL      Pod_INIT_LCD        ;иниц ЖКИ

           LCALL      Pod_CLEAR_LCD       ;стирание ЖКИ

           MOV        ADR_IND_R,#70        ;-----------------------------------
           MOV       DATA_IND_R,#0EFh     ;
           ACALL     Pod_PER_DAT_LCD      ;   индикация заставки указания
           MOV       ADR_IND_R,#71        ;   единиц измерения температуры
           MOV       DATA_IND_R,#43h      ;   (градусы Цельсия)
           ACALL     Pod_PER_DAT_LCD      ;-----------------------------------

            SETB       EA                  ;глоб разрешение прерываний

           MOV        ADCMODE,#00010011b  ;АЦП1 вкл, реж циклич преобразований
```

```
;Начало основного цикла-------------------------------------------------------
La_OSN:    NOP                          ;метка возврата в осн цикле
           NOP
            NOP
            NOP
            NOP
            NOP
            NOP

;Блок обработки результатов и вывода их на индикацию
           CLR        EA                ;глоб запрещение прерываний

           MOV        BYTE_R,ADC1H_R    ;-----------------------------------
            MOV        R2,#0             ;  индикация ADC1H_R с адр 0 ОЗУ ЖКИ
            LCALL      Pod_IND_BYTE      ;-----------------------------------
           MOV        BYTE_R,ADC1L_R    ;-----------------------------------
            MOV        R2,#8             ;  индикация ADC1L_R с адр 8 ОЗУ ЖКИ
            LCALL      Pod_IND_BYTE      ;-----------------------------------

           MOV        A,ADC1H_R         ;
           ANL        A,#01111111b      ; вычит из ст байта результата 80h
           MOV        R5,#0             ;
            MOV        R4,A              ;
           LCALL      B16BCDD           ;преобр числа из дв в дв-дес

          LCALL      BCD5BCD           ;преобаз числа из уп дв-дес в неуп дес

           MOV        R0,#IND_DESTIS_R  ; индицировать с РОНа IND_DESTIS_R
           MOV        R1,#64            ; индицировать с адр 64 ЖКИ
           LCALL      Pod_IND_5ZN       ;индикация температуры в градусах


           SETB       EA                ;глоб разрешение прерываний

           LJMP       La_OSN            ;закрыть основной цикл

;Подпрограммы-----------------------------------------------------------------

;-------------------------------------------------------------------------------
;Подпрограмма инициализации РСН.
;-------------------------------------------------------------------------------
Pod_INIT_RSN:
          MOV        PORT_IND,#00000000b  ;сделать вых все линии порта индик
;Блок настройки АЦП
          MOV        ADCSTAT,#00000000b   ;сбросить все флаги АЦП
          MOV        ADCMODE,#00000000b   ;АЦП0, АЦП1 выкл, реж "снято питание"
           MOV        ADC1CON,#00100000b   ;АЦП1: внутр ИОН,
                                           ;вход - внутр темпер датчик,
                                            ;биполярный режим
          MOV        SF,#0FFh             ;частота обновл выхода 5 Гц
           MOV        ICON,#00000000b      ;отключены все источники тока
           MOV        IE,#01000000b        ;разрешены только прерыв от АЦП
          RET

;-------------------------------------------------------------------------------
;Подпрограмма инициализации РОНов. Обнуляются все РОНы с адресами
;от NACH_ADR до KON_ADR при помощи косвенной адресации.
;-------------------------------------------------------------------------------
Pod_INIT_RON:
          MOV        R0,#NACH_ADR         ;установка начального адреса
Lk_0:     MOV        @R0,#0               ;обнуление очередного РОНа
          INC        R0                   ;переход к следующему адресу
          CJNE       R0,#KON_ADR,Lk_0     ;не достигли ли последнего адреса ?
          MOV        @R0,#0               ;обнуление последнего РОНа
          RET                             ;да, выход

;-------------------------------------------------------------------------
;Подпрограмма вывода на ЖКИ содержимого байта в двоичном коде (побитно) в
```

```
;виде восьми знакомест. Первым выводится СЗР байта. Байт предварительно
;должен быть помещен в РОН BYTE_R. В R2 предварительно следует поместить
;адрес ОЗУ ЖКИ, с которого начнется вывод на индикацию.
;-------------------------------------------------------------------------
Pod_IND_BYTE:
            MOV       R1,#0                ;нач уст счетчика битов в байте

Lj_0:       MOV       ACC,BYTE_R           ;копируем байт в акк
            JB        ACC_7,Lj_1           ;
            MOV       DATA_IND_R,#0        ;
            AJMP      Lj_2                 ;
Lj_1:        MOV       DATA_IND_R,#1        ;
Lj_2:       MOV       ADR_IND_R,R2         ;
            RL        A                    ;
             MOV       BYTE_R,ACC           ; сдвиг байта влево циклический
            ACALL     Pod_PER_DAT_LCD      ;
            INC       R1                   ;
            INC       R2                   ;
            CJNE      R1,#8,Lj_0           ;
            RET

;-------------------------------------------------------------------------
;Подпрограмма вывода на ЖКИ содержимого буфера из 5 РОНов в виде 5 знакомест.
;R0 должен указывать на РОН в буфере, содержимое которого индицируется первым.
;R1 должен содержать нач адрес знакоместа в ОЗУ ЖКИ.
;-------------------------------------------------------------------------
Pod_IND_5ZN:
            MOV        R2,#0               ;нач уст ст цикла
Ii_1:       MOV        DATA_IND_R,@R0      ;
            MOV        ADR_IND_R,R1        ;
            LCALL      Pod_PER_DAT_LCD     ; индикация очередного символа
            INC        R0                  ;
            INC        R1                  ;
            INC        R2                  ;
            CJNE       R2,#5,Ii_1          ;
            RET

;Подключение модулей вывода данных на ЖКИ (с опросом ЖКИ)
;и преобразования представления чисел
          $INCLUDE   (C:\PR_ADUC\ADC1_tem\lcd_opr.asm)
          $INCLUDE   (C:\PR_ADUC\ADC1_tem\preobr.asm)

;Конец исполняемого кода
          END
```

Внешние аналоговые входы, кнопки управления и внешний ИОН в данном случае не используются. Программа производит циклические преобразования входного сигнала с внутреннего температурного датчика, подключенного к входу модуля АЦП1. Результат каждого преобразования отображается на ЖКИ в двоичном коде в виде 16-разрядного слова данных начиная с СЗР старшего байта (адрес 0 ОЗУ ЖКИ) и заканчивая МЗР младшего байта (адрес 15 ОЗУ ЖКИ). Кроме того, этот же результат отображается в десятичном виде в целых °С в нижней строке ЖКИ. Обновление показаний ЖКИ производится с частотой обновления данных на выходе АЦП (5 Гц). Согласно спецификации температурного датчика ADuC824 (табл. 1.1) его точность составляет ±2 °С, поэтому нет особого смысла выводить на индикатор значение температуры с десятыми и сотыми долями °С. Как можно наблюдать на экране ЖКИ, старший разряд 16-разрядного слова результата преобразования равен единице, так как в соответствии с настройкой режима АЦП1 (биполярный) «нулем» шкалы кодирования

является число 8000h (1000000000000000b). Для преобразования результата в десятичную форму программа использует только его старший байт, содержащий целую часть значения температуры в °С (примечания к табл. 1.8). Двоичное значение температуры в целых °С получается при сбросе старшего бита этого байта, что эквивалентно вычитанию из байта числа 80h. Следует заметить, что предложенный способ получения десятичного значения температуры будет давать правильный результат только для температуры, большей 0 °С. Придумать способ получения и вывода на индикацию отрицательных значений температуры в °С читателям предлагается самостоятельно.

После получения результатов от экспериментов с программой adc1_tem.asm читателям предлагается сравнить их с результатами, выдаваемыми программным анализатором АЦП WASP (см. п. 2.6).

3.5. Операции с Flash/EE-памятью данных

Работа пользовательского программного обеспечения с Flash/EE-памятью данных ADuC824 (далее мы будем называть ее EEPROM) иллюстрируется демонстрационной программой, исходный текст которой находится в файле eeprom.asm (листинг 3.11). Для экспериментов с EEPROM подойдет макет, собранный для программ из двух предыдущих примеров (рис. 3.2). Поскольку программе для работы требуются только кнопки управления и ЖКИ, то микросхему ИОН DA1 с элементами ее «обвязки» от макета можно вообще отключить.

Листинг 3.11. Работа с памятью EEPROM

```
;-------------------------------------------------------------------------------
;Тестовая программа обслуживания встроенного EEPROM данных ADuC824.
;Данные в ЖКИ передаются по 4-битной шине, состояние ЖКИ опрашивается,
;подпрограмма опроса возвращает управление, когда ЖКИ готов к приему данных.
;При нажатии на кн 0 производится перебор адресов EEPROM от 0 до 255.
;При нажатии на кн 1 производится перебор кодов символов для записи в EEPROM.
;При нажатии на кн 2 производится запись в EEPROM выбранного байта данных.
;При нажатии на кн 3 производится чтение из EEPROM байта данных.
;Выбранный адрес EEPROM отображается в десятичной форме на ЖКИ начиная
;с адреса 0 ОЗУ ЖКИ.
;ASCII-код выбранного для записи в EEPROM байта отображается по адр 64 ОЗУ ЖКИ.
;ASCII-код записанного в EEPROM по текущему адресу байта отображается по адресу
;68 ОЗУ ЖКИ.
;ASCII-код считанного из EEPROM по текущему адресу байта отображается по адресу
;72 ОЗУ ЖКИ.
;-------------------------------------------------------------------------------
          $INCLUDE  (C:\ADuC\mod824)
          $INCLUDE  (C:\PR_ADUC\EEPROM\824.inc)
;-------------------------------------------------------------------------------
;Описание битов, регистров и констант
;-------------------------------------------------------------------------------
          PORT_KNOP    EQU      P0       ;порт кнопок

          PORT_IND     EQU      P2       ;порт индикации

          _IN_KNOP0    EQU      P0_0     ;------------------------------
          _IN_KNOP1    EQU      P0_1     ;
          _IN_KNOP2    EQU      P0_2     ;  входы кнопок
```

```
          _IN_KNOP3    EQU      P0_3    ;
          _IN_KNOP4    EQU      P0_4    ;
          _IN_KNOP5    EQU      P0_5    ;
          _IN_KNOP6    EQU      P0_6    ;
          _IN_KNOP7    EQU      P0_7    ;-----------------------------

          POROG_K      EQU      50      ;порог подавления дребезга кнопок
          NACH_ADR     EQU      000h    ;начальный адрес обнуления РОНов
          KON_ADR      EQU      07Fh    ;конечный адрес обнуления РОНов

          KNOP0_R      DATA     032h    ;-----------------------------
          KNOP1_R      DATA     033h    ;
          KNOP2_R      DATA     034h    ; регистры, содержащие
          KNOP3_R      DATA     035h    ;
          KNOP4_R      DATA     036h    ; флаги нажатия и удержания
          KNOP5_R      DATA     037h    ;
          KNOP6_R      DATA     038h    ; каждой кнопки
          KNOP7_R      DATA     039h    ;-----------------------------

          NAKOPL0_R    DATA     03Ah    ;-----------------------------
          NAKOPL1_R    DATA     03Bh    ;
          NAKOPL2_R    DATA     03Ch    ; регистры накопления
          NAKOPL3_R    DATA     03Dh    ;
          NAKOPL4_R    DATA     03Eh    ; значения подавления дребезга
          NAKOPL5_R    DATA     03Fh    ;
          NAKOPL6_R    DATA     040h    ; каждой кнопки
          NAKOPL7_R    DATA     041h    ;-----------------------------

          PORT_IND_0   EQU      P2_0    ;-----------------------------
          PORT_IND_1   EQU      P2_1    ;
          PORT_IND_2   EQU      P2_2    ; выходы индикации
          PORT_IND_3   EQU      P2_3    ;
          PORT_IND_4   EQU      P2_4    ;
          PORT_IND_5   EQU      P2_5    ;
          PORT_IND_6   EQU      P2_6    ;
          PORT_IND_7   EQU      P2_7    ;-----------------------------

          RW           EQU      PORT_IND_1 ;--------------------------
          RS           EQU      PORT_IND_2 ; линии обслуживания ЖКИ
          E            EQU      PORT_IND_3 ;--------------------------

          ADR_IND_R    DATA     030h    ;-----------------------------
          COM_IND_R    DATA     030h    ; РОНы обслуживания ЖКИ
          DATA_IND_R   DATA     031h    ;-----------------------------

          EEDATA_R     DATA     044h    ;РОН данных EEPROM
          EEADR_R      DATA     045h    ;РОН адреса EEPROM
          DDD_R        DATA     047h    ;РОН выбора байта данных

          _ERR_WR_EE   BIT      02h     ;флаг "ошибка зап в EEPROM"

;РОНы обслуживания подпрограмм преобразования формы представления чисел
          IND_DESTIS_R DATA     053h    ;РОН десятков тысяч дес числа
          IND_TIS_R    DATA     054h    ;РОН тысяч дес числа
          IND_SOT_R    DATA     055h    ;РОН сотен дес числа
          IND_DES_R    DATA     056h    ;РОН десятков дес числа
          IND_ED_R     DATA     057h    ;РОН единиц дес числа

;Начало исполняемого кода-------------------------------------------------------
               ORG 0h
           LJMP       Lab_START

;Начало основной программы------------------------------------------------------
               ORG 100h
Lab_START: MOV        SP,#080h            ;определить указатель стека
           MOV        PLLCON,#00000000b   ;уст макс частоту ядра (12,58 МГц)
           NOP                            ;
```

```
            ACALL       Pod_INIT_RSN          ;иниц РСН
            ACALL       Pod_INIT_RON          ;иниц РОН
            ACALL       Pod_INIT_LCD          ;иниц ЖКИ
            ACALL       Pod_CLEAR_LCD         ;стирание ЖКИ

            MOV         DDD_R,#021h           ;уст нач значения байта данных

            MOV         R5,#0                 ; исх данные - адрес EEPROM
            MOV         R4,EEADR_R            ;
            LCALL       B16BCDD               ;преобраз двоичн в упак двоичн-дес
            LCALL       BCD5BCD               ;преобраз упак двоичн-дес в неупак
            MOV         DATA_IND_R,IND_SOT_R  ;-----------------------------------
            MOV         ADR_IND_R,#0          ; индикация сотен
            LCALL       Pod_PER_DAT_LCD       ;-----------------------------------
            MOV         DATA_IND_R,IND_DES_R  ;-----------------------------------
            MOV         ADR_IND_R,#1          ; индикация дес
            LCALL       Pod_PER_DAT_LCD       ;-----------------------------------
            MOV         DATA_IND_R,IND_ED_R   ;-----------------------------------
            MOV         ADR_IND_R,#2          ; индикация ед
            LCALL       Pod_PER_DAT_LCD       ;-----------------------------------

;Начало основного цикла-------------------------------------------------------
La_OSN:     NOP                               ;метка возврата осн цикла

        ;Блок, предшествующий вызову подпрограммы опроса кнопки 0.
            MOV         R0,#KNOP0_R           ;
            MOV         R1,#NAKOPL0_R         ;
            ACALL       Pod_OPR_KNOP0         ;
       ;Блок, следующий за вызовом подпрограммы опроса кнопки (уст факта ее наж)
            MOV         ACC,@R0               ;
            JNB         ACC_1,La_0            ;

            CLR         ACC_1                 ;кн была нажата, сброс флага нажат кн
            MOV         @R0,ACC               ;

            MOV         ACC,EEADR_R           ;-----------------------------------
            INC         A                     ; перебор возможных адресов EEPROM
            MOV         EEADR_R,A             ;-----------------------------------

            MOV         R5,#0                 ; исх данные - адрес EEPROM
            MOV         R4,EEADR_R            ;
            LCALL       B16BCDD               ;преобраз двоичн в упак двоичн-дес
            LCALL       BCD5BCD               ;преобраз упак двоичн-дес в неупак
            MOV         DATA_IND_R,IND_SOT_R  ;-----------------------------------
            MOV         ADR_IND_R,#0          ; индикация сотен
            LCALL       Pod_PER_DAT_LCD       ;-----------------------------------
            MOV         DATA_IND_R,IND_DES_R  ;-----------------------------------
            MOV         ADR_IND_R,#1          ; индикация дес
            LCALL       Pod_PER_DAT_LCD       ;-----------------------------------
            MOV         DATA_IND_R,IND_ED_R   ;-----------------------------------
            MOV         ADR_IND_R,#2          ; индикация ед
            LCALL       Pod_PER_DAT_LCD       ;-----------------------------------

       ;Блок, предшествующий вызову подпрограммы опроса кнопки 1.
La_0:       MOV         R0,#KNOP1_R           ;
            MOV         R1,#NAKOPL1_R         ;
            ACALL       Pod_OPR_KNOP1         ;
       ;Блок, следующий за вызовом подпрограммы опроса кнопки (уст факта ее наж)
            MOV         ACC,@R0               ;
            JNB         ACC_1,La_2            ;

            CLR         ACC_1                 ;кн была нажата, сброс флага нажат кн
            MOV         @R0,ACC               ;

            MOV         ACC,DDD_R             ;-----------------------------------
            INC         A                     ;
            CJNE        A,#06Fh,La_1          ; перебор возможных значений байта
            MOV         A,#021h               ; данных
```

```
La_1:       MOV         DDD_R,A              ;-----------------------------------
            MOV         DATA_IND_R,DDD_R     ;-----------------------------------
            MOV         ADR_IND_R,#64        ; индикация выбранного байта
            ACALL       Pod_PER_DAT_LCD      ;-----------------------------------

        ;Блок, предшествующий вызову подпрограммы опроса кнопки 2.
La_2:       MOV         R0,#KNOP2_R          ;
            MOV         R1,#NAKOPL2_R        ;
            ACALL       Pod_OPR_KNOP2        ;

        ;Блок, следующий за вызовом подпрограммы опроса кнопки (уст факта ее наж)
            MOV         ACC,@R0              ;
            JNB         ACC_1,La_3           ;

            CLR         ACC_1                ;кн была нажата, сброс флага нажат кн
            MOV         @R0,ACC              ;

            MOV         EEDATA_R,DDD_R       ;
            ACALL       Pod_WR_EEPROM        ;запись байта в EEPROM

            MOV         DATA_IND_R,EEDATA_R  ;-----------------------------------
            MOV         ADR_IND_R,#68        ; индикация записанного байта
            ACALL       Pod_PER_DAT_LCD      ;-----------------------------------

       ;Блок, предшествующий вызову подпрограммы опроса кнопки 3.
La_3:       MOV         R0,#KNOP3_R          ;
            MOV         R1,#NAKOPL3_R        ;
            ACALL       Pod_OPR_KNOP3        ;

        ;Блок, следующий за вызовом подпрограммы опроса кнопки (уст факта ее наж)
            MOV         ACC,@R0              ;
            JNB         ACC_1,La_4           ;

            CLR         ACC_1                ;кн была нажата, сброс флага нажат кн
            MOV         @R0,ACC              ;

            MOV         EEDATA_R,#0FFh       ;контрольная порча регистра EEDATA_R
            ACALL       Pod_RD_EEPROM        ;чтение байта из EEPROM

            MOV         DATA_IND_R,EEDATA_R  ;-----------------------------------
            MOV         ADR_IND_R,#72        ; индикация считанного байта
            ACALL       Pod_PER_DAT_LCD      ;-----------------------------------

La_4:       AJMP        La_OSN               ;идти на начало осн цикла

;Подпрограммы-----------------------------------------------------------------------
;-----------------------------------------------------------------------------------
;Подпрограмма записи одного байта в EEPROM данных по адресу, предварительно
;помещенному в РОН EEADR_R. Байт данных следует предварительно поместить
;в РОН EEDATA_R. В случае ошибки записи (ошибки верификации) установится флаг
;_ERR_WR_EE - "ошибка записи в EEPROM". Адресуется полько 256 байт EEPROM.
;-----------------------------------------------------------------------------------
Pod_WR_EEPROM:
            ACALL       Pod_EEADR            ;подпр определения адр стр EEPROM
            MOV         ECON,#01h            ;читать страницу
            ACALL       Pod_EEDATA_WR        ;подпр определения адр байта для зап
            MOV         ECON,#05h            ;стереть страницу
            MOV         ECON,#02h            ;записать страницу в EEPROM
            MOV         ECON,#04h            ;верифицировать страницу
            MOV         ACC,ECON             ;не было ли ошибки записи ?
            JZ          Lg_0                 ;нет, выход
            SETB        _ERR_WR_EE           ;да, уст флаг "ошибка записи в EEPROM"
Lg_0:       RET

;-----------------------------------------------------------------------------------
;Подпрограмма определения адреса страницы EEPROM из абсолютного адреса
;-----------------------------------------------------------------------------------
```

```
Pod_EEADR:
           MOV        ACC,EEADR_R          ;--------------------------------
           RR         A                    ; Деление абс адреса на 4 для
           RR         A                    ; получения адреса стр EEPROM
           ANL        ACC,#00111111b       ;--------------------------------
           MOV        EADRL,ACC            ; загрузить подготовл адр страницы
           RET

;-------------------------------------------------------------------------------
;Подпрограмма определения адреса байта (для записи) внутри страницы из абс адреса
;-------------------------------------------------------------------------------
Pod_EEDATA_WR:
           MOV        ACC,EEADR_R          ; выделение двух мл бит
           ANL        ACC,#00000011b       ;--------------------------------
           JZ         Lg_50                ;переход, если оба бита равны 0
           JNB        ACC_1,Lg_40          ;переход, если ст бит равен 0
           JNB        ACC_0,Lg_30          ;переход, если мл бит равен 0
           MOV        EDATA4,EEDATA_R      ;если оба бита равны 0
           RET                             ;
Lg_30:     MOV        EDATA3,EEDATA_R      ;
           RET                             ;
Lg_40:     MOV        EDATA2,EEDATA_R      ;
           RET                             ;
Lg_50:     MOV        EDATA1,EEDATA_R      ;
           RET

;-------------------------------------------------------------------------------
;Подпрограмма чтения одного байта из EEPROM данных по адресу, предварительно
;помещенному в РОН EEADR_R. Прочитанный байт данных оказывается в РОНе EEDATA_R.
;Ошибка чтения не диагностируется. Адресуется полько 256 байт EEPROM.
;-------------------------------------------------------------------------------
Pod_RD_EEPROM:
           ACALL      Pod_EEADR            ;подпр определения адр стр EEPROM
           MOV        ECON,#01h            ;читать страницу
           ACALL      Pod_EEDATA_RD        ;подпр определения адр байта для чтен
           RET

;-------------------------------------------------------------------------------
;Подпрограмма определения адреса байта (для чтения) внутри страницы из абс адреса
;-------------------------------------------------------------------------------
Pod_EEDATA_RD:
           MOV        ACC,EEADR_R          ; выделение двух мл бит
           ANL        ACC,#00000011b       ;--------------------------------
           JZ         Lg_80                ;переход, если оба бита равны 0
           JNB        ACC_1,Lg_70          ;переход, если ст бит равен 0
           JNB        ACC_0,Lg_60          ;переход, если мл бит равен 0
           MOV        EEDATA_R,EDATA4      ;если оба бита равны 0
           RET                             ;
Lg_60:     MOV        EEDATA_R,EDATA3      ;
           RET                             ;
Lg_70:     MOV        EEDATA_R,EDATA2      ;
           RET                             ;
Lg_80:     MOV        EEDATA_R,EDATA1      ;
           RET

;-------------------------------------------------------------------------
;Подпрограмма инициализации РСН.
;-------------------------------------------------------------------------
Pod_INIT_RSN:
          MOV        PORT_KNOP,#11111111b ;сделать вх все линии порта кн
          MOV        PORT_IND,#00000000b  ;сделать вых все линии порта индик
          RET

;-------------------------------------------------------------------------------
;Подпрограмма инициализации РОНов. Обнуляются все РОНы с адресами
;от NACH_ADR до KON_ADR при помощи косвенной адресации.
;-------------------------------------------------------------------------------
Pod_INIT_RON:
```

```
          MOV       R0,#NACH_ADR          ;установка начального адреса
Lk_0:     MOV       @R0,#0                ;обнуление очередного РОНа
          INC       R0                    ;переход к следующему адресу
          CJNE      R0,#KON_ADR,Lk_0      ;не достигли ли последнего адреса ?
          MOV       @R0,#0                ;обнуление последнего РОНа
          RET                             ;да, выход

;Подключение модулей опроса кнопок, вывода данных на ЖКИ (с опросом ЖКИ)
;и преобразования представления чисел
          $INCLUDE  (C:\PR_ADUC\EEPROM\knop.asm)
          $INCLUDE  (C:\PR_ADUC\EEPROM\lcd_opr.asm)
          $INCLUDE  (C:\PR_ADUC\EEPROM\preobr.asm)

;Конец исполняемого кода
          END
```

Следует заметить, что для программных записи и чтения отдельных байтов данных и, в особенности, массивов данных аппаратная организация EEPROM ADuC824 представляется несколько неудобной. Необходимость адресовать страницу памяти и конкретный байт данных внутри страницы создают определенные сложности при написании и оптимизации прикладного пользовательского программного обеспечения. В предлагаемом примере демонстрируется работа пользовательской программы, которая производит операции с EEPROM ADuC824 в более простой и привычной для разработчиков форме. Ячейка EEPROM в программе адресуется в пределах области в 256 байт восемью разрядами (одним байтом) адреса. Запись байта данных в EEPROM производит подпрограмма Pod_WR_EEPROM. Записываемый байт данных следует предварительно поместить в регистр EEDATA_R, а адрес записи – в регистр EEADR_R. Запись производится в нижнюю (младшую) часть EEPROM ADuC824 в пределах всех возможных значений EEADR_R. После осуществления записи подпрограмма производит проверку правильности записи. Проверка реализована как выполнение команды верификации данных в странице EEPROM, куда производилась запись данных. В случае ошибки верификации подпрограмма делает вывод об ошибке записи байта данных и устанавливает пользовательский флаг ошибки записи _ERR_WR_EE. Анализ состояния этого флага можно в реальном проекте возложить на основную программу. В данном примере состояние этого флага программа не анализирует. Вызываемые из Pod_WR_EEPROM подпрограммы Pod_EEADR и Pod_EEDATA_WR осуществляют при записи преобразование адреса байта, содержащегося в регистре EEADR_R соответственно в адрес страницы и в адрес байта внутри страницы. Шестибитовый адрес страницы формируется из адреса байта путем двойного сдвига содержимого EEADR_R вправо и выделения из результата младших шести бит. Эта операция эквивалентна делению содержимого EEADR_R на 4. Информацию об адресе байта внутри страницы несут два младших бита EEADR_R.

Подпрограмма Pod_RD_EEPROM производит чтение байта данных из EEPROM по адресу, который предварительно следует поместить в EEADR_R. Прочитанный байт данных возвращается в EEDATA_R. Вызываемые из

Pod_RD_EEPROM подпрограммы Pod_EEADR и Pod_EEDATA_RD осуществляют при чтении преобразование адреса байта, содержащегося в регистре EEADR_R соответственно в адрес страницы и в адрес байта внутри страницы.

При нажатии на кнопку «0» происходит последовательный циклический перебор всех возможных адресов байта (256 значений). Текущий адрес в десятичной форме (разряды сотен, десятков и единиц) индицируется на ЖКИ начиная с адреса 0 ОЗУ ЖКИ. При нажатии на кнопку «1» происходит перебор в некоторых пределах возможных значений байта данных для записи в EEPROM. ASCII-код текущего значения байта отображается на ЖКИ по адресу 64 ОЗУ ЖКИ. При нажатии на кнопку «2» производится запись выбранного байта данных в EEPROM по выбранному адресу. По окончании записи ASCII-код байта данных отображается на ЖКИ по адресу 68 ОЗУ ЖКИ. При нажатии на кнопку «3» производится чтение байта данных из EEPROM по выбранному адресу. Прочитанный байт данных отображается на ЖКИ по адресу 72 ОЗУ ЖКИ. Запись и чтение по текущему адресу можно производить в любом порядке. В случае, если чтение предшествует записи и читается ячейка EEPROM, в которую запись после программирования ADuC824 не производилась, будет прочитан код чистой ячейки – 0FFh. На ЖКИ он отобразится как черный прямоугольник – полностью закрашенное знакоместо (приложение 3).

В случае, если прикладному коду недостаточно возможности адресации двухсот пятидесяти шести ячеек EEPROM, вниманию читателей предлагается демонстрационная программа с подпрограммами, позволяющими вышеуказанным способом адресовать все имеющиеся на кристалле 640 байт EEPROM. Исходный текст этой программы находится в файле eeprom1.asm (листинг 3.12).

Листинг 3.12. Программа с адресацией 640 байт EEPROM

```
;-------------------------------------------------------------------------------
;Тестовая программа обслуживания встроенного EEPROM данных ADuC824.
;
;Данные в ЖКИ передаются по 4-битной шине, состояние ЖКИ опрашивается,
;подпрограмма опроса возвращает управление, когда ЖКИ готов к приему данных.
;
;При нажатии на кн 0 производится перебор адресов EEPROM от 0 до 639.
;При нажатии на кн 1 производится перебор кодов символов для записи в EEPROM.
;При нажатии на кн 2 производится запись в EEPROM выбранного байта данных.
;При нажатии на кн 3 производится чтение из EEPROM байта данных.
;Выбранный адрес EEPROM отображается в десятичной форме на ЖКИ начиная
;с адреса 0 ОЗУ ЖКИ.
;ASCII-код выбранного для записи в EEPROM байта отображается по адр 64 ОЗУ ЖКИ.
;ASCII-код записанного в EEPROM по текущему адресу байта отображается по адресу
;68 ОЗУ ЖКИ.
;ASCII-код считанного из EEPROM по текущему адресу байта отображается по адресу
;72 ОЗУ ЖКИ.
;-------------------------------------------------------------------------------
          $INCLUDE  (C:\ADuC\mod824)
          $INCLUDE  (C:\PR_ADUC\EEPROM1\824.inc)
;-------------------------------------------------------------------------------
;Описание битов, регистров и констант
;-------------------------------------------------------------------------------
          PORT_KNOP    EQU      P0      ;порт кнопок

          PORT_IND     EQU      P2      ;порт индикации
```

```
          _IN_KNOP0    EQU      P0_0    ;----------------------------
          _IN_KNOP1    EQU      P0_1    ;
          _IN_KNOP2    EQU      P0_2    ;  входы кнопок
          _IN_KNOP3    EQU      P0_3    ;
          _IN_KNOP4    EQU      P0_4    ;
          _IN_KNOP5    EQU      P0_5    ;
          _IN_KNOP6    EQU      P0_6    ;
          _IN_KNOP7    EQU      P0_7    ;----------------------------

          POROG_K      EQU      50      ;порог подавления дребезга кнопок
          NACH_ADR     EQU      000h    ;начальный адрес обнуления РОНов
          KON_ADR      EQU      07Fh    ;конечный адрес обнуления РОНов

          KNOP0_R      DATA     032h    ;----------------------------
          KNOP1_R      DATA     033h    ;
          KNOP2_R      DATA     034h    ; регистры, содержащие
          KNOP3_R      DATA     035h    ;
          KNOP4_R      DATA     036h    ; флаги нажатия и удержания
          KNOP5_R      DATA     037h    ;
          KNOP6_R      DATA     038h    ; каждой кнопки
          KNOP7_R      DATA     039h    ;----------------------------

          NAKOPL0_R    DATA     03Ah    ;----------------------------
          NAKOPL1_R    DATA     03Bh    ;
          NAKOPL2_R    DATA     03Ch    ; регистры накопления
          NAKOPL3_R    DATA     03Dh    ;
          NAKOPL4_R    DATA     03Eh    ; значения подавления дребезга
          NAKOPL5_R    DATA     03Fh    ;
          NAKOPL6_R    DATA     040h    ; каждой кнопки
          NAKOPL7_R    DATA     041h    ;----------------------------

          PORT_IND_0   EQU      P2_0    ;----------------------------
          PORT_IND_1   EQU      P2_1    ;
          PORT_IND_2   EQU      P2_2    ; выходы индикации
          PORT_IND_3   EQU      P2_3    ;
          PORT_IND_4   EQU      P2_4    ;
          PORT_IND_5   EQU      P2_5    ;
          PORT_IND_6   EQU      P2_6    ;
          PORT_IND_7   EQU      P2_7    ;----------------------------

          RW           EQU      PORT_IND_1 ;-------------------------
          RS           EQU      PORT_IND_2 ; линии обслуживания ЖКИ
          E            EQU      PORT_IND_3 ;-------------------------

          ADR_IND_R    DATA     030h    ;----------------------------
          COM_IND_R    DATA     030h    ; РОНы обслуживания ЖКИ
          DATA_IND_R   DATA     031h    ;----------------------------

          EEDATA_R     DATA     044h    ;РОН данных EEPROM
          EEADRH_R     DATA     045h    ;РОН ст половины адреса EEPROM
          EEADRL_R     DATA     046h    ;РОН мл половины адреса EEPROM
          DDD_R        DATA     047h    ;РОН выбора байта данных

          _ERR_WR_EE   BIT      02h     ;флаг "ошибка зап в EEPROM"

;РОНы обслуживания подпрограмм преобразования формы представления чисел
          IND_DESTIS_R DATA     053h    ;РОН десятков тысяч дес числа
          IND_TIS_R    DATA     054h    ;РОН тысяч дес числа
          IND_SOT_R    DATA     055h    ;РОН сотен дес числа
          IND_DES_R    DATA     056h    ;РОН десятков дес числа
          IND_ED_R     DATA     057h    ;РОН единиц дес числа

;Начало исполняемого кода-------------------------------------------------------
                ORG 0h
           LJMP       Lab_START

;Начало основной программы------------------------------------------------------
                ORG 100h
```

```
Lab_START: MOV        SP,#080h             ;определить указатель стека
           MOV        PLLCON,#00000000b    ;уст макс частоту ядра (12,58 МГц)
           NOP                             ;

           ACALL      Pod_INIT_RSN         ;иниц РСН
           ACALL      Pod_INIT_RON         ;иниц РОН
           ACALL      Pod_INIT_LCD         ;иниц ЖКИ
           ACALL      Pod_CLEAR_LCD        ;стирание ЖКИ

           MOV        DDD_R,#021h          ;уст нач значения байта данных

           MOV        R5,EEADRH_R          ; исх данные - адрес EEPROM
           MOV        R4,EEADRL_R          ;
           LCALL      B16BCDD              ;преобраз двоичн в упак двоичн-дес
           LCALL      BCD5BCD              ;преобраз упак двоичн-дес в неупак
           MOV        DATA_IND_R,IND_SOT_R ;-----------------------------------
           MOV        ADR_IND_R,#0         ; индикация сотен
           LCALL      Pod_PER_DAT_LCD      ;-----------------------------------
           MOV        DATA_IND_R,IND_DES_R ;-----------------------------------
           MOV        ADR_IND_R,#1         ; индикация дес
           LCALL      Pod_PER_DAT_LCD      ;-----------------------------------
           MOV        DATA_IND_R,IND_ED_R  ;-----------------------------------
           MOV        ADR_IND_R,#2         ; индикация ед
           LCALL      Pod_PER_DAT_LCD      ;-----------------------------------

;Начало основного цикла-------------------------------------------------------
La_OSN:    NOP                             ;метка возврата осн цикла

        ;Блок, предшествующий вызову подпрограммы опроса кнопки 0.
           MOV        R0,#KNOP0_R          ;
           MOV        R1,#NAKOPL0_R        ;
           ACALL      Pod_OPR_KNOP0        ;
      ;Блок, следующий за вызовом подпрограммы опроса кнопки (уст факта ее наж)
           MOV        ACC,@R0              ;
           JNB        ACC_1,La_0           ;

           CLR        ACC_1                ;кн была нажата, сброс флага нажат кн
           MOV        @R0,ACC              ;

           MOV        ACC,EEADRL_R         ;-----------------------------------
           ADD        A,#1                 ;
           JNC        La_00                ; перебор возможных адресов EEPROM
           INC        EEADRH_R             ;
La_00:     MOV        EEADRL_R,ACC         ;  от 0 до 639
           CJNE       A,#LOW(640),La_000   ;
           MOV        ACC,EEADRH_R         ;
           CJNE       A,#HIGH(640),La_000  ;
           MOV        EEADRL_R,#0          ;
           MOV        EEADRH_R,#0          ;-----------------------------------
La_000:    NOP

           MOV        R5,EEADRH_R          ; исх данные - адрес EEPROM
           MOV        R4,EEADRL_R          ;
           LCALL      B16BCDD              ;преобраз двоичн в упак двоичн-дес
           LCALL      BCD5BCD              ;преобраз упак двоичн-дес в неупак
           MOV        DATA_IND_R,IND_SOT_R ;-----------------------------------
           MOV        ADR_IND_R,#0         ; индикация сотен
           LCALL      Pod_PER_DAT_LCD      ;-----------------------------------
           MOV        DATA_IND_R,IND_DES_R ;-----------------------------------
           MOV        ADR_IND_R,#1         ; индикация дес
           LCALL      Pod_PER_DAT_LCD      ;-----------------------------------
           MOV        DATA_IND_R,IND_ED_R  ;-----------------------------------
           MOV        ADR_IND_R,#2         ; индикация ед
           LCALL      Pod_PER_DAT_LCD      ;-----------------------------------

      ;Блок, предшествующий вызову подпрограммы опроса кнопки 1.
La_0:      MOV        R0,#KNOP1_R          ;
           MOV        R1,#NAKOPL1_R        ;
```

```
            ACALL       Pod_OPR_KNOP1         ;
       ;Блок, следующий за вызовом подпрограммы опроса кнопки (уст факта ее наж)
            MOV         ACC,@R0               ;
            JNB         ACC_1,La_2            ;

            CLR         ACC_1                 ;кн была нажата, сброс флага нажат кн
            MOV         @R0,ACC               ;

            MOV         ACC,DDD_R             ;-------------------------------------
            INC         A                     ;
            CJNE        A,#06Fh,La_1          ; перебор возможных значений байта
            MOV         A,#021h               ; данных
La_1:       MOV         DDD_R,A               ;-------------------------------------

            MOV         DATA_IND_R,DDD_R      ;-------------------------------------
            MOV         ADR_IND_R,#64         ; индикация выбранного байта
            ACALL       Pod_PER_DAT_LCD       ;-------------------------------------

       ;Блок, предшествующий вызову подпрограммы опроса кнопки 2.
La_2:       MOV         R0,#KNOP2_R           ;
            MOV         R1,#NAKOPL2_R         ;
            ACALL       Pod_OPR_KNOP2         ;

       ;Блок, следующий за вызовом подпрограммы опроса кнопки (уст факта ее наж)
            MOV         ACC,@R0               ;
            JNB         ACC_1,La_3            ;

            CLR         ACC_1                 ;кн была нажата, сброс флага нажат кн
            MOV         @R0,ACC               ;

            MOV         EEDATA_R,DDD_R        ;
            ACALL       Pod_WR_EEPROM         ;запись байта в EEPROM

            MOV         DATA_IND_R,EEDATA_R   ;-------------------------------------
            MOV         ADR_IND_R,#68         ; индикация записанного байта
            ACALL       Pod_PER_DAT_LCD       ;-------------------------------------

      ;Блок, предшествующий вызову подпрограммы опроса кнопки 3.
La_3:       MOV         R0,#KNOP3_R           ;
            MOV         R1,#NAKOPL3_R         ;
            ACALL       Pod_OPR_KNOP3         ;

       ;Блок, следующий за вызовом подпрограммы опроса кнопки (уст факта ее наж)
            MOV         ACC,@R0               ;
            JNB         ACC_1,La_4            ;

            CLR         ACC_1                 ;кн была нажата, сброс флага нажат кн
            MOV         @R0,ACC               ;

            MOV         EEDATA_R,#0FFh        ;контрольная порча регистра EEDATA_R
            ACALL       Pod_RD_EEPROM         ;чтение байта из EEPROM

            MOV         DATA_IND_R,EEDATA_R   ;-------------------------------------
            MOV         ADR_IND_R,#72         ; индикация считанного байта
            ACALL       Pod_PER_DAT_LCD       ;-------------------------------------

La_4:       AJMP        La_OSN                ;идти на начало осн цикла


;Подпрограммы-----------------------------------------------------------------------

;-------------------------------------------------------------------------------
;Подпрограмма записи одного байта в EEPROM данных по адресу, предварительно
;помещенному в РОНы EEADRH_R, EEADRL_R. Байт данных следует предварительно
;поместить в РОН EEDATA_R. В случае ошибки записи (ошибки верификации)
;установится флаг _ERR_WR_EE - "ошибка записи в EEPROM". Можно адресовать
;только 640 (от 0 до  639) байт EEPROM.
;-------------------------------------------------------------------------------
```

```
Pod_WR_EEPROM:
            ACALL       Pod_EEADR             ;подпр определения адр стр EEPROM
            MOV         ECON,#01h             ;читать страницу
            ACALL       Pod_EEDATA_WR         ;подпр определения адр байта для зап
            MOV         ECON,#05h             ;стереть страницу
            MOV         ECON,#02h             ;записать страницу в EEPROM
            MOV         ECON,#04h             ;верифицировать страницу
            MOV         ACC,ECON              ;не было ли ошибки записи ?
            JZ          Lg_0                  ;нет, выход
            SETB        _ERR_WR_EE            ;да, уст флаг "ошибка зап в EEPROM"
Lg_0:       RET

;-------------------------------------------------------------------------------
;Подпрограмма определения адреса страницы EEPROM из абсолютного адреса
;-------------------------------------------------------------------------------
Pod_EEADR:
            MOV         ACC,EEADRL_R          ;---------------------------------
            RR          A                     ; Получение шести мл бит адреса
            RR          A                     ; стр EEPROM
            ANL         ACC,#00111111b        ;
            MOV         EADRL,ACC             ;---------------------------------
            MOV         ACC,EEADRH_R          ;---------------------------------
            RR          A                     ; Получение двух ст бит адреса
            RR          A                     ; стр EEPROM
            ANL         ACC,#11000000b        ;
            ORL         A,EADRL               ;---------------------------------
            MOV         EADRL,ACC             ;загрузить подготовл адр страницы
            RET

;-------------------------------------------------------------------------------
;Подпрограмма определения адреса байта (для записи) внутри страницы из мл байта
;абс адреса
;-------------------------------------------------------------------------------
Pod_EEDATA_WR:
            MOV         ACC,EEADRL_R          ; выделение двух мл бит
            ANL         ACC,#00000011b        ;---------------------------------
            JZ          Lg_50                 ;переход, если оба бита равны 0
            JNB         ACC_1,Lg_40           ;переход, если ст бит равен 0
            JNB         ACC_0,Lg_30           ;переход, если мл бит равен 0
            MOV         EDATA4,EEDATA_R       ;если оба бита равны 0
            RET                               ;
Lg_30:      MOV         EDATA3,EEDATA_R       ;
            RET                               ;
Lg_40:      MOV         EDATA2,EEDATA_R       ;
            RET                               ;
Lg_50:      MOV         EDATA1,EEDATA_R       ;
            RET

;-------------------------------------------------------------------------------
;Подпрограмма чтения одного байта из EEPROM данных по адресу, предварительно
;помещенному в РОНы EEADRH_R, EEADRL_R. Прочитанный байт данных оказывается в
;РОНе EEDATA_R. Ошибка чтения не диагностируется. Можно адресовать полько
;640 (от 0 до 639) байт EEPROM.
;-------------------------------------------------------------------------------
Pod_RD_EEPROM:
            ACALL       Pod_EEADR             ;подпр определения адр стр EEPROM
            MOV         ECON,#01h             ;читать страницу
            ACALL       Pod_EEDATA_RD         ;подпр определения адр байта для чтен
            RET

;-------------------------------------------------------------------------------
;Подпрограмма определения адреса байта (для чтения) внутри страницы из мл байта
;абс адреса
;-------------------------------------------------------------------------------
Pod_EEDATA_RD:
            MOV         ACC,EEADRL_R          ; выделение двух мл бит
            ANL         ACC,#00000011b        ;---------------------------------
            JZ          Lg_80                 ;переход, если оба бита равны 0
```

```
            JNB         ACC_1,Lg_70          ;переход, если ст бит равен 0
            JNB         ACC_0,Lg_60          ;переход, если мл бит равен 0
            MOV         EEDATA_R,EDATA4      ;если оба бита равны 0
            RET                              ;
Lg_60:      MOV         EEDATA_R,EDATA3      ;
            RET                              ;
Lg_70:      MOV         EEDATA_R,EDATA2      ;
            RET                              ;
Lg_80:      MOV         EEDATA_R,EDATA1      ;
            RET

;-------------------------------------------------------------------------
;Подпрограмма инициализации PCH.
;-------------------------------------------------------------------------
Pod_INIT_RSN:
           MOV         PORT_KNOP,#11111111b ;сделать вх все линии порта кн
           MOV         PORT_IND,#00000000b  ;сделать вых все линии порта индик
           RET

;-------------------------------------------------------------------------
;Подпрограмма инициализации РОНов. Обнуляются все РОНы с адресами
;от NACH_ADR до KON_ADR при помощи косвенной адресации.
;-------------------------------------------------------------------------
Pod_INIT_RON:
           MOV         R0,#NACH_ADR         ;установка начального адреса
Lk_0:      MOV         @R0,#0               ;обнуление очередного РОНа
           INC         R0                   ;переход к следующему адресу
           CJNE        R0,#KON_ADR,Lk_0     ;не достигли ли последнего адреса ?
           MOV         @R0,#0               ;обнуление последнего РОНа
           RET                              ;да, выход

;Подключение модулей опроса кнопок, вывода данных на ЖКИ (с опросом ЖКИ)
;и преобразования представлений чисел
           $INCLUDE   (C:\PR_ADUC\EEPROM1\knop.asm)
           $INCLUDE   (C:\PR_ADUC\EEPROM1\lcd_opr.asm)
           $INCLUDE   (C:\PR_ADUC\EEPROM1\preobr.asm)

;Конец исполняемого кода
           END
```

Работает она точно так же, как и предыдущая, но адрес ячейки данных в массиве EEPROM здесь не восьми- , а десятиразрядный. Его старшая и младшая части содержатся соответственно в регистрах EEADRH_R, EEADRL_R. (Старшая часть в регистре EEADRH_R занимает два младших разряда.) Как и в предыдущем примере, подпрограммы Pod_EEADR, Pod_EEDATA_WR и Pod_EEDATA_RD осуществляют преобразование адреса байта, содержащегося в EEADRH_R, EEADRL_R в адрес страницы и в адрес байта внутри страницы. Восьмибитовый адрес страницы формируется из десятибитового адреса байта путем двойного сдвига содержимого EEADRL_R вправо и выделения из результата младших шести бит. Затем содержимое EEADRH_R также дважды сдвигается вправо и из результата выделяются два старших бита. Получившиеся два числа логически складываются инструкцией ORL, что и дает в результате один байт адреса страницы, помещаемой подпрограммой соответственно в регистр EADRL. Следует заметить, что это, наверняка, не самый оптимальный алгоритм получения требуемого результата. Особо «продвинутые» читатели могут попытаться самостоятельно придумать что-нибудь более компактное.

Информацию об адресе байта внутри страницы, как и в предыдущем случае, несут два младших бита EEADR_R.

3.6. Использование интерфейса SPI для подключения внешних устройств

Использование последовательного периферийного интерфейса SPI ADuC824 иллюстрируется демонстрационной программой, исходный текст которой содержится в файле spi_df.asm (листинг 3.13). Обмен по шине SPI для режима ведущего ADuC824 реализован программно с использованием аппаратного модуля SPI микроконвертора.

Листинг 3.13. Использование интерфейса SPI

```
;-------------------------------------------------------------------------------
;Демонстрационная программа организации обмена между внешним устройством памяти
;AT45DB041 и ADuC824 по шине SPI. Напряжение питания не должно превышать 3,6 В.
;Данные в ЖКИ передаются по 4-битной шине, состояние ЖКИ опрашивается.
;ADuC824 является ведущим устройством SPI.
;Протокол обмена по шине SPI реализован аппаратно-программно.
;(Используется аппаратный модуль SPI ADuC824).
;При нажатии на кнопку 0 производится перебор адресов ячеек для записи данных в
;AT45DB041. Выбранное значение адреса ячейки в десятичн виде отображается на ЖКИ
;начиная с адреса 0 ОЗУ ЖКИ.
;
;При нажатии на кнопку 1 производится выбор байта данных для записи в AT45DB041.
;Выбранный байт данных в ASCII коде  выводится на ЖКИ по адресу 64 ОЗУ ЖКИ.
;
;При нажатии на кнопку 2 производится запись выбранного байта данных в AT45DB041
;по выбранному ранее адресу. Записанный байт данных выводится на ЖКИ по
;адресу 70 ОЗУ ЖКИ.
;
;При нажатии на кнопку 3 производится чтение байта данных из AT45DB041 по
;выбранному ранее адресу. Прочитанный байт данных в ASCII коде отображается на
;ЖКИ по адресу 72 ОЗУ ЖКИ.
;-------------------------------------------------------------------------------
          $INCLUDE  (C:\ADuC\mod824)
          $INCLUDE  (C:\PR_ADUC\SPI_DF\824.inc)
;-------------------------------------------------------------------------
;Описание битов, регистров и констант
;-------------------------------------------------------------------------
;Порты и линии ввода-вывода
          PORT_KNOP    EQU      P0      ;порт кнопок

          PORT_IND     EQU      P2      ;порт индикации

          _IN_KNOP0    EQU      P0_0    ;-----------------------------
          _IN_KNOP1    EQU      P0_1    ;
          _IN_KNOP2    EQU      P0_2    ;  входы кнопок
          _IN_KNOP3    EQU      P0_3    ;
          _IN_KNOP4    EQU      P0_4    ;
          _IN_KNOP5    EQU      P0_5    ;
          _IN_KNOP6    EQU      P0_6    ;
          _IN_KNOP7    EQU      P0_7    ;-----------------------------

          PORT_IND_0   EQU      P2_0    ;-----------------------------
          PORT_IND_1   EQU      P2_1    ;
          PORT_IND_2   EQU      P2_2    ; выходы индикации
          PORT_IND_3   EQU      P2_3    ;
          PORT_IND_4   EQU      P2_4    ;
          PORT_IND_5   EQU      P2_5    ;
          PORT_IND_6   EQU      P2_6    ;
          PORT_IND_7   EQU      P2_7    ;-----------------------------

          RW           EQU      PORT_IND_1 ;--------------------------
```

```
          RS            EQU      PORT_IND_2 ; линии управления ЖКИ
          E             EQU      PORT_IND_3 ;--------------------------

          SC            EQU      P3_4    ;выход выбора устройства SPI

;РОНы обслуживания ЖКИ и кнопок
          ADR_IND_R     DATA     030h    ;-----------------------------
          COM_IND_R     DATA     030h    ; РОНы обслуживания ЖКИ
          DATA_IND_R    DATA     031h    ;-----------------------------

          KNOP0_R       DATA     032h    ;-----------------------------
          KNOP1_R       DATA     033h    ;
          KNOP2_R       DATA     034h    ; регистры, содержащие
          KNOP3_R       DATA     035h    ;
          KNOP4_R       DATA     036h    ; флаги нажатия и удержания
          KNOP5_R       DATA     037h    ;
          KNOP6_R       DATA     038h    ; каждой кнопки
          KNOP7_R       DATA     039h    ;-----------------------------

          NAKOPL0_R     DATA     03Ah    ;-----------------------------
          NAKOPL1_R     DATA     03Bh    ;
          NAKOPL2_R     DATA     03Ch    ; регистры накопления
          NAKOPL3_R     DATA     03Dh    ;
          NAKOPL4_R     DATA     03Eh    ; значения подавления дребезга
          NAKOPL5_R     DATA     03Fh    ;
          NAKOPL6_R     DATA     040h    ; каждой кнопки
          NAKOPL7_R     DATA     041h    ;-----------------------------

;РОНы обслуживания подпрограмм преобразования формы представления чисел
          IND_MILL_R    DATA     04Eh    ;РОН миллиардов дес числа
          IND_SOTMIL_R  DATA     04Fh    ;РОН сотен миллионов дес числа
          IND_DESMIL_R  DATA     050h    ;РОН десятков миллионов дес числа
          IND_MIL_R     DATA     051h    ;РОН миллионов дес числа
          IND_SOTTIS_R  DATA     052h    ;РОН сотен тысяч дес числа
          IND_DESTIS_R  DATA     053h    ;РОН десятков тысяч дес числа
          IND_TIS_R     DATA     054h    ;РОН тысяч дес числа
          IND_SOT_R     DATA     055h    ;РОН сотен дес числа
          IND_DES_R     DATA     056h    ;РОН десятков дес числа
          IND_ED_R      DATA     057h    ;РОН единиц дес числа

          BYTE_0_R      DATA     05Bh    ;байт 0 упакованного двоичн-дес числа
          BYTE_1_R      DATA     05Ch    ;байт 1 упакованного двоичн-дес числа
          BYTE_2_R      DATA     05Dh    ;байт 2 упакованного двоичн-дес числа
          BYTE_3_R      DATA     05Eh    ;байт 3 упакованного двоичн-дес числа
          BYTE_4_R      DATA     05Fh    ;байт 4 упакованного двоичн-дес числа

;РОНы обслуживания обмена с AT45DB041 по интерфейсу SPI
          DATASPI_R     DATA     064h    ;РОН данных чтения и записи SPI DF
          ADRF_L_R      DATA     065h    ;РОН мл байта абсолютного адр ячейки DF
          ADRF_M_R      DATA     066h    ;РОН ср байта абсолютного адр ячейки DF
          ADRF_H_R      DATA     067h    ;РОН ст байта абсолютного адр ячейки DF

;Константы
          NACH_ADR      EQU      000h    ;начальный адрес обнуления РОНов
          KON_ADR       EQU      07Fh    ;конечный адрес обнуления РОНов
          POROG_K       EQU      50      ;порог подавления дребезга кнопок
;Начало исполняемого кода-------------------------------------------------
                ORG 0h
           AJMP       Lab_START           ;идти на начало осн программы

;Начало осн программы-----------------------------------------------------
                ORG 100h
Lab_START: MOV        SP,#080h            ;определить указатель стека
           MOV        PLLCON,#00000000b   ;уст макс частоту ядра (12,58 МГц)
           NOP                            ;
           LCALL      Pod_INIT_RSN        ;иниц РСН
           LCALL      Pod_INIT_RON        ;иниц РОН
```

```
            MOV         R5,#021h        ;задание нач значения индицируемого байта 1

            LCALL       Pod_INIT_LCD          ;иниц ЖКИ

            LCALL       Pod_CLEAR_LCD         ;стирание ЖКИ

            MOV         R0,#BYTE_0_R          ;
            MOV         R1,#ADRF_L_R          ;
            MOV         ADRF_H_R+1,#0        ; обнуление ст незначащего РОНа
            LCALL       B32BCD               ;преобр адреса из двоичн в двоичн-дес

            MOV         R0,#BYTE_0_R          ;
            MOV         R1,#IND_MILL_R        ;
            LCALL       BCD10BCD             ;преобр адреса из уп дв-дес в неуп дес

            MOV         R0,#IND_MILL_R        ; индицировать с РОНа IND_MILL_R
            MOV         R1,#0                 ; индицировать с адр 0 ЖКИ
            LCALL       Pod_IND_10ZN          ;индикация абсолютного адреса

;Начало основного цикла-------------------------------------------------------
La_OSN:     NOP                               ;метка возврата в осн цикле

;Блок, предшествующий вызову подпрограммы опроса кнопки 0.
            MOV         R0,#KNOP0_R           ;
            MOV         R1,#NAKOPL0_R         ;
            LCALL       Pod_OPR_KNOP0         ;
;Блок, следующий за вызовом подпрограммы опроса кнопки (уст факта ее наж)
            MOV         ACC,@R0               ;
            JNB         ACC_1,La_100          ;

            CLR         ACC_1                 ;кн была нажата, сброс флага нажат кн
            MOV         @R0,ACC               ;

            MOV         ACC,ADRF_M_R          ;---------------------------------
            ADD         A,#32                 ; Блок перебора адресов внутри
            JC          La_10                 ; адресного пространства
            MOV         ADRF_M_R,ACC          ; с шагом (32 × 256) = 8192
            SJMP        La_50                 ;
La_10:      MOV         ADRF_M_R,ACC          ;
            INC         ADRF_H_R              ;
            MOV         ACC,ADRF_H_R          ;
            CJNE        A,#00001000b,La_50    ;
            MOV         ADRF_M_R,#0           ;
            MOV         ADRF_H_R,#0           ;
La_50:      NOP                               ;---------------------------------

            MOV         R0,#BYTE_0_R          ;
            MOV         R1,#ADRF_L_R          ;
            MOV         ADRF_H_R+1,#0         ; обнуление ст незначащего РОНа
            LCALL       B32BCD                ;преобр адреса из двоичн в двоичн-дес

            MOV         R0,#BYTE_0_R          ;
            MOV         R1,#IND_MILL_R        ;
            LCALL       BCD10BCD             ;преобр адреса из уп дв-дес в неуп дес

            MOV         R0,#IND_MILL_R        ; индицировать с РОНа IND_MILL_R
            MOV         R1,#0                 ; индицировать с адр 0 ЖКИ
            LCALL       Pod_IND_10ZN          ;индикация абсолютного адреса

;Блок, предшествующий вызову подпрограммы опроса кнопки 1.
La_100:     MOV         R0,#KNOP1_R           ;
            MOV         R1,#NAKOPL1_R         ;
            LCALL       Pod_OPR_KNOP1         ;
;Блок, следующий за вызовом подпрограммы опроса кнопки (уст факта ее наж)
            MOV         ACC,@R0               ;
            JNB         ACC_1,La_200          ;

            CLR         ACC_1                 ;кн была нажата, сброс флага нажат кн
```

```
            MOV         @R0,ACC             ;

            INC         R5                  ;-----------------------------------
            CJNE        R5,#06Fh,La_00      ; перебор возможн знач индицир байта
            MOV         R5,#021h            ;-----------------------------------

La_00:      MOV         DATA_IND_R,R5       ;-----------------------------------
            MOV         ADR_IND_R,#64       ; индикация  байта (символа)
            ACALL       Pod_PER_DAT_LCD     ;-----------------------------------

;Блок, предшествующий вызову подпрограммы опроса кнопки 2.
La_200:     MOV         R0,#KNOP2_R         ;
            MOV         R1,#NAKOPL2_R       ;
            LCALL       Pod_OPR_KNOP2       ;
;Блок, следующий за вызовом подпрограммы опроса кнопки (уст факта ее наж)
            MOV         ACC,@R0             ;
            JNB         ACC_1,La_300        ;

            CLR         ACC_1               ;кн была нажата, сброс флага нажат кн
            MOV         @R0,ACC             ;

            MOV         DATASPI_R,R5        ;подготовка данных для зап в DF SPI
            LCALL       Pod_WRITE_BYTE      ;запись в DF SPI

            MOV         DATA_IND_R,DATASPI_R ;-----------------------------------
            MOV         ADR_IND_R,#70      ; индикация записанного байта (символа)
            ACALL       Pod_PER_DAT_LCD     ;-----------------------------------

;Блок, предшествующий вызову подпрограммы опроса кнопки 3.
La_300:     MOV         R0,#KNOP3_R         ;
            MOV         R1,#NAKOPL3_R       ;
            LCALL       Pod_OPR_KNOP3       ;
;Блок, следующий за вызовом подпрограммы опроса кнопки (уст факта ее наж)
            MOV         ACC,@R0             ;
            JNB         ACC_1,La_400        ;

            CLR         ACC_1               ;кн была нажата, сброс флага нажат кн
            MOV         @R0,ACC             ;

            MOV         DATASPI_R,#0        ;контрольная порча РОНа данных SPI
            LCALL       Pod_READ_BYTE       ;чтение из DF SPI

            MOV         DATA_IND_R,DATASPI_R ;-----------------------------------
            MOV         ADR_IND_R,#72     ; индикация прочитанного байта (символа)
            ACALL       Pod_PER_DAT_LCD     ;-----------------------------------

La_400:     LJMP        La_OSN              ;закрыть основной цикл

;Подпрограммы-------------------------------------------------------------------

;--------------------------------------------------------------------
;Подпрограммы байтового обмена с DataFlash AT45D041.
;ADRF_H_R, ADRF_M_R, ADRF_L_R - РОНы хранения абсолютного адреса
;записи/чтения байта в массиве памяти (ст  ср  мл).
;DATA_R - РОН хранения данных записи/чтения.
;Адресные регистры не портятся.
;
;         Структура абсолютного адреса байта в массиве:
;
;      YYYYYYYY ZZZZZZZZ HHHHHHHH     - 3  байта адреса (ст  ср  мл)
; СЗР                              МЗР
;      xxxx                           - 4 незначащих бита
;          XXXX XXXXXXX               - 11 битов адреса стр (0...2047)
;                       X XXXXXXXX    - 9 битов адреса байта в стр (0...264)
;
;      0000XXXX XXXXXXX0 XXXXXXXX     - рекомендуемая маска абс адреса
;                                    (в случае использования этой маски в
;                                    странице доступно только 256 байт из 264)
```

```
;---------------------------------------------------------------------
;Подпрограмма чтения байта из массива памяти по произвольному адресу.
;Прочитанный байт возвращается в РОНе DATA_R.
;Абсолютный адрес байта в массиве следует предварительно поместить
;в РОНы ADRF_H_R, ADRF_M_R, ADRF_L_R.
Pod_READ_BYTE:
        LCALL      Pod_READ_SR          ;чтение рег статуса DataFlash
        CLR        SC                   ;выбор устройства
         MOV        ACC,#053h           ;
        LCALL      Pod_TX_BYTE          ;передача КОПа перемещ стр в буфер 1
        LCALL      Pod_TX_ADRF          ;передача абс адреса байта в массиве
        SETB       SC                   ;запрет выбора устройсва
        LCALL      Pod_READ_SR          ;чтение рег статуса DataFlash
        CLR        SC                   ;выбор устройства
        MOV        ACC,#0D4h            ;
        LCALL      Pod_TX_BYTE          ;передача КОПа чтения буфера 1
         LCALL      Pod_TX_ADRF         ;передача абс адреса байта в массиве
         LCALL      Pod_TX_BYTE         ;передача незнач байта
         LCALL      Pod_TX_BYTE         ;прием байта данных из массива
         MOV        DATASPI_R,ACC       ;сохранение принятого байта
        SETB       SC                   ;запрет выбора устройсва
        RET

;Подпрограмма чтения регистра статуса устройства памяти DataFlahs.
;Содержимое регистра статуса возвращается в ACC.
Pod_READ_SR:
        CLR        SC                   ;выбор устройства
        MOV        ACC,#0D7h            ;
        LCALL      Pod_TX_BYTE          ;передача КОПа чтения рег статуса
Re_0:   MOV        ACC,#0FFh            ;
        LCALL      Pod_TX_BYTE          ;чтение регистра статуса
        JNB        ACC_7,Re_0           ;свободно ли устройство
        SETB       SC                   ;запрет выбора устройства
        RET

;Подпрограмма передачи трех байт абсолютного адреса байта в массиве памяти.
;Передача начинается со старшего байта адреса.
Pod_TX_ADRF:
        MOV        ACC,ADRF_H_R         ;передать ст байт адр
        LCALL      Pod_TX_BYTE          ;передача байта
        MOV        ACC,ADRF_M_R         ;передать ср байт адр
        LCALL      Pod_TX_BYTE          ;передача байта
        MOV        ACC,ADRF_L_R         ;передать мл байт адр
        LCALL      Pod_TX_BYTE          ;передача байта
        RET

;Подпрограмма записи байта в массив памяти по произвольноиу адресу.
;Записываемый байт следует предварительно поместить в РОН DATASPI_R.
;Абсолютный адрес байта следует предварительно поместить
;в РОНы ADRF_H_R, ADRF_M_R, ADRF_L_R.
Pod_WRITE_BYTE:
        LCALL      Pod_READ_SR          ;чтение рег статуса DataFlash
        CLR        SC                   ;выбор устройства
         MOV        ACC,#084h           ;
         LCALL      Pod_TX_BYTE         ;передача КОПа записи в буфер 1
         LCALL      Pod_TX_ADRF         ;передача абс адреса байта в массиве
        MOV        ACC,DATASPI_R        ;
        LCALL      Pod_TX_BYTE          ;запись байта данных
        SETB       SC                   ;запрет выбора устройства
        LCALL      Pod_READ_SR          ;чтение рег статуса DataFlash
        CLR        SC                   ;выбор устройства
        MOV        ACC,#083h            ;
         LCALL      Pod_TX_BYTE         ;передача КОПа записи буфера 1 в стр
        LCALL      Pod_TX_ADRF          ;передача абс адреса байта в массиве
        SETB       SC                   ;запрет выбора устройства
        RET

;Подпрограмма перед в DataFlash по SPI одного байта, который предварит следует
```

```
;поместить в ACC. Одновременно происходит прием байта из DataFlash. Принятый
;байт возвращается в ACC.
Pod_TX_BYTE:
        CLR         ISPI                ;сбросить флаг оконч перед/прм по SPI
        MOV         SPIDAT,ACC          ;начать передачу байта по SPI
        JNB         ISPI,$              ;ожидать окончания передачи
        MOV         ACC,SPIDAT          ;сохранить принятый байт
        RET

;-------------------------------------------------------------------------------
;Подпрограмма инициализации PCH.
;-------------------------------------------------------------------------------
Pod_INIT_RSN:
          MOV       PORT_KNOP,#11111111b ;сделать вх все линии порта кн
          MOV       PORT_IND,#00000000b  ;сделать вых все линии порта индик

;Настройка модуля SPI
          MOV       SPICON,#00110000b    ;SPI разреш, SPI ведущ, CPOL=0, CPHA=0
                                         ;(mode SPI - 0), скорость - F/2
          SETB      SC                   ;запрет выбора ведомого устройства SPI
          RET

;-------------------------------------------------------------------------------
;Подпрограмма инициализации РОНов. Обнуляются все РОНы с адресами
;от NACH_ADR до KON_ADR при помощи косвенной адресации.
;-------------------------------------------------------------------------------
Pod_INIT_RON:
          MOV       R0,#NACH_ADR         ;установка начального адреса
Lk_0:     MOV       @R0,#0               ;обнуление очередного РОНа
          INC       R0                   ;переход к следующему адресу
          CJNE      R0,#KON_ADR,Lk_0     ;не достигли ли последнего адреса ?
          MOV       @R0,#0               ;обнуление последнего РОНа
          RET                            ;да, выход

;-------------------------------------------------------------------------------
;Подпрограмма вывода на ЖКИ содержимого буфера из 10 РОНов в виде 10 знакомест.
;R0 должен указывать на РОН в буфере, содержимое которого индицируется первым.
;R1 должен содержать нач адрес знакоместа в ОЗУ ЖКИ.
;-------------------------------------------------------------------------------
Pod_IND_10ZN:
           MOV      R2,#0                ;нач уст ст цикла
Ii_0:      MOV      DATA_IND_R,@R0       ;
           MOV      ADR_IND_R,R1         ;
           LCALL    Pod_PER_DAT_LCD      ; индикация очередного символа
           INC      R0                   ;
           INC      R1                   ;
           INC      R2                   ;
           CJNE     R2,#10,Ii_0          ;
           RET

;Подключение модулей опроса кнопок, вывода данных на ЖКИ (с опросом ЖКИ)
;и преобразования представления чисел
          $INCLUDE  (C:\PR_ADUC\SPI_DF\knop.asm)
          $INCLUDE  (C:\PR_ADUC\SPI_DF\lcd_opr.asm)
          $INCLUDE  (C:\PR_ADUC\SPI_DF\preobr.asm)

;Конец исполняемого кода
          END
```

Для примера, в качестве ведомого устройства SPI используется микросхема памяти DataFlash AD45DB041B фирмы Atmel. Подробное описание микросхемы [13] доступно на сайте http://www.atmel.com. Она представляет собой высокоскоростное устройство энергонезависимой памяти с последовательным доступом и низковольтным питанием. Применение такого устройства для хра-

нения данных, например, в целях создания архива при их сборе, может представлять определенный интерес в практических конструкциях на базе ADuC824. Микросхема позволяет считывать и записывать 8-битные данные и организована как массив памяти, состоящий из 2048 страниц, по 264 байта на каждой странице. Страницы объединены в блоки, а блоки – в сектора. Блок-схема микросхемы приведена на рис. 3.4, а таблица, отражающая архитектурное построение памяти AD45DB041B, показана на рис. 3.5. Микросхема, кроме собственно массива памяти, содержит два SRAM-буфера размером каждый по 264 байта, служащих для промежуточного хранения считываемых и записываемых данных.

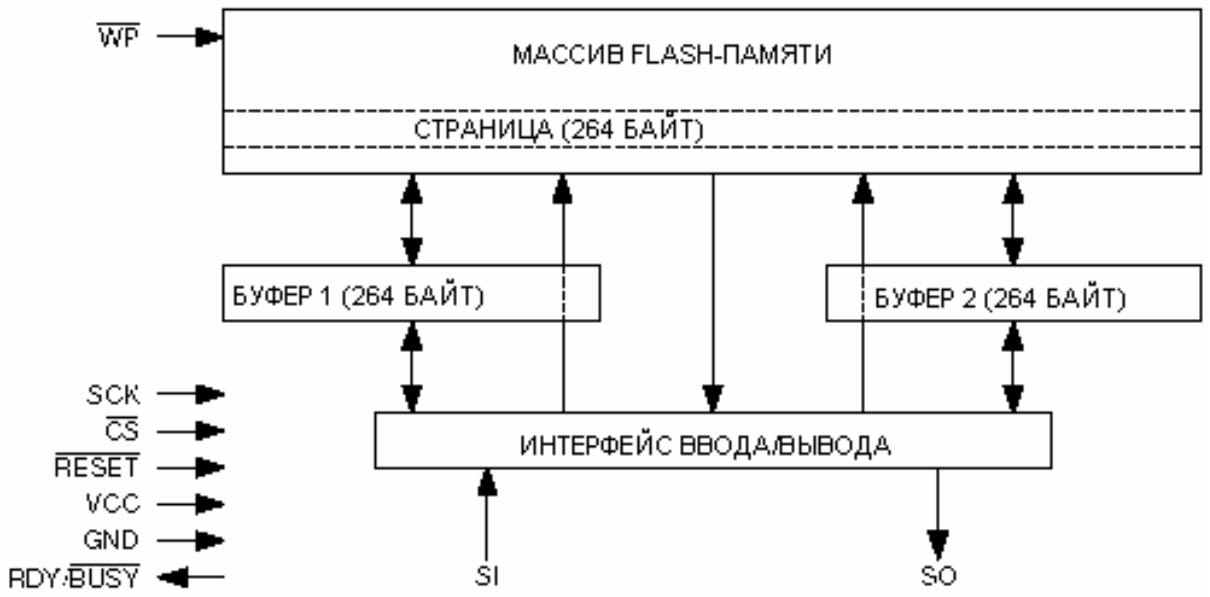


Рис. 3.4. Устройство микросхемы DataFlash

Обмен данными с ведущим SPI-устройством производится при участии внутреннего регистра статуса DataFlash, по содержимому которого ведущее устройство может судить о готовности AD45DB041B принимать команды и данные. Формат регистра статуса показан на рис. 3.6. Пять старших битов регистра содержат информацию устройства, а три младших зарезервированы и могут иметь неопределенные значения. Если бит 7 прочитанного регистра статуса равен единице, то устройство не занято и готово принимать следующую команду.

В противном случае ведущее устройство должно циклически производить операцию чтения регистра статуса, ожидая, когда установится бит 7. Результат сравнения текущей страницы памяти с буфером отражен битом 6 регистра состояния. Если он равен нулю, то данные в памяти страницы совпадают с данными в буфере. Биты 3, 4, 5 регистра состояния кодируют емкость памяти микросхемы. Для AD45DB041B они имеют значения соответственно 1, 1, 0.

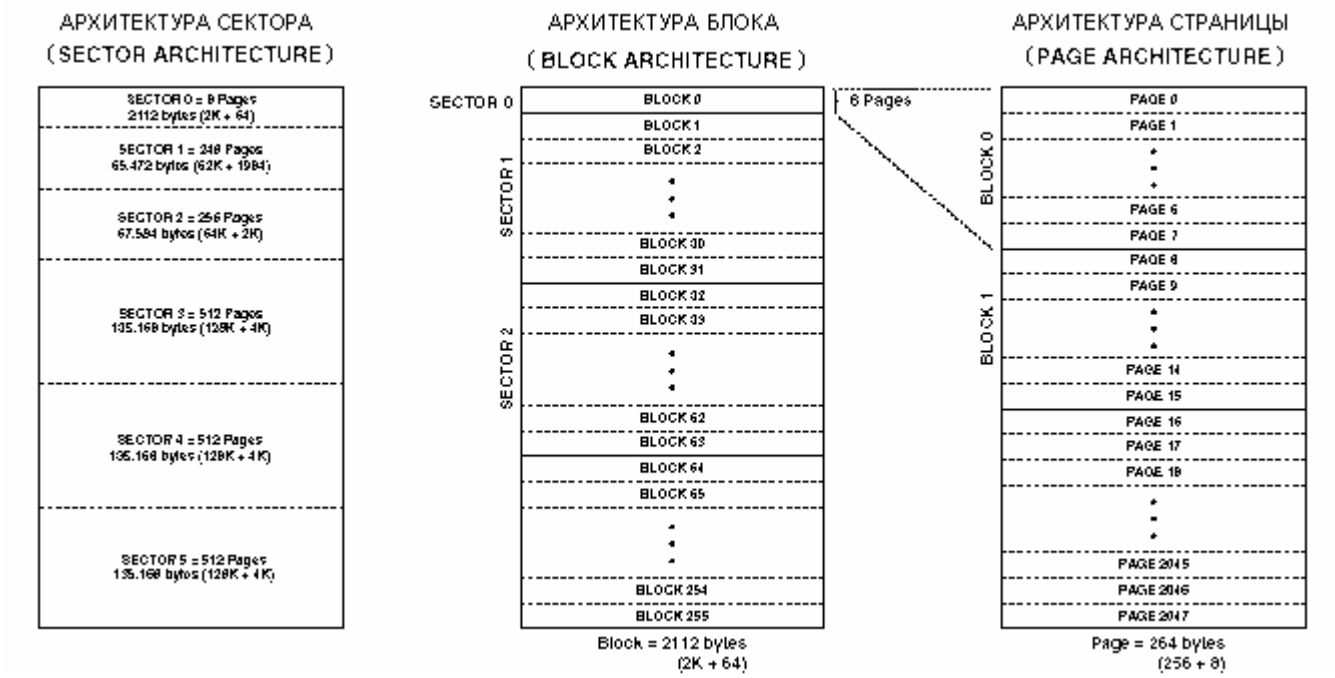


Рис. 3.5. Структура памяти микросхемы DataFlash

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
RDY/$\overline{\text{BUSY}}$	COMP	0	1	1	X	X	X

Рис. 3.6. Формат регистра статуса микросхемы DataFlash

AD45DB041B поддерживает режимы SPI 0 и 3 и имеет сложную двухуровневую систему адресации данных (страничную и внутристраничную) и кодирования команд (операций). Микросхема поддерживает следующие основные операции: последовательное чтение всего массива памяти, чтение страницы памяти, чтение регистра статуса, программирование страницы содержимым буфера с предварительным стиранием и без него, стирание страницы, стирание блока, копирование страницы в буфер, сравнение содержимого страницы с содержимым буфера, автоперезапись страницы (копирование ее самой в себя через буфер).

Операции, поддерживаемые AD45DB041B, сведены в табл. 3.1, а детализация битов командно-адресных последовательностей показана в табл. 3.2.

Таблица 3.1

Операции, поддерживаемые микросхемой AT45DB041B

Операция (команда)	**Режим SCK**	**Код операции**
Команды чтения		
Чтение непрерывного массива	Неактивный уровень сигнала синхронизации низкий или высокий Режим SPI 0 или 3	68h E8h
Чтение страницы памяти	Неактивный уровень сигнала синхронизации низкий или высокий Режим SPI 0 или 3	52h D2h
Чтение буфера 1	Неактивный уровень сигнала синхронизации низкий или высокий Режим SPI 0 или 3	54h D4h
Чтение буфера 2	Неактивный уровень сигнала синхронизации низкий или высокий Режим SPI 0 или 3	56h D6h
Чтение регистра статуса	Неактивный уровень сигнала синхронизации низкий или высокий Режим SPI 0 или 3	57h D7h
Команды стирания и программирования		
Запись в буфер 1	Любой[1]	84h
Запись в буфер 2	Любой[1]	87h
Программирование содержимым буфера 1 страницы памяти с предварительным стиранием	Любой[1]	83h
Программирование содержимым буфера 2 страницы памяти с предварительным стиранием	Любой[1]	86h
Программирование содержимым буфера 1 страницы памяти без предварительного стирания	Любой[1]	88h
Программирование содержимым буфера 2 страницы памяти без предварительного стирания	Любой[1]	89h
Стирание страницы	Любой[1]	81h
Стирание блока	Любой[1]	50h

Окончание табл. 3.1

Операция (команда)	Режим SCK	Код операции
Программирование страницы памяти через буфер 1	Любой[1]	82h
Программирование страницы памяти через буфер 2	Любой[1]	85h
Дополнительные команды		
Перемещение страницы памяти в буфер 1	Любой[1]	53h
Перемещение страницы памяти в буфер 2	Любой[1]	55h
Сравнение страницы памяти с буфером 1	Любой[1]	60h
Сравнение страницы памяти с буфером 2	Любой[1]	61h
Автоперезапись страницы через буфер 1	Любой[1]	58h
Автоперезапись страницы через буфер 2	Любой[1]	59h

[1] Режим SCK «Любой» означает любой из четырех возможных режимов работы: неактивный уровень сигнала синхронизации низкий, неактивный уровень сигнала синхронизации высокий, режим SPI 0, режим SPI 3.

Таблица 3.2

Детализация битов командных последовательностей операций, поддерживаемых AT45DB041B

Код операции	Код операции	Адресные биты								Адресные биты								Адресные биты								Доп. незначащие биты
		Резервир.	Резервир.	Резервир.	Резервир.	PA10	PA9	PA8	PA7	PA6	PA5	PA4	PA3	PA2	PA1	PA0	BA8	BA7	BA6	BA5	BA4	BA3	BA2	BA1	BA0	
50H	0 1 0 1 0 0 0 0	r	r	r	r	P	P	P	P	P	P	P	P	x	x	x	x	x	x	x	x	x	x	x	x	—
52H	0 1 0 1 0 0 1 0	r	r	r	r	P	P	P	P	P	P	P	P	P	P	P	B	B	B	B	B	B	B	B	B	4 байта
53H	0 1 0 1 0 0 1 1	r	r	r	r	P	P	P	P	P	P	P	P	P	P	P	x	x	x	x	x	x	x	x	x	—
54H	0 1 0 1 0 1 0 0	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	B	B	B	B	B	B	B	B	B	1 байт
55H	0 1 0 1 0 1 0 1	r	r	r	r	P	P	P	P	P	P	P	P	P	P	P	x	x	x	x	x	x	x	x	x	—
56H	0 1 0 1 0 1 1 0	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	B	B	B	B	B	B	B	B	B	1 байт
57H	0 1 0 1 0 1 1 1					—							—								—					—
58H	0 1 0 1 1 0 0 0	r	r	r	r	P	P	P	P	P	P	P	P	P	P	P	x	x	x	x	x	x	x	x	x	—
59H	0 1 0 1 1 0 0 1	r	r	r	r	P	P	P	P	P	P	P	P	P	P	P	x	x	x	x	x	x	x	x	x	—
60H	0 1 1 0 0 0 0 0	r	r	r	r	P	P	P	P	P	P	P	P	P	P	P	x	x	x	x	x	x	x	x	x	—
61H	0 1 1 0 0 0 0 1	r	r	r	r	P	P	P	P	P	P	P	P	P	P	P	x	x	x	x	x	x	x	x	x	—
68H	0 1 1 0 1 0 0 0	r	r	r	r	P	P	P	P	P	P	P	P	P	P	P	B	B	B	B	B	B	B	B	B	4 байта
81H	1 0 0 0 0 0 0 1	r	r	r	r	P	P	P	P	P	P	P	P	P	P	P	x	x	x	x	x	x	x	x	x	—
82H	1 0 0 0 0 0 1 0	r	r	r	r	P	P	P	P	P	P	P	P	P	P	P	B	B	B	B	B	B	B	B	B	—
83H	1 0 0 0 0 0 1 1	r	r	r	r	P	P	P	P	P	P	P	P	P	P	P	x	x	x	x	x	x	x	x	x	—
84H	1 0 0 0 0 1 0 0	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	B	B	B	B	B	B	B	B	B	—
85H	1 0 0 0 0 1 0 1	r	r	r	r	P	P	P	P	P	P	P	P	P	P	P	B	B	B	B	B	B	B	B	B	—
86H	1 0 0 0 0 1 1 0	r	r	r	r	P	P	P	P	P	P	P	P	P	P	P	x	x	x	x	x	x	x	x	x	—
87H	1 0 0 0 0 1 1 1	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	B	B	B	B	B	B	B	B	B	—
88H	1 0 0 0 1 0 0 0	r	r	r	r	P	P	P	P	P	P	P	P	P	P	P	x	x	x	x	x	x	x	x	x	—
89H	1 0 0 0 1 0 0 1	r	r	r	r	P	P	P	P	P	P	P	P	P	P	P	x	x	x	x	x	x	x	x	x	—
D2H	1 1 0 1 0 0 1 0	r	r	r	r	P	P	P	P	P	P	P	P	P	P	P	B	B	B	B	B	B	B	B	B	4 байта
D4H	1 1 0 1 0 1 0 0	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	B	B	B	B	B	B	B	B	B	1 байт
D6H	1 1 0 1 0 1 1 0	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	B	B	B	B	B	B	B	B	B	1 байт
D7H	1 1 0 1 0 1 1 1					—							—								—					—
E8H	1 1 1 0 1 0 0 0	r	r	r	r	P	P	P	P	P	P	P	P	P	P	P	B	B	B	B	B	B	B	B	B	4 байта

r – зарезервированные биты,
x – незначащие биты,
P – биты адресации страницы,
B – биты адресации байта/буфера.

Определение состояния текущей занятости DataFlash ведущим устройством может производиться программно (путем чтения регистра статуса), как это сделано в предлагаемой здесь программе, или аппаратно (путем чтения ведущим устройством уровня на ножке RDY/BUSY DataFlash). Блочные диаграммы программирования и чтения страницы памяти AD45DB041 показаны соответственно на рис. 3.7 и рис. 3.8.

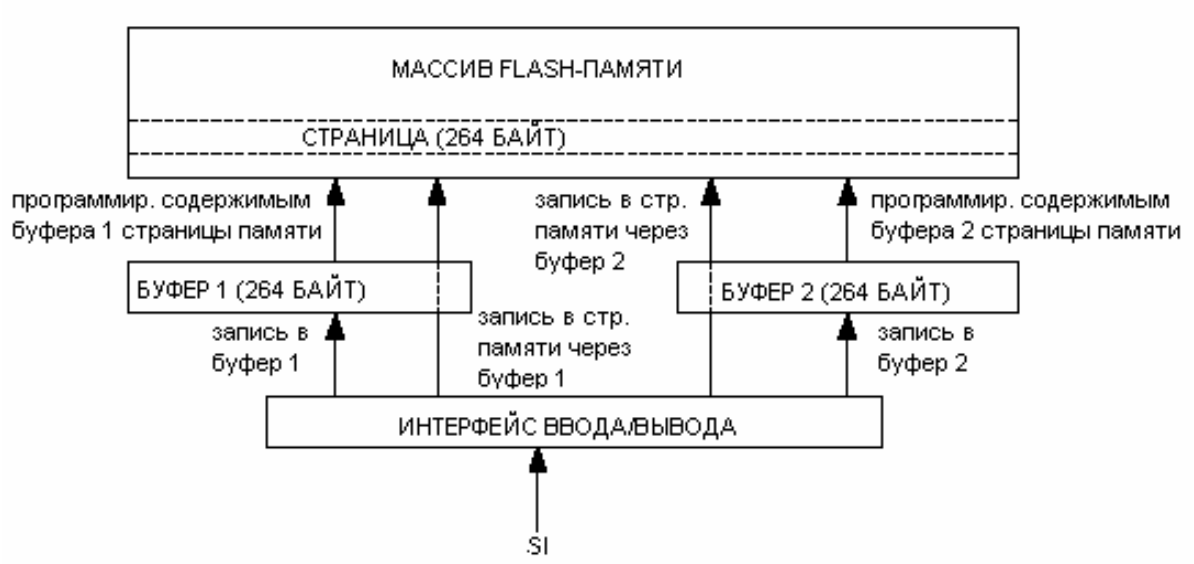


Рис. 3.7. Запись станицы в DataFlash

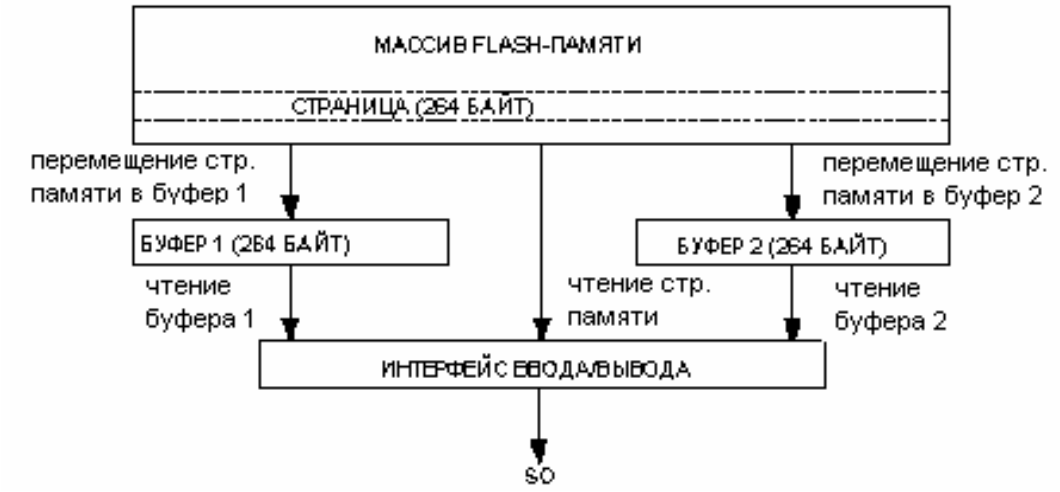


Рис. 3.8. Чтение страницы из DataFlash

Блок-схема алгоритма, рекомендуемого фирмой-производителем для произвольной модификации данных, приведена на рис. 3.9.

В данном примере демонстрируется работа набора подпрограмм, позволяющих производить запись и чтение одного байта данных AD45DB041B по произвольному адресу с использованием так называемого абсолютного адреса байта в массиве. Абсолютный адрес предварительно помещается пользовательской программой в регистры ОЗУ микроконтроллера с именами ADRF_L_R, ADRF_M_R, ADRF_H_R (младший, средний, старший), а байт данных, записываемый пользовательской программой в DataFlash, предварительно следует поместить в регистр ОЗУ DATASPI_R. В этом же регистре возвращается прочитанный из DataFlash байт данных.

Рис. 3.9. Алгоритм модификации данных в DataFlash

Подпрограмма Pod_READ_BYTE реализует чтение байта данных из DataFlash по произвольному абсолютному адресу, подпрограмма Pod_WRITE_BYTE – запись байта данных в DataFlash по произвольному абсолютному адресу. На область значений абсолютного адреса накладываются следующие ограничения: значащие биты не должны размещаться в старшей тетраде ADRF_H_R, так как этого не позволяет общая емкость AD45DB041, а младший бит ADRF_M_R должен быть всегда равен 0. Последнее ограничение связано с тем, что байт внутри страницы DataFlash адресуется девятью разрядами, но в странице содержится не 512, а только 264 байта, то есть, существует некая адресуемая область внутри страницы, в которой отсутствуют физические ячейки памяти. Чтобы исключить возможные попытки записи и чтения данных по этим адресам, условно считаем, что в странице нам доступно только 256 байт данных, а не 264, и ограничиваем разрядность адреса страницы только восемью разрядами.

Принципиальная схема макета, который необходимо собрать для экспериментов с DataFlash и ADuC824, приведена на рис. 3.10. Напряжение питания всей схемы не должно превышать +3,6 В (верхнего предела питающего напряжения AD45DB041B).

В качестве линии /SS интерфейса ведущего SPI можно использовать любую линию вывода МК ADuC824. В данном примере программа использует линию P3.4. Нумерация выводов DataFlash указана для исполнения в корпусе 8-SOIC, а в скобках – для корпуса 28-SOIC.

Программа spi_df.asm устанавливает значение ADRF_L_R равным 0, а значения ADRF_M_R и ADRF_H_R позволяет пользователю выбрать самостоятельно путем нажатий на кнопку «0» в пределах адресного пространства AD45DB041, но не подряд, а с некоторым фиксированным шагом (в программе он равен 32). Получившийся абсолютный адрес в десятичном представлении индицируется в верхней строке ЖКИ. Программа также позволяет пользователю выбрать значение байта данных для записи в DataFlash путем нажатий на кнопку «1». ASCII-код последнего выбранного байта индицируется в нижней строке ЖКИ по адресу 64 ОЗУ ЖКИ. Перебор возможных значений байта данных производится кнопкой «1» по кольцу в некоторой (заданной программно) области значений. При нажатии на кнопку «2» производится запись выбранного значения байта данных в DataFlash. По окончании записи ASCII-код записанного байта отображается в нижней строке ЖКИ по адресу 70, однако об успешности записи программа самостоятельно судить не может, так как по определению интерфейса SPI в нем отсутствуют какие-либо аппаратные средства диагностики ошибки ведомого устройства.

Об успешности записи может судить сам пользователь. При нажатии на кнопку «3» производится чтение байта данных из DataFlash по текущему адресу. ASCII-код прочитанного байта отображается в нижней строке ЖКИ по адресу 72. При правильно собранной схеме, исправных микросхемах ADuC824 и DataFlash и корректном питании записанные и прочитанные данные должны быть одинаковыми. Таким образом, варьируя адреса и данные записи, пользователь может наглядно убедиться в работоспособности предложенного программного модуля.

Аппаратную поддержку интерфейса SPI ADuC824 в программе использует подпрограмма Pod_TX_BYTE. Она осуществляет запись в ведомое устройство SPI одного байта данных из аккумулятора с одновременным чтением из ведомого устройства одного байта данных, который затем помещает в аккумулятор. Подпрограмма возвращает управление по окончании передачи. Прерывания по окончании передачи по SPI здесь не используются (запрещены), так как в данном конкретном примере они не дают большого выигрыша в быстродействии. Так, при максимальной частоте ядра и максимальной скорости передачи по шине SPI (Fcore/2=6,28 МГц) суммарное время перехода к обработке прерывания по окончании передачи по SPI и возврата из него в основную программу (с учетом необходимости сохранения и восстановления контекста) будет сопоставимо с временем простого ожидания окончания передачи по SPI в основной программе.

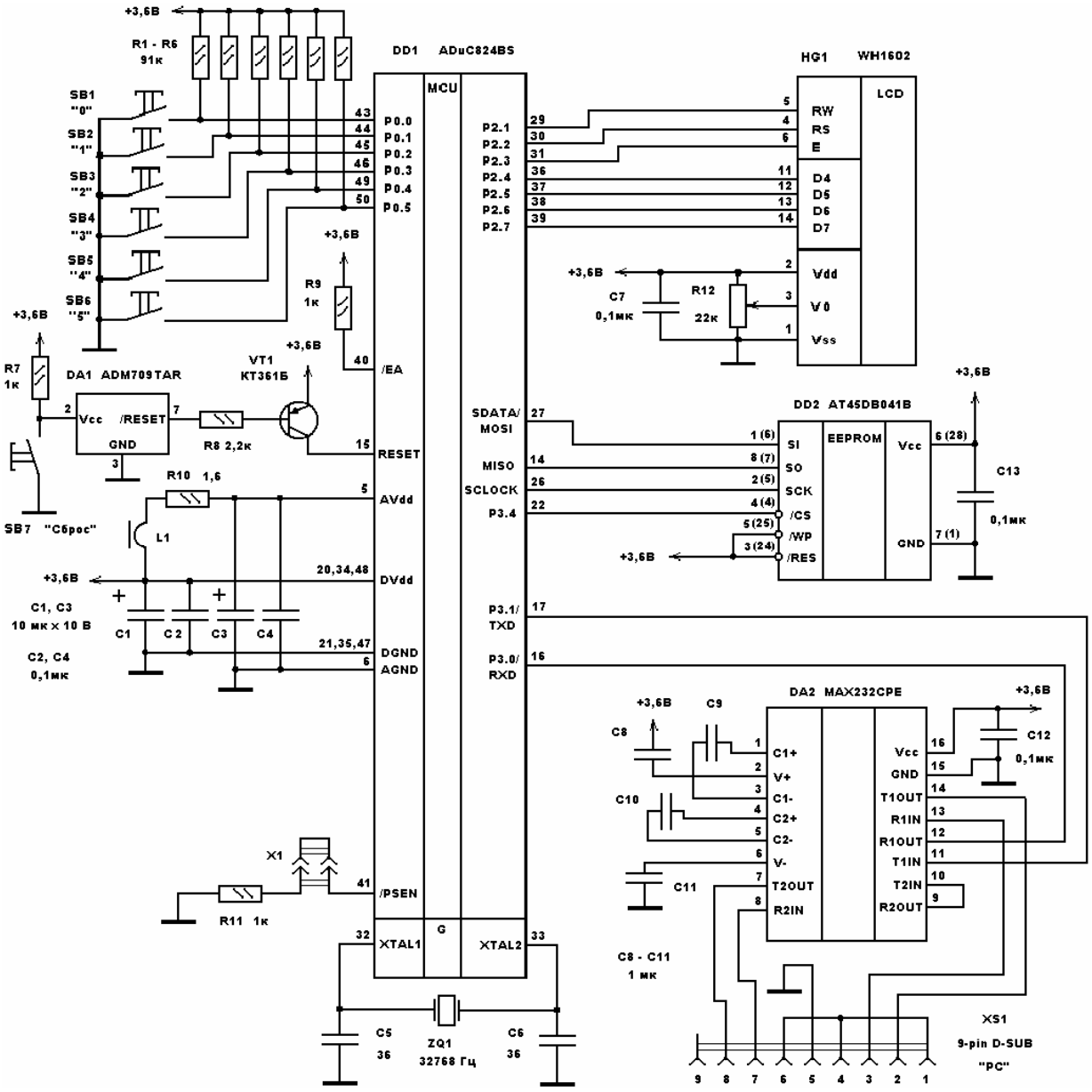


Рис. 3.10. Макет для исследования DataFlash

Настройку модуля SPI ADuC824 программа производит в подпрограмме Pod_INIT_RSN. Путем записи соответствующей константы в регистр SPICON можно установить режим SPI (0, 1, 2 или 3) и скорость обмена по шине.

Программа, исходный текст которой содержится в файле spi_df1.asm (листинг 3.14), функционально полностью идентична коду предыдущего примера, однако реализует чисто программную организацию интерфейса SPI. В качестве линий SCLOCK, MISO и MOSI интерфейса программа использует линии ввода-вывода общего назначения ADuC824 P3.5, P3.6 и P3.7 соответственно. Фрагмент принципиальной схемы, на котором показаны соединения DataFlash с ADuC824 для этого примера, приведен на рис. 3.11.

Листинг 3.14. Программная реализация интерфейса SPI

```
;-------------------------------------------------------------------------------
;Демонстрационная программа организации обмена между внешним устройством памяти
;AT45DB041 и ADuC824 по шине SPI. Напряжение питания не должно превышать 3,6 В.
;Данные в ЖКИ передаются по 4-битной шине, состояние ЖКИ опрашивается.
;ADuC824 является ведущим устройством SPI.
;Протокол обмена по шине SPI реализован чисто программно.
;
;При нажатии на кнопку 0 производится перебор адресов ячеек для записи данных в
;AT45DB041. Выбранное значение адреса ячейки в десятичн виде отображается на ЖКИ
;начиная с адреса 0 ОЗУ ЖКИ.
;
;При нажатии на кнопку 1 производится выбор байта данных для записи в AT45DB041.
;Выбранный байт данных в ASCII коде  выводится на ЖКИ по адресу 64 ОЗУ ЖКИ.
;
;При нажатии на кнопку 2 производится запись выбранного байта данных в AT45DB041
;по выбранному ранее адресу. Записанный байт данных выводится на ЖКИ по
;адресу 70 ОЗУ ЖКИ.
;
;При нажатии на кнопку 3 производится чтение байта данных из AT45DB041 по
;выбранному ранее адресу. Прочитанный байт данных в ASCII коде отображается на
;ЖКИ по адресу 72 ОЗУ ЖКИ.
;-------------------------------------------------------------------------------
          $INCLUDE  (C:\ADuC\mod824)
          $INCLUDE  (C:\PR_ADUC\SPI_DF1\824.inc)
;-------------------------------------------------------------------------------
;Описание битов, регистров и констант
;-------------------------------------------------------------------------------
;Порты и линии ввода-вывода
          PORT_KNOP    EQU      P0      ;порт кнопок

          PORT_IND     EQU      P2      ;порт индикации

          _IN_KNOP0    EQU      P0_0    ;------------------------------
          _IN_KNOP1    EQU      P0_1    ;
          _IN_KNOP2    EQU      P0_2    ;  входы кнопок
          _IN_KNOP3    EQU      P0_3    ;
          _IN_KNOP4    EQU      P0_4    ;
          _IN_KNOP5    EQU      P0_5    ;
          _IN_KNOP6    EQU      P0_6    ;
          _IN_KNOP7    EQU      P0_7    ;------------------------------

          PORT_IND_0   EQU      P2_0    ;------------------------------
          PORT_IND_1   EQU      P2_1    ;
          PORT_IND_2   EQU      P2_2    ; выходы индикации
          PORT_IND_3   EQU      P2_3    ;
          PORT_IND_4   EQU      P2_4    ;
          PORT_IND_5   EQU      P2_5    ;
          PORT_IND_6   EQU      P2_6    ;
          PORT_IND_7   EQU      P2_7    ;------------------------------

          RW           EQU      PORT_IND_1 ;--------------------------
          RS           EQU      PORT_IND_2 ; линии управления ЖКИ
          E            EQU      PORT_IND_3 ;--------------------------

          SC           EQU      P3_4    ;выход выбора устройства SPI
          SCLOCK       EQU      P3_5    ;выход тактового сигнала SPI
          MISO         EQU      P3_6    ;вход данных SPI
          MOSI         EQU      P3_7    ;выход данных SPI

;РОНы обслуживания ЖКИ и кнопок
          ADR_IND_R    DATA     030h    ;------------------------------
          COM_IND_R    DATA     030h    ; РОНы обслуживания ЖКИ
          DATA_IND_R   DATA     031h    ;------------------------------

          KNOP0_R      DATA     032h    ;------------------------------
          KNOP1_R      DATA     033h    ;
```

```
          KNOP2_R      DATA     034h    ; регистры, содержащие
          KNOP3_R      DATA     035h    ;
          KNOP4_R      DATA     036h    ; флаги нажатия и удержания
          KNOP5_R      DATA     037h    ;
          KNOP6_R      DATA     038h    ; каждой кнопки
          KNOP7_R      DATA     039h    ;-----------------------------

          NAKOPL0_R    DATA     03Ah    ;-----------------------------
          NAKOPL1_R    DATA     03Bh    ;
          NAKOPL2_R    DATA     03Ch    ; регистры накопления
          NAKOPL3_R    DATA     03Dh    ;
          NAKOPL4_R    DATA     03Eh    ; значения подавления дребезга
          NAKOPL5_R    DATA     03Fh    ;
          NAKOPL6_R    DATA     040h    ; каждой кнопки
          NAKOPL7_R    DATA     041h    ;-----------------------------

;РОНы обслуживания подпрограмм преобразования формы представления чисел
          IND_MILL_R   DATA     04Eh    ;РОН миллиардов дес числа
          IND_SOTMIL_R DATA     04Fh    ;РОН сотен миллионов дес числа
          IND_DESMIL_R DATA     050h    ;РОН десятков миллионов дес числа
          IND_MIL_R    DATA     051h    ;РОН миллионов дес числа
          IND_SOTTIS_R DATA     052h    ;РОН сотен тысяч дес числа
          IND_DESTIS_R DATA     053h    ;РОН десятков тысяч дес числа
          IND_TIS_R    DATA     054h    ;РОН тысяч дес числа
          IND_SOT_R    DATA     055h    ;РОН сотен дес числа
          IND_DES_R    DATA     056h    ;РОН десятков дес числа
          IND_ED_R     DATA     057h    ;РОН единиц дес числа

          BYTE_0_R     DATA     05Bh    ;байт 0 упакованного двоичн-дес числа
          BYTE_1_R     DATA     05Ch    ;байт 1 упакованного двоичн-дес числа
          BYTE_2_R     DATA     05Dh    ;байт 2 упакованного двоичн-дес числа
          BYTE_3_R     DATA     05Eh    ;байт 3 упакованного двоичн-дес числа
          BYTE_4_R     DATA     05Fh    ;байт 4 упакованного двоичн-дес числа

;РОНы обслуживания обмена с AT45DB041 по интерфейсу SPI
          DATASPI_R    DATA     064h    ;РОН данных чтения и записи SPI DF
          ADRF_L_R     DATA     065h    ;РОН мл байта абсолютного адр ячейки DF
          ADRF_M_R     DATA     066h    ;РОН ср байта абсолютного адр ячейки DF
          ADRF_H_R     DATA     067h    ;РОН ст байта абсолютного адр ячейки DF

;Константы
          NACH_ADR     EQU      000h    ;начальный адрес обнуления РОНов
          KON_ADR      EQU      07Fh    ;конечный адрес обнуления РОНов
          POROG_K      EQU      50      ;порог подавления дребезга кнопок
;Начало исполняемого кода-------------------------------------------------------
                ORG 0h
           AJMP       Lab_START            ;идти на начало осн программы

;Начало осн программы-----------------------------------------------------------
                ORG 100h
Lab_START: MOV        SP,#080h             ;определить указатель стека
           MOV        PLLCON,#00000000b    ;уст макс частоту ядра (12,58 МГц)
           NOP                             ;
           LCALL      Pod_INIT_RSN         ;иниц РСН
           LCALL      Pod_INIT_RON         ;иниц РОН

         MOV        R5,#021h         ;задание нач значения индицируемого байта 1

           LCALL      Pod_INIT_LCD         ;иниц ЖКИ

           LCALL      Pod_CLEAR_LCD        ;стирание ЖКИ

           MOV        R0,#BYTE_0_R         ;
           MOV        R1,#ADRF_L_R         ;
           MOV        ADRF_H_R+1,#0        ; обнуление ст незначащего РОНа
           LCALL      B32BCD               ;преобр адреса из двоичн в двоичн-дес

           MOV        R0,#BYTE_0_R         ;
```

```
           MOV        R1,#IND_MILL_R       ;
          LCALL      BCD10BCD             ;преобр адреса из уп дв-дес в неуп дес

           MOV        R0,#IND_MILL_R       ; индицировать с РОНа IND_MILL_R
           MOV        R1,#0                ; индицировать с адр 0 ЖКИ
           LCALL      Pod_IND_10ZN         ;индикация абсолютного адреса

;Начало основного цикла-----------------------------------------------------
La_OSN:    NOP                             ;метка возврата в осн цикле

;Блок, предшествующий вызову подпрограммы опроса кнопки 0.
           MOV        R0,#KNOP0_R          ;
           MOV        R1,#NAKOPL0_R        ;
           LCALL      Pod_OPR_KNOP0        ;
;Блок, следующий за вызовом подпрограммы опроса кнопки (уст факта ее наж)
           MOV        ACC,@R0              ;
           JNB        ACC_1,La_100         ;

           CLR        ACC_1                ;кн была нажата, сброс флага нажат кн
           MOV        @R0,ACC              ;

           MOV        ACC,ADRF_M_R         ;--------------------------------
           ADD        A,#32                ; Блок перебора адресов внутри
           JC         La_10                ; адресного пространства
           MOV        ADRF_M_R,ACC         ; с шагом (32 x 256) = 8192
           SJMP       La_50                ;
La_10:     MOV        ADRF_M_R,ACC         ;
           INC        ADRF_H_R             ;
           MOV        ACC,ADRF_H_R         ;
           CJNE       A,#00001000b,La_50   ;
           MOV        ADRF_M_R,#0          ;
           MOV        ADRF_H_R,#0          ;
La_50:     NOP                             ;--------------------------------

           MOV        R0,#BYTE_0_R         ;
           MOV        R1,#ADRF_L_R         ;
           MOV        ADRF_H_R+1,#0        ; обнуление ст незначащего РОНа
           LCALL      B32BCD               ;преобр адреса из двоичн в двоичн-дес

           MOV        R0,#BYTE_0_R         ;
           MOV        R1,#IND_MILL_R       ;
           LCALL      BCD10BCD             ;преобр адреса из уп дв-дес в неуп дес

           MOV        R0,#IND_MILL_R       ; индицировать с РОНа IND_MILL_R
           MOV        R1,#0                ; индицировать с адр 0 ЖКИ
           LCALL      Pod_IND_10ZN         ;индикация абсолютного адреса

;Блок, предшествующий вызову подпрограммы опроса кнопки 1.
La_100:    MOV        R0,#KNOP1_R          ;
           MOV        R1,#NAKOPL1_R        ;
           LCALL      Pod_OPR_KNOP1        ;
;Блок, следующий за вызовом подпрограммы опроса кнопки (уст факта ее наж)
           MOV        ACC,@R0              ;
           JNB        ACC_1,La_200         ;

           CLR        ACC_1                ;кн была нажата, сброс флага нажат кн
           MOV        @R0,ACC              ;

           INC        R5                   ;-----------------------------------
           CJNE       R5,#06Fh,La_00       ; перебор возможн знач индицир байта
           MOV        R5,#021h             ;-----------------------------------

La_00:     MOV        DATA_IND_R,R5        ;-----------------------------------
           MOV        ADR_IND_R,#64        ; индикация  байта (символа)
           ACALL      Pod_PER_DAT_LCD      ;-----------------------------------

;Блок, предшествующий вызову подпрограммы опроса кнопки 2.
La_200:    MOV        R0,#KNOP2_R          ;
```

```
           MOV        R1,#NAKOPL2_R        ;
           LCALL      Pod_OPR_KNOP2        ;
;Блок, следующий за вызовом подпрограммы опроса кнопки (уст факта ее наж)
           MOV        ACC,@R0              ;
           JNB        ACC_1,La_300         ;

           CLR        ACC_1                ;кн была нажата, сброс флага нажат кн
           MOV        @R0,ACC              ;

           MOV        DATASPI_R,R5         ;подготовка данных для зап в DF SPI
           LCALL      Pod_WRITE_BYTE       ;запись в DF SPI

           MOV        DATA_IND_R,DATASPI_R ;-----------------------------------
           MOV        ADR_IND_R,#70       ;индикация записанного байта (символа)
           ACALL      Pod_PER_DAT_LCD      ;-----------------------------------

;Блок, предшествующий вызову подпрограммы опроса кнопки 3.
La_300:    MOV        R0,#KNOP3_R          ;
           MOV        R1,#NAKOPL3_R        ;
           LCALL      Pod_OPR_KNOP3        ;
;Блок, следующий за вызовом подпрограммы опроса кнопки (уст факта ее наж)
           MOV        ACC,@R0              ;
           JNB        ACC_1,La_400         ;

           CLR        ACC_1                ;кн была нажата, сброс флага нажат кн
           MOV        @R0,ACC              ;

           MOV        DATASPI_R,#0         ;контрольная порча РОНа данных SPI
           LCALL      Pod_READ_BYTE        ;чтение из DF SPI

           MOV        DATA_IND_R,DATASPI_R ;-----------------------------------
           MOV        ADR_IND_R,#72     ; индикация прочитанного байта (символа)
           ACALL      Pod_PER_DAT_LCD      ;-----------------------------------

La_400:    LJMP       La_OSN               ;закрыть основной цикл


;Подпрограммы-------------------------------------------------------------------

;-------------------------------------------------------------------------
;Подпрограммы байтового обмена с DataFlash AT45D041.
;ADRF_H_R, ADRF_M_R, ADRF_L_R - РОНы хранения абсолютного адреса
;записи/чтения байта в массиве памяти (ст  ср  мл).
;DATA_R - РОН хранения данных записи/чтения.
;Адресные регистры не портятся.
;
;         Структура абсолютного адреса байта в массиве:
;
;      YYYYYYYY ZZZZZZZZ HHHHHHHH     - 3  байта адреса (ст  ср  мл)
; СЗР                              МЗР
;      xxxx                           - 4 незначащих бита
;          XXXX XXXXXXX               - 11 битов адреса стр (0...2047)
;                       X XXXXXXXX    - 9 битов адреса байта в стр (0...264)
;
;      0000XXXX XXXXXXX0 XXXXXXXX     - рекомендуемая маска абс адреса
;                                    (в случае использования этой маски в
;                                    странице доступно только 256 байт из 264)
;-------------------------------------------------------------------------
;Подпрограмма чтения байта из массива памяти по произвольному адресу.
;Прочитанный байт возвращается в РОНе DATA_R.
;Абсолютный адрес байта в массиве следует предварительно поместить
;в РОНы ADRF_H_R, ADRF_M_R, ADRF_L_R.
Pod_READ_BYTE:
        LCALL      Pod_READ_SR          ;чтение рег статуса DataFlash
        CLR        SC                   ;выбор устройства
         MOV        ACC,#053h           ;
        LCALL      Pod_TX_BYTE          ;передача КОПа перемещ стр в буфер 1
        LCALL      Pod_TX_ADRF          ;передача абс адреса байта в массиве
```

```
        SETB        SC                  ;запрет выбора устройства
        LCALL       Pod_READ_SR         ;чтение рег статуса DataFlash
        CLR         SC                  ;выбор устройства
        MOV         ACC,#0D4h           ;
        LCALL       Pod_TX_BYTE         ;передача КОПа чтения буфера 1
         LCALL       Pod_TX_ADRF         ;передача абс адреса байта в массиве
         LCALL       Pod_TX_BYTE         ;передача незнач байта
         LCALL       Pod_TX_BYTE         ;прием байта данных из массива
         MOV         DATASPI_R,ACC       ;сохранение принятого байта
        SETB        SC                  ;запрет выбора устройсва
        RET

;Подпрограмма чтения регистра статуса устройства памяти DataFlahs.
;Содержимое регистра статуса возвращается в ACC.
Pod_READ_SR:
        CLR         SC                  ;выбор устройства
        MOV         ACC,#0D7h           ;
        LCALL       Pod_TX_BYTE         ;передача КОПа чтения рег статуса
Re_0:   MOV         ACC,#0FFh           ;
        LCALL       Pod_TX_BYTE         ;чтение регистра статуса
        JNB         ACC_7,Re_0          ;свободно ли устройство
        SETB        SC                  ;запрет выбора устройсва
        RET

;Подпрограмма передачи трех байт абсолютного адреса байта в массиве памяти.
;Передача начинается со старшего байта адреса.
Pod_TX_ADRF:
        MOV         ACC,ADRF_H_R        ;передать ст байт адр
        LCALL       Pod_TX_BYTE         ;передача байта
        MOV         ACC,ADRF_M_R        ;передать ср байт адр
        LCALL       Pod_TX_BYTE         ;передача байта
        MOV         ACC,ADRF_L_R        ;передать мл байт адр
        LCALL       Pod_TX_BYTE         ;передача байта
        RET

;Подпрограмма записи байта в массив памяти по произвольному адресу.
;Записываемый байт следует предварительно поместить в РОН DATASPI_R.
;Абсолютный адрес байта следует предварительно поместить
;в РОНы ADRF_H_R, ADRF_M_R, ADRF_L_R.
Pod_WRITE_BYTE:
        LCALL       Pod_READ_SR         ;чтение рег статуса DataFlash
        CLR         SC                  ;выбор устройства
        MOV         ACC,#084h           ;
        LCALL       Pod_TX_BYTE         ;передача КОПа записи в буфер 1
        LCALL       Pod_TX_ADRF         ;передача абс адреса байта в массиве
        MOV         ACC,DATASPI_R       ;
        LCALL       Pod_TX_BYTE         ;запись байта данных
        SETB        SC                  ;запрет выбора устройсва
        LCALL       Pod_READ_SR         ;чтение рег статуса DataFlash
        CLR         SC                  ;выбор устройства
        MOV         ACC,#083h           ;
        LCALL       Pod_TX_BYTE         ;передача КОПа записи буфера 1 в стр
        LCALL       Pod_TX_ADRF         ;передача абс адреса байта в массиве
        SETB        SC                  ;запрет выбора устройсва
        RET

;Подпрограмма передачи в DataFlash по SPI одного байта, который предварительно
; следует поместить в ACC. Одновременно происходит прием байта из DataFlash.
;Принятый байт возвращается в ACC.
Pod_TX_BYTE:
        MOV          R0,#8              ;уст нач знач ст цикла
Tb_0:   MOV          C,ACC_7            ;----------------------------
        MOV          MOSI,C             ; передача бита данных
        NOP                             ;
        NOP                             ;----------------------------
        SETB         SCLOCK             ;фронт импульса SCLOCK
        NOP                             ;----------------------------
        NOP                             ; прием бита данных
```

```
        MOV         C,MISO              ;
        MOV         ACC_7,C             ;--------------------------
        CLR         SCLOCK              ;спад импульса SCLOCK
        RL          A                   ;сдвиг на следующий бит данных
        DJNZ        R0,Tb_0             ; не окончена ли передача?
        RET

;-------------------------------------------------------------------------------
;Подпрограмма инициализации РСН.
;-------------------------------------------------------------------------------
Pod_INIT_RSN:
          MOV       PORT_KNOP,#11111111b ;сделать вх все линии порта кн
          MOV       PORT_IND,#00000000b  ;сделать вых все линии порта индик

;Настройка линий программного SPI
          CLR       SCLOCK               ;сделать SCLOCK выходом
          CLR       MOSI                 ;сделать MOSI выходом
          SETB      MISO                 ;сделать MISO входом
          SETB      SC                   ;запрет выбора ведомого устр SPI
          RET

;-------------------------------------------------------------------------------
;Подпрограмма инициализации РОНов. Обнуляются все РОНы с адресами
;от NACH_ADR до KON_ADR при помощи косвенной адресации.
;-------------------------------------------------------------------------------
Pod_INIT_RON:
          MOV       R0,#NACH_ADR         ;установка начального адреса
Lk_0:     MOV       @R0,#0               ;обнуление очередного РОНа
          INC       R0                   ;переход к следующему адресу
          CJNE      R0,#KON_ADR,Lk_0     ;не достигли ли последнего адреса ?
          MOV       @R0,#0               ;обнуление последнего РОНа
          RET                            ;да, выход

;-------------------------------------------------------------------------------
;Подпрограмма вывода на ЖКИ содержимого буфера из 10 РОНов в виде 10 знакомест.
;R0 должен указывать на РОН в буфере, содержимое которого индицируется первым.
;R1 должен содержать нач адрес знакоместа в ОЗУ ЖКИ.
;-------------------------------------------------------------------------------
Pod_IND_10ZN:
           MOV      R2,#0                ;нач уст ст цикла
Ii_0:      MOV      DATA_IND_R,@R0       ;
           MOV      ADR_IND_R,R1         ;
           LCALL    Pod_PER_DAT_LCD      ; индикация очередного символа
           INC      R0                   ;
           INC      R1                   ;
           INC      R2                   ;
           CJNE     R2,#10,Ii_0          ;
           RET

;Подключение модулей опроса кнопок, вывода данных на ЖКИ (с опросом ЖКИ)
;и преобразования представления чисел
          $INCLUDE  (C:\PR_ADUC\SPI_DF1\knop.asm)
          $INCLUDE  (C:\PR_ADUC\SPI_DF1\lcd_opr.asm)
          $INCLUDE  (C:\PR_ADUC\SPI_DF1\preobr.asm)

;Конец исполняемого кода
          END
```

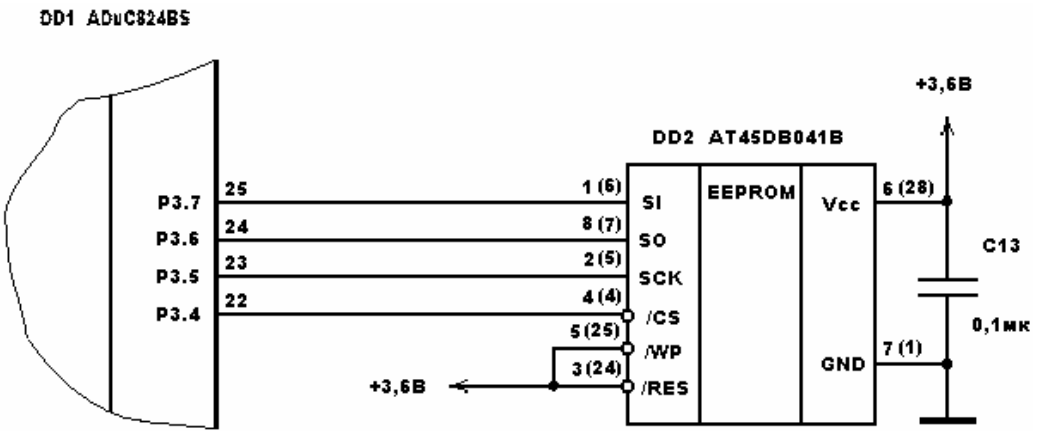


Рис. 3.11. Подключение DataFlash к МК при программной реализации интерфейса SPI

Настройка SPI в подпрограмме Pod_INIT_RSN сводится к установке на этих линиях начальных уровней. Подпрограмма записи/чтения байта данных Pod_TX_BYTE здесь получилась несколько длиннее, а скорость обмена данными несколько ниже, чем предельная скорость, обеспечиваемая аппаратным модулем. Режим SPI (режим 0) и скорость обмена заложены в самом порядке следования и количестве инструкций подпрограммы и не могут быть изменены (настроены) одной инструкцией записи в регистр, как это делается в предыдущем примере.

Описанный в файле spi_df1.asm программный интерфейс SPI может быть использован в любых 51-совместимых микроконтроллеров, не имеющих такого рода аппаратного интерфейса.

В заключение следует заметить, что при работе от низковольтного источника питания контрастность ЖКИ может оказаться недостаточной. В этом случае рекомендуется подавать на вход V0 ЖКИ отрицательное напряжение от внешнего источника или использовать модели ЖКИ, адаптированные для низкого питающего напряжения.

3.7. Использование интерфейса I^2C для подключения внешних устройств

Аппаратный модуль последовательного двухпроводного интерфейса, совместимого с I^2C, в ADuC824 может быть сконфигурирован пользовательским кодом либо как «программный ведущий», либо как «аппаратный ведомый». Соответственно, микроконвертор будет являться либо ведущим устройством на шине (Master), либо ведомым (Slave). Использование совместимого с I^2C интерфейса ADuC824 с примерами программ подробно описано в [6]. В некоторой степени, используя материалы этого источника, настоящая глава существенно дополняет его. Подробное общее описание протокола шины, совместимой с I^2C, широко представлено в литературе, например, в [5], поэтому в нашем случае ограничимся лишь комментированием временных диаграмм, отражающих процедуры обмена данными, реализованные описанными ниже программами. Несмотря на наличие только двух возможных режимов шины I^2C

ADuC824 вниманию читателей предлагается несколько различных программ, каждая из которых позволяет реализовать взаимодействие по шине, совместимой с I^2C, с некоторыми специфическими особенностями.

В качестве примера внешнего устройства, подключаемого к ADuC824 по интерфейсу I^2C, выбрана микросхема энергонезависимой многократно программируемой памяти с последовательным доступом 24LC64 производства фирмы Microchip [14]. Микросхемы указанного типа широко распространены и поставляются на рынок несколькими фирмами-производителями (Atmel, Microchip, STM и т. д.). 24LC64 организована в виде массива памяти, состоящего из 8192 8-разрядных ячеек хранения данных. В реальных проектах на основе ADuC824 такое устройство памяти можно использовать, например, для хранения калибровочных констант, табулированных функций, архивации собранных результатов измерений, линеаризации характеристик измерительных датчиков и т. д. Обмен ADuC824 с 24LC64 с использованием аппаратного модуля интерфейса, совместимого с I^2C, в режиме «программный ведущий» иллюстрируется демонстрационной программой, исходный текст которой находится в файле i2c_eep1.asm (листинг 3.15). Для экспериментов с этой программой необходимо собрать макет, принципиальная схема которого приведена на рис. 3.12.

Листинг 3.15. Исследование интерфейса I^2C

```
;-------------------------------------------------------------------------------
;Демонстрационная программа организации обмена между микросхемой EEPROM
;24LC64 и ADuC824 по шине I2C.
;ADuC824 является мастер-устройством I2C.
;Протокол обмена по шине I2C реализован программно-аппаратно
;(Используется аппаратный модуль в режиме "программный ведущий").
;Программное слежение за состоянием линии SCL  не производится.
;
;При нажатии на кнопку 0 производится перебор адресов ячеек для записи данных в
;EEPROM. Выбранное значение адреса ячейки в десятичном виде отображается на ЖКИ
;начиная с адреса 0 ОЗУ ЖКИ.
;
;При нажатии на кнопку 1 производится выбор байта данных для записи в EEPROM.
;Выбранный байт данных в ASCII коде  выводится на ЖКИ по адресу 64 ОЗУ ЖКИ.
;
;При нажатии на кнопку 2 производится запись выбранного байта данных в EEPROM
;по выбранному ранее адресу. Записанный байт данных выводится на ЖКИ по
;адресу 70 ОЗУ ЖКИ.
;
;При нажатии на кнопку 3 производится чтение байта данных из EEPROM по
;выбранному ранее адресу. Прочитанный байт данных в ASCII коде отображается на
;ЖКИ по адресу 72 ОЗУ ЖКИ.
;-------------------------------------------------------------------------------
          $INCLUDE  (C:\ADuC\mod824)
          $INCLUDE  (C:\PR_ADUC\I2C_EEP1\824.inc)
;-------------------------------------------------------------------------------
;Описание битов, регистров и констант
;-------------------------------------------------------------------------------
;Порты и линии ввода-вывода
          PORT_KNOP    EQU      P0      ;порт кнопок

          PORT_IND     EQU      P2      ;порт индикации

          _IN_KNOP0    EQU      P0_0    ;-----------------------------
          _IN_KNOP1    EQU      P0_1    ;
          _IN_KNOP2    EQU      P0_2    ;  входы кнопок
          _IN_KNOP3    EQU      P0_3    ;
```

```
          _IN_KNOP4    EQU      P0_4    ;
          _IN_KNOP5    EQU      P0_5    ;
          _IN_KNOP6    EQU      P0_6    ;
          _IN_KNOP7    EQU      P0_7    ;-----------------------------

          PORT_IND_0   EQU      P2_0    ;-----------------------------
          PORT_IND_1   EQU      P2_1    ;
          PORT_IND_2   EQU      P2_2    ; выходы индикации
          PORT_IND_3   EQU      P2_3    ;
          PORT_IND_4   EQU      P2_4    ;
          PORT_IND_5   EQU      P2_5    ;
          PORT_IND_6   EQU      P2_6    ;
          PORT_IND_7   EQU      P2_7    ;-----------------------------

          RW           EQU      PORT_IND_1 ;--------------------------
          RS           EQU      PORT_IND_2 ; линии управления ЖКИ
          E            EQU      PORT_IND_3 ;--------------------------

;РОНы обслуживания ЖКИ и кнопок
          ADR_IND_R    DATA     030h    ;-----------------------------
          COM_IND_R    DATA     030h    ; РОНы обслуживания ЖКИ
          DATA_IND_R   DATA     031h    ;-----------------------------

          KNOP0_R      DATA     032h    ;-----------------------------
          KNOP1_R      DATA     033h    ;
          KNOP2_R      DATA     034h    ; регистры, содержащие
          KNOP3_R      DATA     035h    ;
          KNOP4_R      DATA     036h    ; флаги нажатия и удержания
          KNOP5_R      DATA     037h    ;
          KNOP6_R      DATA     038h    ; каждой кнопки
          KNOP7_R      DATA     039h    ;-----------------------------

          NAKOPL0_R    DATA     03Ah    ;-----------------------------
          NAKOPL1_R    DATA     03Bh    ;
          NAKOPL2_R    DATA     03Ch    ; регистры накопления
          NAKOPL3_R    DATA     03Dh    ;
          NAKOPL4_R    DATA     03Eh    ; значения подавления дребезга
          NAKOPL5_R    DATA     03Fh    ;
          NAKOPL6_R    DATA     040h    ; каждой кнопки
          NAKOPL7_R    DATA     041h    ;-----------------------------

;РОНы обслуживания подпрограмм преобразования формы представления чисел
          IND_MILL_R   DATA     04Eh    ;РОН миллиардов дес числа
          IND_SOTMIL_R DATA     04Fh    ;РОН сотен миллионов дес числа
          IND_DESMIL_R DATA     050h    ;РОН десятков миллионов дес числа
          IND_MIL_R    DATA     051h    ;РОН миллионов дес числа
          IND_SOTTIS_R DATA     052h    ;РОН сотен тысяч дес числа
          IND_DESTIS_R DATA     053h    ;РОН десятков тысяч дес числа
          IND_TIS_R    DATA     054h    ;РОН тысяч дес числа
          IND_SOT_R    DATA     055h    ;РОН сотен дес числа
          IND_DES_R    DATA     056h    ;РОН десятков дес числа
          IND_ED_R     DATA     057h    ;РОН единиц дес числа

          BYTE_0_R     DATA     05Bh    ;байт 0 упакованного двоичн-дес числа
          BYTE_1_R     DATA     05Ch    ;байт 1 упакованного двоичн-дес числа
          BYTE_2_R     DATA     05Dh    ;байт 2 упакованного двоичн-дес числа
          BYTE_3_R     DATA     05Eh    ;байт 3 упакованного двоичн-дес числа
          BYTE_4_R     DATA     05Fh    ;байт 4 упакованного двоичн-дес числа

;РОНы обслуживания интерфейса программный ведущий-I2C
       DATA_I2C_R   DATA     060h    ;РОН данных I2C
       SLUG_I2C_R   DATA     061h    ;РОН служ информ I2C (адр устр и призн зап)
       ADRL_I2C_R   DATA     062h    ;РОН мл байта адр ячейки в ведомом устр I2C
       ADRH_I2C_R   DATA     063h    ;РОН ст байта адр ячейки в ведомом устр I2C

;Флаги
          _ERR_I2C     BIT      07h     ;флаг "ошибка I2C"
```

```
;Константа обслуживания EEPROM I2C
       EE_I2C_K     EQU      10101110b ;служ слово обращ к EEPROM I2C, включает:
                                       ;тип устройства I2C (1010), номер устр
                                       ;I2C на шине (111) и признак записи (0)
;Прочие константы
          NACH_ADR     EQU      000h    ;начальный адрес обнуления РОНов
          KON_ADR      EQU      07Fh    ;конечный адрес обнуления РОНов
          POROG_K      EQU      50      ;порог подавления дребезга кнопок

;Начало исполняемого кода-------------------------------------------------------
               ORG 0h
           AJMP       Lab_START           ;идти на начало осн программы

;Начало осн программы-----------------------------------------------------------
               ORG 100h
Lab_START: MOV        SP,#080h            ;определить указатель стека
           MOV        PLLCON,#00000000b   ;уст макс частоту ядра (12,58 МГц)
           NOP                            ;
           LCALL      Pod_INIT_RSN        ;иниц РСН
           LCALL      Pod_INIT_RON        ;иниц РОН

           MOV        R5,#021h         ;задание нач значен индицируемого байта

           LCALL      Pod_INIT_LCD        ;иниц ЖКИ

           LCALL      Pod_CLEAR_LCD       ;стирание ЖКИ

           MOV        R0,#BYTE_0_R        ;
           MOV        R1,#ADRL_I2C_R      ;
           MOV        ADRH_I2C_R+1,#0     ; обнуление ст незначащих РОНов
           MOV        ADRH_I2C_R+2,#0     ;------------------------------
           LCALL      B32BCD            ;преобр адреса из двоичн в двоичн-дес

           MOV        R0,#BYTE_0_R        ;
           MOV        R1,#IND_MILL_R      ;
           LCALL      BCD10BCD          ;преобр адреса из уп дв-дес в неуп дес

           MOV        R0,#IND_MILL_R      ; индицировать с РОНа IND_MILL_R
           MOV        R1,#0               ; индицировать с адр 0 ЖКИ
           LCALL      Pod_IND_10ZN        ;индикация адреса

;Начало основного цикла---------------------------------------------------------
La_OSN:    NOP                            ;метка возврата в осн цикле

;Блок, предшествующий вызову подпрограммы опроса кнопки 0.
           MOV        R0,#KNOP0_R         ;
           MOV        R1,#NAKOPL0_R       ;
           LCALL      Pod_OPR_KNOP0       ;
;Блок, следующий за вызовом подпрограммы опроса кнопки (уст факта ее наж)
           MOV        ACC,@R0             ;
           JNB        ACC_1,La_100        ;

           CLR        ACC_1               ;кн была нажата, сброс флага нажат кн
           MOV        @R0,ACC             ;

           MOV        ACC,ADRL_I2C_R      ;---------------------------------
           ADD        A,#64               ; Блок перебора адресов внутри
           JC         La_10               ; адресного пространства 0 - 1FFFh
           MOV        ADRL_I2C_R,ACC      ; с шагом 64
           SJMP       La_50               ;
La_10:     MOV        ADRL_I2C_R,ACC      ;
           INC        ADRH_I2C_R          ;
           MOV        ACC,ADRH_I2C_R      ;
           CJNE       A,#020h,La_50       ;
           MOV        ADRL_I2C_R,#0       ;
           MOV        ADRH_I2C_R,#0       ;
La_50:     NOP                            ;---------------------------------
```

```
            MOV         R0,#BYTE_0_R          ;
            MOV         R1,#ADRL_I2C_R        ;
            MOV         ADRH_I2C_R+1,#0       ; обнуление ст незначащих РОНов
            MOV         ADRH_I2C_R+2,#0       ;-----------------------------
            LCALL       B32BCD              ;преобр адреса из двоичн в двоичн-дес

            MOV         R0,#BYTE_0_R          ;
            MOV         R1,#IND_MILL_R        ;
            LCALL       BCD10BCD            ;преобр адреса из уп дв-дес в неуп дес

            MOV         R0,#IND_MILL_R        ; индицировать с РОНа IND_MILL_R
            MOV         R1,#0                 ; индицировать с адр 0 ЖКИ
            LCALL       Pod_IND_10ZN          ;индикация адреса

;Блок, предшествующий вызову подпрограммы опроса кнопки 1.
La_100:     MOV         R0,#KNOP1_R           ;
            MOV         R1,#NAKOPL1_R         ;
            LCALL       Pod_OPR_KNOP1         ;
;Блок, следующий за вызовом подпрограммы опроса кнопки (уст факта ее наж)
            MOV         ACC,@R0               ;
            JNB         ACC_1,La_200          ;

            CLR         ACC_1                 ;кн была нажата, сброс флага нажат кн
            MOV         @R0,ACC               ;

            INC         R5                    ;-----------------------------------
            CJNE        R5,#06Fh,La_00        ; перебор возможн знач индицир байта
            MOV         R5,#021h              ;-----------------------------------

La_00:      MOV         DATA_IND_R,R5         ;-----------------------------------
            MOV         ADR_IND_R,#64         ; индикация  байта (символа)
            ACALL       Pod_PER_DAT_LCD       ;-----------------------------------

;Блок, предшествующий вызову подпрограммы опроса кнопки 2.
La_200:     MOV         R0,#KNOP2_R           ;
            MOV         R1,#NAKOPL2_R         ;
            LCALL       Pod_OPR_KNOP2         ;
;Блок, следующий за вызовом подпрограммы опроса кнопки (уст факта ее наж)
            MOV         ACC,@R0               ;
            JNB         ACC_1,La_300          ;

            CLR         ACC_1                 ;кн была нажата, сброс флага нажат кн
            MOV         @R0,ACC               ;

            MOV         SLUG_I2C_R,#EE_I2C_K ;подготовка служебной константы I2C
            MOV         DATA_I2C_R,R5       ;подготовка данных для зап в EEPROM I2C
            LCALL       Pod_WRITE_I2C         ;запись в EEPROM I2C

            MOV         DATA_IND_R,DATA_I2C_R;-----------------------------------
            MOV         ADR_IND_R,#70       ; индикация записанного байта (символа)
            ACALL       Pod_PER_DAT_LCD       ;-----------------------------------

;Блок, предшествующий вызову подпрограммы опроса кнопки 3.
La_300:     MOV         R0,#KNOP3_R           ;
            MOV         R1,#NAKOPL3_R         ;
            LCALL       Pod_OPR_KNOP3         ;
;Блок, следующий за вызовом подпрограммы опроса кнопки (уст факта ее наж)
            MOV         ACC,@R0               ;
            JNB         ACC_1,La_400          ;

            CLR         ACC_1                 ;кн была нажата, сброс флага нажат кн
            MOV         @R0,ACC               ;

            MOV         SLUG_I2C_R,#EE_I2C_K ;подготовка служебной константы I2C
            MOV         DATA_I2C_R,#0FFh      ;контрольная порча РОНа данных I2C
            LCALL       Pod_READ_I2C          ;чтение из EEPROM I2C

            MOV         DATA_IND_R,DATA_I2C_R;-----------------------------------
```

```
            MOV         ADR_IND_R,#72       ;индикация прочитанного байта (символа)
            ACALL       Pod_PER_DAT_LCD      ;-----------------------------------

La_400:     LJMP        La_OSN               ;закрыть основной цикл


;Подпрограммы-------------------------------------------------------------------

;Подпрограммы организации обмена (записи и чтения по произвольному адресу)
;по шине I2C для мастер-устройства.
;-------------------------------------------------------------------------------
;Подпрограмма базового интервала I2C
;-------------------------------------------------------------------------------
Pod_DEL_I2C:
            NOP                             ; тело задержки
            RET

;-------------------------------------------------------------------------------
;Подпрограмма генерации условия START I2C
;-------------------------------------------------------------------------------
Pod_START:
            SETB        MDE                 ;SDA - выход
            CLR         MCO                 ;
            CALL        Pod_DEL_I2C         ;
            SETB        MDO                 ;
            CALL        Pod_DEL_I2C         ;
            SETB        MCO                 ;
            CALL        Pod_DEL_I2C         ;
            CLR         MDO                 ;START
            CALL        Pod_DEL_I2C         ;
            CLR         MCO                 ;
            RET

;-------------------------------------------------------------------------------
;Подпрограмма генерации условия STOP I2C
;-------------------------------------------------------------------------------
Pod_STOP:   SETB        MDE                 ;SDA - выход
            CLR         MCO                 ;
            CALL        Pod_DEL_I2C         ;
            CLR         MDO                 ;
            CALL        Pod_DEL_I2C         ;
            SETB        MCO                 ;
            CALL        Pod_DEL_I2C         ;
            SETB        MDO                 ;STOP
            RET

;-------------------------------------------------------------------------------
;Подпрограмма опроса подтверждения на линии SDA I2C.
;В случае отсутствия подтверждения (выс уровень на линии SDA) устанавливается
;флаг ошибки шины I2C - _ERR_I2C.
;-------------------------------------------------------------------------------
Pod_ZACK:   CLR         MDE                 ;SDA - вход
            SETB        MCO                 ;-------------------------------
            CALL        Pod_DEL_I2C         ;
            JNB         MDI,Zack_0          ;есть ли подтверждение ?
            SETB        _ERR_I2C            ;подтверждения нет - уст флаг ошибк I2C
Zack_0:     CLR         MCO                 ;-------------------------------
            RET

;-------------------------------------------------------------------------------
;Подпрограмма выдачи неподтверждения на линию SDA I2C.
;-------------------------------------------------------------------------------
Pod_NACK:   SETB        MDE                 ;SDA - выход
            SETB        MDO                 ;
            CALL        Pod_DEL_I2C         ;
            SETB        MCO                 ;-------------------------------
            CALL        Pod_DEL_I2C         ;
            CLR         MCO                 ;-------------------------------
```

```
          RET

;-------------------------------------------------------------------------------
;Подпрограмма передачи байта по шине I2C. Передаваемый байт следует
;предварительно поместить в аккумулятор.  Используется счетчик циклов R3.
;Аккумулятор портится.
;-------------------------------------------------------------------------------
Pod_OUT_BYTE:
          SETB        MDE                 ;SDA - выход
          MOV         R3,#8               ;нач уст ст циклов
Out_0:    RLC         A                   ;сдвиг A влево через C, бит 7 идет в C
          MOV         MDO,C               ;передача бита
          CALL        Pod_DEL_I2C         ;
          SETB        MCO                 ;--------------------------------
          CALL        Pod_DEL_I2C         ;
          CLR         MCO                 ;--------------------------------
          DJNZ        R3,Out_0            ;
          RET

;-------------------------------------------------------------------------------
;Подпрограмма приема байта по шине I2C. Принятый байт оказывается в аккумуляторе.
;Используется счетчик циклов R3.
;-------------------------------------------------------------------------------
Pod_IN_BYTE:
          CLR         MDE                 ;SDA - вход
          MOV         R3,#8               ;нач уст ст циклов
In_0:     SETB        MCO                 ;--------------------------------
          CALL        Pod_DEL_I2C         ;
          MOV         C,MDI               ;прием бита
          CLR         MCO                 ;--------------------------------
          RLC         A                   ;сдвиг A влево через C, C идет в бит 0
          DJNZ        R3,In_0             ;
          RET

;-------------------------------------------------------------------------------
;Подпрограмма фиктивной записи (обращение с признаком записи по
;некоторому адресу ячейки без собственно записи) по шине I2C.
;-------------------------------------------------------------------------------
Pod_PWRITE_I2C:
          LCALL       Pod_START           ;START
          MOV         A,SLUG_I2C_R        ;  передача адр устройства
          LCALL       Pod_OUT_BYTE        ;  и признака записи
          LCALL       Pod_ZACK            ;запрос подтверждения
          MOV         A,ADRH_I2C_R        ;  передача ст байта адреса ячейки
          LCALL       Pod_OUT_BYTE        ;  в устройстве
          LCALL       Pod_ZACK            ;запрос подтверждения
          MOV         A,ADRL_I2C_R        ;  передача мл байта адреса ячейки
          LCALL       Pod_OUT_BYTE        ;  в устройстве
          LCALL       Pod_ZACK            ;запрос подтверждения
          RET

;-------------------------------------------------------------------------------
;Подпрограмма записи байта в ведомое устройство EEPROM по шине I2C.
;Предварительно следует поместить в РОНы ADRH_I2C_R, ADRL_I2C_R адрес ячейки,
;в РОН DATA_I2C_R - данные записи, в РОН SLUG_I2C_R - адрес устройства и признак
;операции (запись).
;-------------------------------------------------------------------------------
Pod_WRITE_I2C:
          LCALL       Pod_PWRITE_I2C      ;фиктивная запись - указание на ячейку
          MOV         A,DATA_I2C_R        ;  передача данных записи
          LCALL       Pod_OUT_BYTE        ;
          LCALL       Pod_ZACK            ;запрос подтверждения
          LCALL       Pod_STOP            ;STOP
          RET

;-------------------------------------------------------------------------------
;Подпрограмма чтения байта из ведомого устройства EEPROM по шине I2C.
;Предварит следует поместить в РОНы ADRH_I2C_R, ADRL_I2C_R адрес ячейки, а в
```

```
;РОН SLUG_I2C_R - адрес устройства и признак операции (запись).
;Прочитанный байт данных возвращается в РОНе DATA_I2C_R.
;-------------------------------------------------------------------------
Pod_READ_I2C:
          LCALL       Pod_PWRITE_I2C      ;фиктивная запись - указание на ячейку
          LCALL       Pod_START           ;повторный START
          MOV         A,SLUG_I2C_R        ;
          SETB        ACC_0               ;уст признак чтения данных
          LCALL       Pod_OUT_BYTE        ;передача адреса устр и признака чтения
          LCALL       Pod_ZACK            ;запрос подтверждения
          LCALL       Pod_IN_BYTE         ;чтение байта данных из устройства
          LCALL       Pod_NACK            ;не даем подтверждения
          LCALL       Pod_STOP            ;STOP
          MOV         DATA_I2C_R,A        ;
          RET

;-------------------------------------------------------------------------
;Подпрограмма инициализации РСН.
;-------------------------------------------------------------------------
Pod_INIT_RSN:
          MOV         PORT_KNOP,#11111111b ;сделать вх все линии порта кн
          MOV         PORT_IND,#00000000b  ;сделать вых все линии порта индик

;настройка модуля I2C
          MOV         I2CCON,#10101000b    ;уст SDA и SCL, SDA - вход,
                                           ;уст режим ведущего
          RET

;-------------------------------------------------------------------------
;Подпрограмма инициализации РОНов. Обнуляются все РОНы с адресами
;от NACH_ADR до KON_ADR при помощи косвенной адресации.
;-------------------------------------------------------------------------
Pod_INIT_RON:
          MOV         R0,#NACH_ADR         ;установка начального адреса
Lk_0:     MOV         @R0,#0               ;обнуление очередного РОНа
          INC         R0                   ;переход к следующему адресу
          CJNE        R0,#KON_ADR,Lk_0     ;не достигли ли последнего адреса ?
          MOV         @R0,#0               ;обнуление последнего РОНа
          RET                              ;да, выход

;-------------------------------------------------------------------------
;Подпрограмма вывода на ЖКИ содержимого буфера из 10 РОНов в виде 10 знакомест.
;R0 должен указывать на РОН в буфере, содержимое которого индицируется первым.
;R1 должен содержать нач адрес знакоместа в ОЗУ ЖКИ.
;-------------------------------------------------------------------------
Pod_IND_10ZN:
           MOV        R2,#0                ;нач уст ст цикла
Ii_0:      MOV        DATA_IND_R,@R0       ;
           MOV        ADR_IND_R,R1         ;
           LCALL      Pod_PER_DAT_LCD      ; индикация очередного символа
           INC        R0                   ;
           INC        R1                   ;
           INC        R2                   ;
           CJNE       R2,#10,Ii_0          ;
           RET

;Подключение модулей опроса кнопок, вывода данных на ЖКИ (с опросом ЖКИ)
;и преобразования представления чисел
          $INCLUDE   (C:\PR_ADUC\I2C_EEP1\knop.asm)
          $INCLUDE   (C:\PR_ADUC\I2C_EEP1\lcd_opr.asm)
          $INCLUDE   (C:\PR_ADUC\I2C_EEP1\preobr.asm)

;Конец исполняемого кода
          END
```

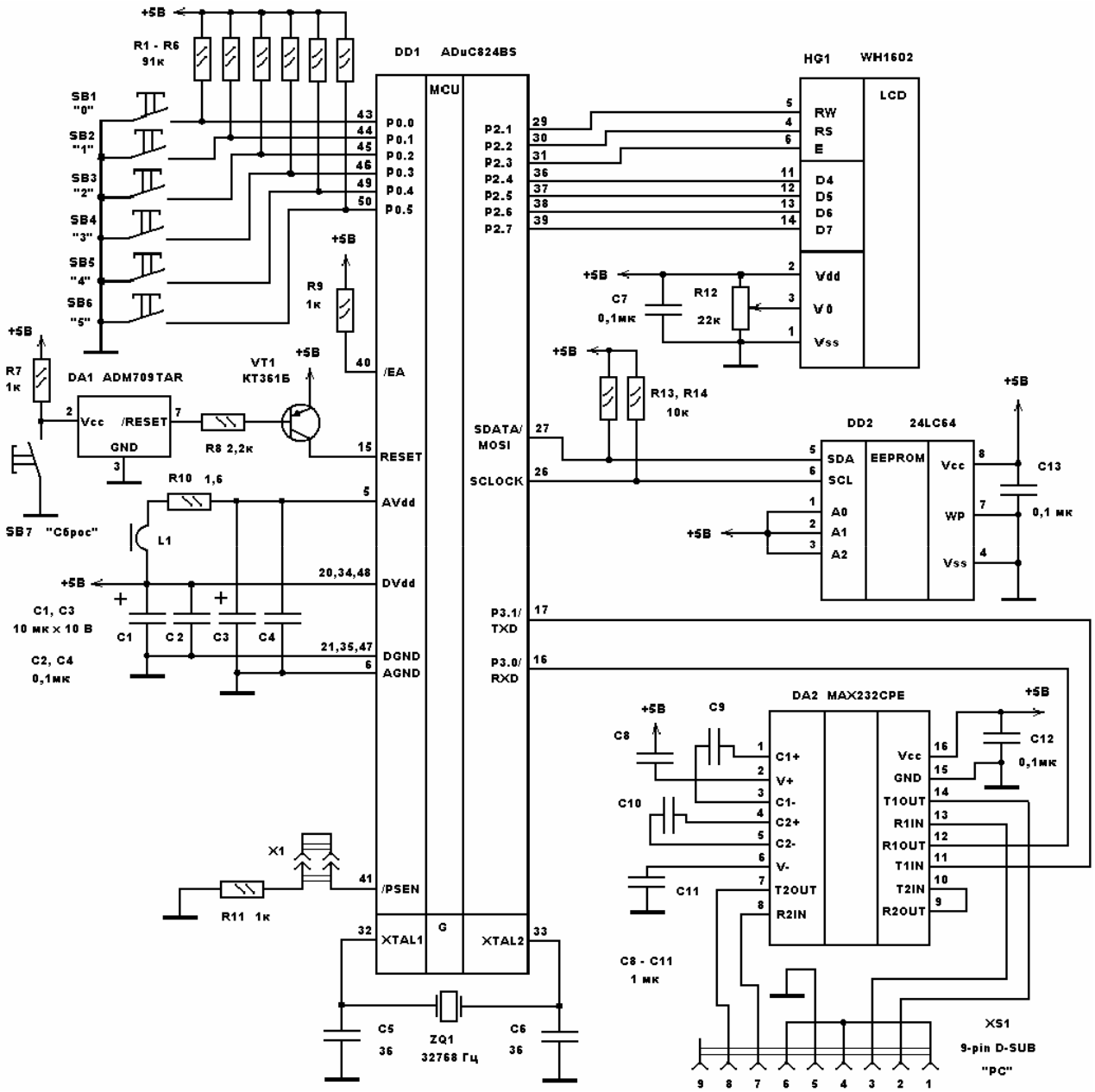


Рис. 3.12. Схема для исследования интерфейса I^2C

Временные диаграммы записи и чтения байта 24LC64 по произвольному адресу приведены соответственно на рис. 3.13 а, б.

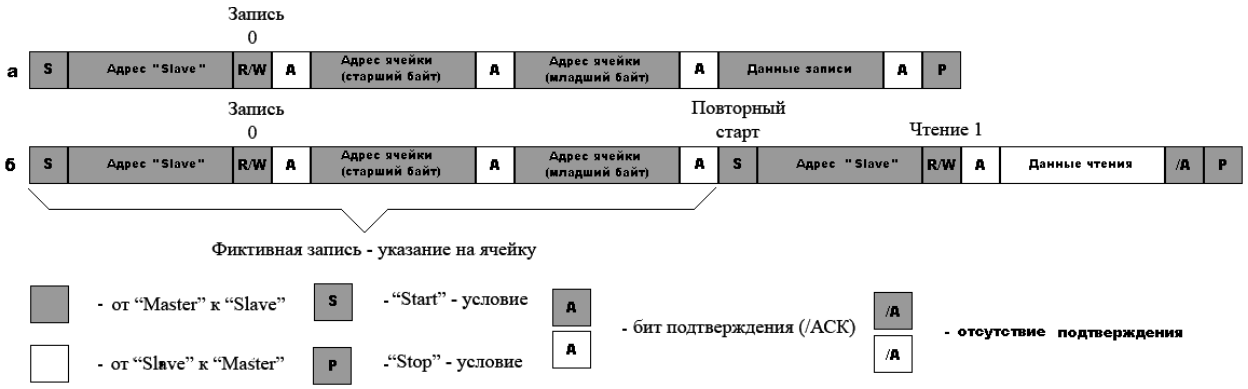


Рис. 3.13. Запись и чтение байта по интерфейсу I^2C

Адресация Slave-устройства 7-битовая, а адрес ячейки I2C EEPROM передается двумя байтами, тат как микросхема имеет емкость больше 256 байт. Первые четыре бита в 7-битовом адресе Slave-устройства являются маской, установленной разработчиками стандарта для микросхем EEPROM I^2C, – 1010, а оставшиеся три бита образуют собственный адрес Slave-устройства на шине – 111. Этот адрес задается подачей на адресные входы DD2 A0, A1, A2 (рис. 3.12) соответствующих (высоких) логических уровней.

При нажатии на кнопку «0» производится перебор адресов ячеек для записи данных в I^2C EEPROM. Выбранное значение адреса ячейки в десятичном виде отображается на ЖКИ с адреса 0 ОЗУ ЖКИ. При нажатии на кнопку «1» производится выбор байта данных для записи в I2C EEPROM. Выбранный байт данных в ASCII-коде индицируется по адресу 64 ОЗУ ЖКИ. При нажатии на кнопку «2» производится запись выбранного байта данных в EEPROM по выбранному ранее адресу. Записанный байт данных индицируется по адресу 70 ОЗУ ЖКИ. При нажатии на кнопку «3» производится чтение байта данных из I^2C EEPROM по выбранному ранее адресу. Прочитанный байт данных в ASCII-коде индицируется по адресу 72 ОЗУ ЖКИ.

Для обеспечения возможности записи и чтения по произвольным адресам программа производит процедуру так называемой фиктивной записи – обращение к ячейке памяти без собственно записи или чтения (указание на ячейку для последующих записи или чтения). Эта процедура реализована в подпрограмме Pod_PWRITE_I2C. В случае осуществления записи после этой процедуры Master-устройство выдает на шину байт данных записи, дожидается от Slave-устройства подтверждения и завершает всю процедуру обмен условием «Stop». Запись байта данных в I^2C EEPROM по произвольному адресу реализуется подпрограммой Pod_WRITE_I2C. В случае осуществления чтения после процедуры фиктивной записи Master-устройство повторно генерирует условие «Start», повторно выдает на шину адрес Slave-устройства уже с признаком чтения, дожидается подтверждения, а затем принимает байт данных чтения, передаваемых Slave-устройством. По завершении приема Master-устройство выдает на шину бит неподтверждения и завершает всю процедуру чтения выдачей ус-

ловия «Stop». Чтение байта данных из I^2C EEPROM по произвольному адресу реализуется подпрограммой Pod_READ_I2C.

Описанные подпрограммы процедур записи и чтения получаются довольно большими по количеству инструкций, реальная скорость обмена оказывается невысокой (по сравнению со скоростью записи и чтения по текущему адресу), но это плата за возможность обращения к ячейкам памяти с произвольными адресами.

Следует заметить, аппаратный модуль I2C ADuC824 имеет один довольно серьезный недостаток – при операциях обмена по шине модуль не осуществляет слежение за текущим состоянием линии синхронизации SCL. В соответствии с протоколом шины I^2C Slave-устройство в процессе приема данных может удерживать линию SCL в низком уровне, тем самым, давая указание Master-устройству на приостановку передачи по шине. Это может происходить, когда скорость обмена по шине, задаваемая Master-устройством, превышает предельную скорость обмена, допустимую для Slave-устройства. Master-устройство, установив перед передачей очередного бита данных линию SCL в высокий уровень («отпустив» CSL), должно убедиться в отсутствии такого удержания и только в этом случае продолжить передачу по шине. Если такой контроль не осуществляется, то в описанной выше ситуации возможна потеря передаваемых по шине данных. Следует также отметить, что удержание SCL могут производить не все Slave-устройства. Например, микросхемы I^2C EEPROM, являясь чисто аппаратными устройствами и имея высокую предельную скорость обмена, не удерживают линию SCL.

Отсутствие аппаратного механизма контроля состояния SCL в модуле I2C ADuC824 побудило реализовать такой механизм программно. Документ [6] рекомендует для контроля состояния SCL использовать любую линию ввода-вывода общего назначения микроконвертора, настроенную как вход и соединенную в устройстве с линией SCL I^2C. Реализация режима «программный ведущий» при наличии такого рода слежения за SCL иллюстрируется программой, исходный текст которой содержится в файле i2c_eep2.asm (листинг 3.16). Функционально эта программа полностью идентична коду из предыдущего примера. Фрагмент принципиальной схемы, на котором показано подключение EEPROM к ADuC824 для этого примера, приведен на рис. 3.14.

Листинг 3.16. Работа по интерфейсу I^2C с контролем линий

```
;-------------------------------------------------------------------------
;Демонстрационная программа организации обмена между микросхемой EEPROM
;24LC64 и ADuC824 по шине I2C.
;
;ADuC824 является мастер-устройством I2C.
;Протокол обмена по шине I2C реализован программно-аппаратно
;(Используется аппаратный модуль в режиме "программный ведущий").
```

```
;Дополнительно программно реализовано слежение за состоянием линии SCL,
;производимое через линию ввода-вывода общего назначения (_M_SCL),
;которая должна быть соединена с линией SCL.
;
;При нажатии на кнопку 0 производится перебор адресов ячеек для записи данных в
;EEPROM. Выбранное значение адреса ячейки в десятичном виде отображается на ЖКИ
;начиная с адреса 0 ОЗУ ЖКИ.
;
;При нажатии на кнопку 1 производится выбор байта данных для записи в EEPROM.
;Выбранный байт данных в ASCII коде  выводится на ЖКИ по адресу 64 ОЗУ ЖКИ.
;
;При нажатии на кнопку 2 производится запись выбранного байта данных в EEPROM
;по выбранному ранее адресу. Записанный байт данных выводится на ЖКИ по
;адресу 70 ОЗУ ЖКИ.
;
;При нажатии на кнопку 3 производится чтение байта данных из EEPROM по
;выбранному ранее адресу. Прочитанный байт данных в ASCII коде отображается на
;ЖКИ по адресу 72 ОЗУ ЖКИ.
;-------------------------------------------------------------------------------
          $INCLUDE  (C:\ADuC\mod824)
          $INCLUDE  (C:\PR_ADUC\I2C_EEP2\824.inc)
;-------------------------------------------------------------------------------
;Описание битов, регистров и констант
;-------------------------------------------------------------------------------
;Порты и линии ввода-вывода
          PORT_KNOP    EQU      P0      ;порт кнопок

          PORT_IND     EQU      P2      ;порт индикации

          _IN_KNOP0    EQU      P0_0    ;-----------------------------
          _IN_KNOP1    EQU      P0_1    ;
          _IN_KNOP2    EQU      P0_2    ;  входы кнопок
          _IN_KNOP3    EQU      P0_3    ;
          _IN_KNOP4    EQU      P0_4    ;
          _IN_KNOP5    EQU      P0_5    ;
          _IN_KNOP6    EQU      P0_6    ;
          _IN_KNOP7    EQU      P0_7    ;-----------------------------

          PORT_IND_0   EQU      P2_0    ;-----------------------------
          PORT_IND_1   EQU      P2_1    ;
          PORT_IND_2   EQU      P2_2    ; выходы индикации
          PORT_IND_3   EQU      P2_3    ;
          PORT_IND_4   EQU      P2_4    ;
          PORT_IND_5   EQU      P2_5    ;
          PORT_IND_6   EQU      P2_6    ;
          PORT_IND_7   EQU      P2_7    ;-----------------------------

          RW           EQU      PORT_IND_1 ;-------------------------
          RS           EQU      PORT_IND_2 ; линии управления ЖКИ
          E            EQU      PORT_IND_3 ;-------------------------

          _M_SCL       EQU      P0_7    ;линия слеж за сост линии SCL I2C

;РОНы обслуживания ЖКИ и кнопок
          ADR_IND_R    DATA     030h    ;-----------------------------
          COM_IND_R    DATA     030h    ; РОНы обслуживания ЖКИ
          DATA_IND_R   DATA     031h    ;-----------------------------

          KNOP0_R      DATA     032h    ;-----------------------------
          KNOP1_R      DATA     033h    ;
          KNOP2_R      DATA     034h    ; регистры, содержащие
          KNOP3_R      DATA     035h    ;
          KNOP4_R      DATA     036h    ; флаги нажатия и удержания
          KNOP5_R      DATA     037h    ;
          KNOP6_R      DATA     038h    ; каждой кнопки
          KNOP7_R      DATA     039h    ;-----------------------------

          NAKOPL0_R    DATA     03Ah    ;-----------------------------
```

```
          NAKOPL1_R    DATA     03Bh    ;
          NAKOPL2_R    DATA     03Ch    ; регистры накопления
          NAKOPL3_R    DATA     03Dh    ;
          NAKOPL4_R    DATA     03Eh    ; значения подавления дребезга
          NAKOPL5_R    DATA     03Fh    ;
          NAKOPL6_R    DATA     040h    ; каждой кнопки
          NAKOPL7_R    DATA     041h    ;-----------------------------

;РОНы обслуживания подпрограмм преобразования формы представления чисел
          IND_MILL_R   DATA     04Eh    ;РОН миллиардов дес числа
          IND_SOTMIL_R DATA     04Fh    ;РОН сотен миллионов дес числа
          IND_DESMIL_R DATA     050h    ;РОН десятков миллионов дес числа
          IND_MIL_R    DATA     051h    ;РОН миллионов дес числа
          IND_SOTTIS_R DATA     052h    ;РОН сотен тысяч дес числа
          IND_DESTIS_R DATA     053h    ;РОН десятков тысяч дес числа
          IND_TIS_R    DATA     054h    ;РОН тысяч дес числа
          IND_SOT_R    DATA     055h    ;РОН сотен дес числа
          IND_DES_R    DATA     056h    ;РОН десятков дес числа
          IND_ED_R     DATA     057h    ;РОН единиц дес числа

          BYTE_0_R     DATA     05Bh    ;байт 0 упакованного двоичн-дес числа
          BYTE_1_R     DATA     05Ch    ;байт 1 упакованного двоичн-дес числа
          BYTE_2_R     DATA     05Dh    ;байт 2 упакованного двоичн-дес числа
          BYTE_3_R     DATA     05Eh    ;байт 3 упакованного двоичн-дес числа
          BYTE_4_R     DATA     05Fh    ;байт 4 упакованного двоичн-дес числа

;РОНы обслуживания интерфейса программный ведущий-I2C
       DATA_I2C_R   DATA     060h    ;РОН данных I2C
       SLUG_I2C_R   DATA     061h    ;РОН служ информ I2C (адр устр и призн зап)
       ADRL_I2C_R   DATA     062h    ;РОН мл байта адр ячейки в ведомом устр I2C
       ADRH_I2C_R   DATA     063h    ;РОН ст байта адр ячейки в ведомом устр I2C

;Флаги
          _ERR_I2C     BIT      07h     ;флаг "ошибка I2C"

;Константа обслуживания EEPROM I2C
       EE_I2C_K     EQU      10101110b ;служ слово обращ к EEPROM I2C, включает:
                                       ;тип устройства I2C (1010), номер устр
                                       ;I2C на шине (111) и признак записи (0)
;Прочие константы
          NACH_ADR     EQU      000h    ;начальный адрес обнуления РОНов
          KON_ADR      EQU      07Fh    ;конечный адрес обнуления РОНов
          POROG_K      EQU      50      ;порог подавления дребезга кнопок

;Начало исполняемого кода-------------------------------------------------------
                ORG 0h
           AJMP       Lab_START            ;идти на начало осн программы

;Начало осн программы-----------------------------------------------------------
                ORG 100h
Lab_START: MOV        SP,#080h             ;определить указатель стека
           MOV        PLLCON,#00000000b    ;уст макс частоту ядра (12,58 МГц)
           NOP                             ;
           LCALL      Pod_INIT_RSN         ;иниц РСН
           LCALL      Pod_INIT_RON         ;иниц РОН

         MOV        R5,#021h             ;задание нач значен индицируемого байта

           LCALL      Pod_INIT_LCD         ;иниц ЖКИ

           LCALL      Pod_CLEAR_LCD        ;стирание ЖКИ

           MOV        R0,#BYTE_0_R         ;
           MOV        R1,#ADRL_I2C_R       ;
           MOV        ADRH_I2C_R+1,#0      ; обнуление ст незначащих РОНов
           MOV        ADRH_I2C_R+2,#0      ;-----------------------------
           LCALL      B32BCD               ;преобр адреса из двоичн в двоичн-дес
```

```
            MOV         R0,#BYTE_0_R          ;
            MOV         R1,#IND_MILL_R        ;
            LCALL       BCD10BCD             ;преобр адреса из уп дв-дес в неуп дес

            MOV         R0,#IND_MILL_R        ; индицировать с РОНа IND_MILL_R
            MOV         R1,#0                 ; индицировать с адр 0 ЖКИ
            LCALL       Pod_IND_10ZN          ;индикация адреса

;Начало основного цикла-----------------------------------------------------------
La_OSN:     NOP                               ;метка возврата в осн цикле

;Блок, предшествующий вызову подпрограммы опроса кнопки 0.
            MOV         R0,#KNOP0_R           ;
            MOV         R1,#NAKOPL0_R         ;
            LCALL       Pod_OPR_KNOP0         ;
;Блок, следующий за вызовом подпрограммы опроса кнопки (уст факта ее наж)
            MOV         ACC,@R0               ;
            JNB         ACC_1,La_100          ;

            CLR         ACC_1                 ;кн была нажата, сброс флага нажат кн
            MOV         @R0,ACC               ;

            MOV         ACC,ADRL_I2C_R        ;---------------------------------
            ADD         A,#64                 ; Блок перебора адресов внутри
            JC          La_10                 ; адресного пространства 0 - 1FFFh
            MOV         ADRL_I2C_R,ACC        ; с шагом 64
            SJMP        La_50                 ;
La_10:      MOV         ADRL_I2C_R,ACC        ;
            INC         ADRH_I2C_R            ;
            MOV         ACC,ADRH_I2C_R        ;
            CJNE        A,#020h,La_50         ;
            MOV         ADRL_I2C_R,#0         ;
            MOV         ADRH_I2C_R,#0         ;
La_50:      NOP                               ;---------------------------------

            MOV         R0,#BYTE_0_R          ;
            MOV         R1,#ADRL_I2C_R        ;
            MOV         ADRH_I2C_R+1,#0       ; обнуление ст незначащих РОНов
            MOV         ADRH_I2C_R+2,#0       ;-----------------------------
            LCALL       B32BCD               ;преобр адреса из двоичн в двоичн-дес

            MOV         R0,#BYTE_0_R          ;
            MOV         R1,#IND_MILL_R        ;
            LCALL       BCD10BCD             ;преобр адреса из уп дв-дес в неуп дес

            MOV         R0,#IND_MILL_R        ; индицировать с РОНа IND_MILL_R
            MOV         R1,#0                 ; индицировать с адр 0 ЖКИ
            LCALL       Pod_IND_10ZN          ;индикация адреса

;Блок, предшествующий вызову подпрограммы опроса кнопки 1.
La_100:     MOV         R0,#KNOP1_R           ;
            MOV         R1,#NAKOPL1_R         ;
            LCALL       Pod_OPR_KNOP1         ;
;Блок, следующий за вызовом подпрограммы опроса кнопки (уст факта ее наж)
            MOV         ACC,@R0               ;
            JNB         ACC_1,La_200          ;

            CLR         ACC_1                 ;кн была нажата, сброс флага нажат кн
            MOV         @R0,ACC               ;

            INC         R5                    ;-----------------------------------
            CJNE        R5,#06Fh,La_00        ; перебор возможн знач индицир байта
            MOV         R5,#021h              ;-----------------------------------

La_00:      MOV         DATA_IND_R,R5         ;-----------------------------------
            MOV         ADR_IND_R,#64         ; индикация  байта (символа)
            ACALL       Pod_PER_DAT_LCD       ;-----------------------------------
```

```
;Блок, предшествующий вызову подпрограммы опроса кнопки 2.
La_200:    MOV        R0,#KNOP2_R          ;
           MOV        R1,#NAKOPL2_R        ;
           LCALL      Pod_OPR_KNOP2        ;
;Блок, следующий за вызовом подпрограммы опроса кнопки (уст факта ее наж)
           MOV        ACC,@R0              ;
           JNB        ACC_1,La_300         ;

           CLR        ACC_1                ;кн была нажата, сброс флага нажат кн
           MOV        @R0,ACC              ;

           MOV        SLUG_I2C_R,#EE_I2C_K ;подготовка служебной константы I2C
         MOV        DATA_I2C_R,R5        ;подготовка данных для зап в EEPROM I2C
           LCALL      Pod_WRITE_I2C        ;запись в EEPROM I2C

           MOV        DATA_IND_R,DATA_I2C_R;-----------------------------------
           MOV        ADR_IND_R,#70        ;индикация записанного байта (символа)
           ACALL      Pod_PER_DAT_LCD      ;-----------------------------------

;Блок, предшествующий вызову подпрограммы опроса кнопки 3.
La_300:    MOV        R0,#KNOP3_R          ;
           MOV        R1,#NAKOPL3_R        ;
           LCALL      Pod_OPR_KNOP3        ;
;Блок, следующий за вызовом подпрограммы опроса кнопки (уст факта ее наж)
           MOV        ACC,@R0              ;
           JNB        ACC_1,La_400         ;

           CLR        ACC_1                ;кн была нажата, сброс флага нажат кн
           MOV        @R0,ACC              ;

           MOV        SLUG_I2C_R,#EE_I2C_K ;подготовка служебной константы I2C
           MOV        DATA_I2C_R,#0FFh     ;контрольная порча РОНа данных I2C
           LCALL      Pod_READ_I2C         ;чтение из EEPROM I2C

           MOV        DATA_IND_R,DATA_I2C_R;-----------------------------------
           MOV        ADR_IND_R,#72        ;индикация прочитанного байта (символа)
           ACALL      Pod_PER_DAT_LCD      ;-----------------------------------

La_400:    LJMP       La_OSN               ;закрыть основной цикл


;Подпрограммы-------------------------------------------------------------------

;Подпрограммы организации обмена (записи и чтения по произвольному адресу)
;по шине I2C для мастер-устройства.

;-------------------------------------------------------------------------------
;Подпрограмма слежения за линией SCL интерфейса I2C. Подпрограмма возвращает
;управление, как только ведомый отпускает линию SCL (переводит ее в высокий
;уровень). Слежение производится через линию ввода-вывода общего назначения
;_M_SCL.
;-------------------------------------------------------------------------------
Pod_MON_SCL:
          JNB         _M_SCL,$             ;дожидаемся перехода SCL в выс уров
          RET

;-------------------------------------------------------------------------------
;Подпрограмма базового интервала I2C
;-------------------------------------------------------------------------------
Pod_DEL_I2C:
          NOP                              ; тело задержки
          RET

;-------------------------------------------------------------------------------
;Подпрограмма генерации условия START I2C
;-------------------------------------------------------------------------------
Pod_START:
          SETB        MDE                  ;SDA - выход
```

```
          CLR         MCO                 ;
          CALL        Pod_DEL_I2C         ;
          SETB        MDO                 ;
          CALL        Pod_DEL_I2C         ;
          SETB        MCO                 ;
          CALL        Pod_MON_SCL         ;дожидаемся перехода SCL в выс уров
          CLR         MDO                 ;START
          CALL        Pod_DEL_I2C         ;
          CLR         MCO                 ;
          RET

;-------------------------------------------------------------------------------
;Подпрограмма генерации условия STOP I2C
;-------------------------------------------------------------------------------
Pod_STOP: SETB        MDE                 ;SDA - выход
          CLR         MCO                 ;
          CALL        Pod_DEL_I2C         ;
          CLR         MDO                 ;
          CALL        Pod_DEL_I2C         ;
          SETB        MCO                 ;
          CALL        Pod_MON_SCL         ;дожидаемся перехода SCL в выс уров
          SETB        MDO                 ;STOP
          RET

;-------------------------------------------------------------------------------
;Подпрограмма опроса подтверждения на линии SDA I2C.
;В случае отсутствия подтверждения (выс уровень на линии SDA) устанавливается
;флаг ошибки шины I2C - _ERR_I2C.
;-------------------------------------------------------------------------------
Pod_ZACK: CLR         MDE                 ;SDA - вход
          SETB        MCO                 ;-----------------------------------
          CALL        Pod_MON_SCL         ;дожидаемся перехода SCL в выс уров
          JNB         MDI,Zack_0          ;есть ли подтверждение ?
          SETB        _ERR_I2C           ;подтверждения нет - уст флаг ошибк I2C
Zack_0:   CLR         MCO                 ;-----------------------------------
          RET

;-------------------------------------------------------------------------------
;Подпрограмма выдачи неподтверждения на линию SDA I2C.
;-------------------------------------------------------------------------------
Pod_NACK: SETB        MDE                 ;SDA - выход
          SETB        MDO                 ;
          CALL        Pod_DEL_I2C         ;
          SETB        MCO                 ;-----------------------------------
          CALL        Pod_MON_SCL         ;дожидаемся перехода SCL в выс уров
          CLR         MCO                 ;-----------------------------------
          RET

;-------------------------------------------------------------------------------
;Подпрограмма передачи байта по шине I2C. Передаваемый байт следует
;предварительно поместить в аккумулятор.  Используется счетчик циклов R3.
;Аккумулятор портится.
;-------------------------------------------------------------------------------
Pod_OUT_BYTE:
          SETB        MDE                 ;SDA - выход
          MOV         R3,#8               ;нач уст ст циклов
Out_0:    RLC         A                   ;сдвиг А влево через С, бит 7 идет в С
          MOV         MDO,C               ;передача бита
          CALL        Pod_DEL_I2C         ;
          SETB        MCO                 ;-----------------------------------
          CALL        Pod_MON_SCL         ;дожидаемся перехода SCL в выс уров
          CLR         MCO                 ;-----------------------------------
          DJNZ        R3,Out_0            ;
          RET

;-------------------------------------------------------------------------------
;Подпрограмма приема байта по шине I2C. Принятый байт оказывается в аккумуляторе.
;Используется счетчик циклов R3.
```

```
;-------------------------------------------------------------------------------
Pod_IN_BYTE:
          CLR         MDE                 ;SDA - вход
          MOV         R3,#8               ;нач уст ст циклов
In_0:     SETB        MCO                 ;----------------------------------
          CALL        Pod_MON_SCL         ;дожидаемся перехода SCL в выс уров
          MOV         C,MDI               ;прием бита
          CLR         MCO                 ;----------------------------------
          RLC         A                   ;сдвиг A влево через C, C идет в бит 0
          DJNZ        R3,In_0             ;
          RET

;-------------------------------------------------------------------------------
;Подпрограмма фиктивной записи (обращение с признаком записи по
;некоторому адресу ячейки без собственно записи) по шине I2C.
;-------------------------------------------------------------------------------
Pod_PWRITE_I2C:
          LCALL       Pod_START           ;START
          MOV         A,SLUG_I2C_R        ;  передача адр устройства
          LCALL       Pod_OUT_BYTE        ;  и признака записи
          LCALL       Pod_ZACK            ;запрос подтверждения
          MOV         A,ADRH_I2C_R        ;  передача ст байта адреса ячейки
          LCALL       Pod_OUT_BYTE        ;  в устройстве
          LCALL       Pod_ZACK            ;запрос подтверждения
          MOV         A,ADRL_I2C_R        ;  передача мл байта адреса ячейки
          LCALL       Pod_OUT_BYTE        ;  в устройстве
          LCALL       Pod_ZACK            ;запрос подтверждения
          RET

;-------------------------------------------------------------------------------
;Подпрограмма записи байта в ведомое устройство EEPROM по шине I2C.
;Предварительно следует поместить в РОНы ADRH_I2C_R, ADRL_I2C_R адрес ячейки,
;в РОН DATA_I2C_R - данные записи, в РОН SLUG_I2C_R - адрес устройства и признак
;операции (запись).
;-------------------------------------------------------------------------------
Pod_WRITE_I2C:
          LCALL       Pod_PWRITE_I2C      ;фиктивная запись - указание на ячейку
          MOV         A,DATA_I2C_R        ;  передача данных записи
          LCALL       Pod_OUT_BYTE        ;
          LCALL       Pod_ZACK            ;запрос подтверждения
          LCALL       Pod_STOP            ;STOP
          RET

;-------------------------------------------------------------------------------
;Подпрограмма чтения байта из ведомого устройства EEPROM по шине I2C.
;Предварит следует поместить в РОНы ADRH_I2C_R, ADRL_I2C_R адрес ячейки, а в
;РОН SLUG_I2C_R - адрес устройства и признак операции (запись).
;Прочитанный байт данных возвращается в РОНе DATA_I2C_R.
;-------------------------------------------------------------------------------
Pod_READ_I2C:
          LCALL       Pod_PWRITE_I2C      ;фиктивная запись - указание на ячейку
          LCALL       Pod_START           ;повторный START
          MOV         A,SLUG_I2C_R        ;
          SETB        ACC_0               ;уст признак чтения данных
          LCALL       Pod_OUT_BYTE       ;передача адреса устр и признака чтения
          LCALL       Pod_ZACK            ;запрос подтверждения
          LCALL       Pod_IN_BYTE         ;чтение байта данных из устройства
          LCALL       Pod_NACK            ;не даем подтверждения
          LCALL       Pod_STOP            ;STOP
          MOV         DATA_I2C_R,A        ;
          RET

;-------------------------------------------------------------------------------
;Подпрограмма инициализации РСН.
;-------------------------------------------------------------------------------
Pod_INIT_RSN:
          MOV        PORT_KNOP,#11111111b ;сделать вх все линии порта кн
          MOV        PORT_IND,#00000000b  ;сделать вых все линии порта индик
```

```
;настройка модуля I2C
          MOV       I2CCON,#10101000b    ;уст SDA и SCL, SDA - вход,
                                         ;уст режим ведущего
          RET

;-------------------------------------------------------------------------------
;Подпрограмма инициализации РОНов. Обнуляются все РОНы с адресами
;от NACH_ADR до KON_ADR при помощи косвенной адресации.
;-------------------------------------------------------------------------------
Pod_INIT_RON:
          MOV       R0,#NACH_ADR         ;установка начального адреса
Lk_0:     MOV       @R0,#0               ;обнуление очередного РОНа
          INC       R0                   ;переход к следующему адресу
          CJNE      R0,#KON_ADR,Lk_0     ;не достигли ли последнего адреса ?
          MOV       @R0,#0               ;обнуление последнего РОНа
          RET                            ;да, выход

;-------------------------------------------------------------------------------
;Подпрограмма вывода на ЖКИ содержимого буфера из 10 РОНов в виде 10 знакомест.
;R0 должен указывать на РОН в буфере, содержимое которого индицируется первым.
;R1 должен содержать нач адрес знакоместа в ОЗУ ЖКИ.
;-------------------------------------------------------------------------------
Pod_IND_10ZN:
           MOV       R2,#0               ;нач уст ст цикла
Ii_0:      MOV       DATA_IND_R,@R0      ;
           MOV       ADR_IND_R,R1        ;
           LCALL     Pod_PER_DAT_LCD     ; индикация очередного символа
           INC       R0                  ;
           INC       R1                  ;
           INC       R2                  ;
           CJNE      R2,#10,Ii_0         ;
           RET

;Подключение модулей опроса кнопок, вывода данных на ЖКИ (с опросом ЖКИ)
;и преобразования представления чисел
          $INCLUDE  (C:\PR_ADUC\I2C_EEP2\knop.asm)
          $INCLUDE  (C:\PR_ADUC\I2C_EEP2\lcd_opr.asm)
          $INCLUDE  (C:\PR_ADUC\I2C_EEP2\preobr.asm)

;Конец исполняемого кода
          END
```

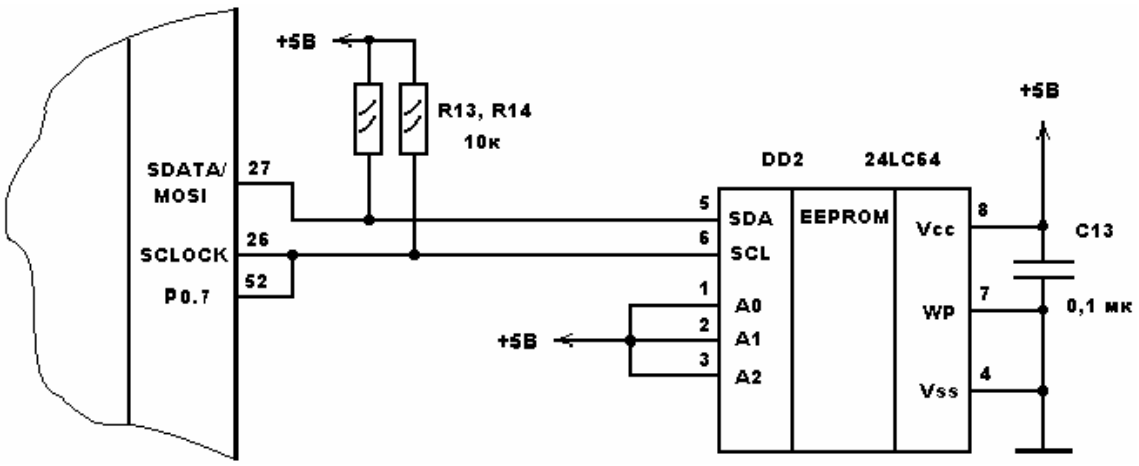


Рис. 3.14. Интерфейс I^2C с контролем состояния линий связи

Вывод SDA 24LC64 подключается к выводу 27 ADuC824 (SDATA), вывод SCL 24LC64 – к выводу 26 ADuC824 (SCLOCK) и к линии слежения за со-

стоянием SCL – выводу 52 ADuC824 (P 0.7). Предложенное решение обеспечивает отсутствие потерь данных при обмене, но для организации обмена по шине I^2C теперь приходится выделять не две, а три сигнальных линии микроконвертора. Кроме того, у пользователя в его приложении может возникнуть необходимость в одновременном использовании интерфейсов I^2C и SPI, а в данном примере это не представляется возможным, поскольку встроенные аппаратные модули этих интерфейсов разделяют одни и те же линии ADuC824.

Исходя из перечисленных соображений, был написан код, реализующий режим «ведущий» I^2C чисто программно без участия встроенного аппаратного модуля I2C. Исходный текст этой программы содержится в файле i2c_eep.asm (листинг 3.17). Функционально программа полностью идентична двум предыдущим. Фрагмент принципиальной схемы подключения EEPROM к ADuC824 для этого примера приведен на рис. 3.15.

Листинг 3.17. Программно реализуемый интерфейс I^2C

```
;-------------------------------------------------------------------------
;Демонстрационная программа организации обмена между микросхемой EEPROM
;24LC64 и ADuC824 по шине I2C.
;
;ADuC824 является мастер-устройством I2C.
;Протокол обмена по шине I2C реализован чисто программно. В качестве линий
;SDA и SCL можно использовать любые линии ввода-вывода общего назначения.
;Производится слежение за состоянием линии SCL.
;
;При нажатии на кнопку 0 производится перебор адресов ячеек для записи данных в
;EEPROM. Выбранное значение адреса ячейки в десятичном виде отображается на ЖКИ
;начиная с адреса 0 ОЗУ ЖКИ.
;
;При нажатии на кнопку 1 производится выбор байта данных для записи в EEPROM.
;Выбранный байт данных в ASCII коде  выводится на ЖКИ по адресу 64 ОЗУ ЖКИ.
;
;При нажатии на кнопку 2 производится запись выбранного байта данных в EEPROM
;по выбранному ранее адресу. Записанный байт данных выводится на ЖКИ по
;адресу 70 ОЗУ ЖКИ.
;
;При нажатии на кнопку 3 производится чтение байта данных из EEPROM по
;выбранному ранее адресу. Прочитанный байт данных в ASCII коде отображается на
;ЖКИ по адресу 72 ОЗУ ЖКИ.
;-------------------------------------------------------------------------------
          $INCLUDE  (C:\ADuC\mod824)
          $INCLUDE  (C:\PR_ADUC\I2C_EEP\824.inc)
;-------------------------------------------------------------------------
;Описание битов, регистров и констант
;-------------------------------------------------------------------------
;Порты и линии ввода-вывода
          PORT_KNOP    EQU      P0      ;порт кнопок

          PORT_IND     EQU      P2      ;порт индикации

          _IN_KNOP0    EQU      P0_0    ;-----------------------------
          _IN_KNOP1    EQU      P0_1    ;
          _IN_KNOP2    EQU      P0_2    ;  входы кнопок
          _IN_KNOP3    EQU      P0_3    ;
          _IN_KNOP4    EQU      P0_4    ;
          _IN_KNOP5    EQU      P0_5    ;
          _IN_KNOP6    EQU      P0_6    ;
          _IN_KNOP7    EQU      P0_7    ;-----------------------------

          PORT_IND_0   EQU      P2_0    ;-----------------------------
          PORT_IND_1   EQU      P2_1    ;
```

```
          PORT_IND_2   EQU      P2_2    ; выходы индикации
          PORT_IND_3   EQU      P2_3    ;
          PORT_IND_4   EQU      P2_4    ;
          PORT_IND_5   EQU      P2_5    ;
          PORT_IND_6   EQU      P2_6    ;
          PORT_IND_7   EQU      P2_7    ;------------------------------

          RW           EQU      PORT_IND_1 ;--------------------------
          RS           EQU      PORT_IND_2 ; линии управления ЖКИ
          E            EQU      PORT_IND_3 ;--------------------------

          _SDA         EQU      P0_6    ;линия данных  шины I2C
          _SCL         EQU      P0_7    ;линия синхронизации шины I2C

;РОНы обслуживания ЖКИ и кнопок
          ADR_IND_R    DATA     030h    ;------------------------------
          COM_IND_R    DATA     030h    ; РОНы обслуживания ЖКИ
          DATA_IND_R   DATA     031h    ;------------------------------

          KNOP0_R      DATA     032h    ;------------------------------
          KNOP1_R      DATA     033h    ;
          KNOP2_R      DATA     034h    ; регистры, содержащие
          KNOP3_R      DATA     035h    ;
          KNOP4_R      DATA     036h    ; флаги нажатия и удержания
          KNOP5_R      DATA     037h    ;
          KNOP6_R      DATA     038h    ; каждой кнопки
          KNOP7_R      DATA     039h    ;------------------------------

          NAKOPL0_R    DATA     03Ah    ;------------------------------
          NAKOPL1_R    DATA     03Bh    ;
          NAKOPL2_R    DATA     03Ch    ; регистры накопления
          NAKOPL3_R    DATA     03Dh    ;
          NAKOPL4_R    DATA     03Eh    ; значения подавления дребезга
          NAKOPL5_R    DATA     03Fh    ;
          NAKOPL6_R    DATA     040h    ; каждой кнопки
          NAKOPL7_R    DATA     041h    ;------------------------------

;РОНы обслуживания подпрограмм преобразования формы представления чисел
          IND_MILL_R   DATA     04Eh    ;РОН миллиардов дес числа
          IND_SOTMIL_R DATA     04Fh    ;РОН сотен миллионов дес числа
          IND_DESMIL_R DATA     050h    ;РОН десятков миллионов дес числа
          IND_MIL_R    DATA     051h    ;РОН миллионов дес числа
          IND_SOTTIS_R DATA     052h    ;РОН сотен тысяч дес числа
          IND_DESTIS_R DATA     053h    ;РОН десятков тысяч дес числа
          IND_TIS_R    DATA     054h    ;РОН тысяч дес числа
          IND_SOT_R    DATA     055h    ;РОН сотен дес числа
          IND_DES_R    DATA     056h    ;РОН десятков дес числа
          IND_ED_R     DATA     057h    ;РОН единиц дес числа

          BYTE_0_R     DATA     05Bh    ;байт 0 упакованного двоичн-дес числа
          BYTE_1_R     DATA     05Ch    ;байт 1 упакованного двоичн-дес числа
          BYTE_2_R     DATA     05Dh    ;байт 2 упакованного двоичн-дес числа
          BYTE_3_R     DATA     05Eh    ;байт 3 упакованного двоичн-дес числа
          BYTE_4_R     DATA     05Fh    ;байт 4 упакованного двоичн-дес числа

;РОНы обслуживания интерфейса программный ведущий-I2C
       DATA_I2C_R   DATA     060h    ;РОН данных I2C
       SLUG_I2C_R   DATA     061h    ;РОН служ информ I2C (адр устр и призн зап)
       ADRL_I2C_R   DATA     062h    ;РОН мл байта адр ячейки в ведомом устр I2C
       ADRH_I2C_R   DATA     063h    ;РОН ст байта адр ячейки в ведомом устр I2C

;Флаги
          _ERR_I2C     BIT      07h     ;флаг "ошибка I2C"

;Константа обслуживания EEPROM I2C
        EE_I2C_K     EQU     10101110b ;служ слово обращ к EEPROM I2C, включает:
                                       ;тип устройства I2C (1010), номер устр
                                       ;I2C на шине (111) и признак записи (0)
```

```
;Прочие константы
          NACH_ADR     EQU      000h    ;начальный адрес обнуления РОНов
          KON_ADR      EQU      07Fh    ;конечный адрес обнуления РОНов
          POROG_K      EQU      50      ;порог подавления дребезга кнопок
;Начало исполняемого кода-------------------------------------------------------
               ORG 0h
           AJMP       Lab_START           ;идти на начало осн программы


;Начало осн программы-----------------------------------------------------------
               ORG 100h
Lab_START: MOV        SP,#080h            ;определить указатель стека
           MOV        PLLCON,#00000000b   ;уст макс частоту ядра (12,58 МГц)
           NOP                            ;
           LCALL      Pod_INIT_RSN        ;иниц РСН
           LCALL      Pod_INIT_RON        ;иниц РОН

           MOV        R5,#021h           ;задание нач значен индицируемого байта

           LCALL      Pod_INIT_LCD        ;иниц ЖКИ

           LCALL      Pod_CLEAR_LCD       ;стирание ЖКИ

           MOV        R0,#BYTE_0_R        ;
           MOV        R1,#ADRL_I2C_R      ;
           MOV        ADRH_I2C_R+1,#0     ; обнуление ст незначащих РОНов
           MOV        ADRH_I2C_R+2,#0     ;------------------------------
           LCALL      B32BCD             ;преобр адреса из двоичн в двоичн-дес

           MOV        R0,#BYTE_0_R        ;
           MOV        R1,#IND_MILL_R      ;
           LCALL      BCD10BCD           ;преобр адреса из уп дв-дес в неуп дес

           MOV        R0,#IND_MILL_R      ; индицировать с РОНа IND_MILL_R
           MOV        R1,#0               ; индицировать с адр 0 ЖКИ
           LCALL      Pod_IND_10ZN        ;индикация адреса

;Начало основного цикла---------------------------------------------------------
La_OSN:    NOP                            ;метка возврата в осн цикле

;Блок, предшествующий вызову подпрограммы опроса кнопки 0.
           MOV        R0,#KNOP0_R         ;
           MOV        R1,#NAKOPL0_R       ;
           LCALL      Pod_OPR_KNOP0       ;
;Блок, следующий за вызовом подпрограммы опроса кнопки (уст факта ее наж)
           MOV        ACC,@R0             ;
           JNB        ACC_1,La_100        ;

           CLR        ACC_1               ;кн была нажата, сброс флага нажат кн
           MOV        @R0,ACC             ;

           MOV        ACC,ADRL_I2C_R      ;---------------------------------
           ADD        A,#64               ; Блок перебора адресов внутри
           JC         La_10               ; адресного пространства 0 - 1FFFh
           MOV        ADRL_I2C_R,ACC      ; с шагом 64
           SJMP       La_50               ;
La_10:     MOV        ADRL_I2C_R,ACC      ;
           INC        ADRH_I2C_R          ;
           MOV        ACC,ADRH_I2C_R      ;
           CJNE       A,#020h,La_50       ;
           MOV        ADRL_I2C_R,#0       ;
           MOV        ADRH_I2C_R,#0       ;
La_50:     NOP                            ;---------------------------------

           MOV        R0,#BYTE_0_R        ;
           MOV        R1,#ADRL_I2C_R      ;
           MOV        ADRH_I2C_R+1,#0     ; обнуление ст незначащих РОНов
           MOV        ADRH_I2C_R+2,#0     ;------------------------------
```

```
            LCALL       B32BCD              ;преобр адреса из двоичн в двоичн-дес

            MOV         R0,#BYTE_0_R        ;
            MOV         R1,#IND_MILL_R      ;
            LCALL       BCD10BCD           ;преобр адреса из уп дв-дес в неуп дес

            MOV         R0,#IND_MILL_R      ; индицировать с РОНа IND_MILL_R
            MOV         R1,#0               ; индицировать с адр 0 ЖКИ
            LCALL       Pod_IND_10ZN        ;индикация адреса

;Блок, предшествующий вызову подпрограммы опроса кнопки 1.
La_100:     MOV         R0,#KNOP1_R         ;
            MOV         R1,#NAKOPL1_R       ;
            LCALL       Pod_OPR_KNOP1       ;
;Блок, следующий за вызовом подпрограммы опроса кнопки (уст факта ее наж)
            MOV         ACC,@R0             ;
            JNB         ACC_1,La_200        ;

            CLR         ACC_1               ;кн была нажата, сброс флага нажат кн
            MOV         @R0,ACC             ;

            INC         R5                  ;-----------------------------------
            CJNE        R5,#06Fh,La_00      ; перебор возможн знач индицир байта
            MOV         R5,#021h            ;-----------------------------------

La_00:      MOV         DATA_IND_R,R5       ;-----------------------------------
            MOV         ADR_IND_R,#64       ; индикация  байта (символа)
            ACALL       Pod_PER_DAT_LCD     ;-----------------------------------

;Блок, предшествующий вызову подпрограммы опроса кнопки 2.
La_200:     MOV         R0,#KNOP2_R         ;
            MOV         R1,#NAKOPL2_R       ;
            LCALL       Pod_OPR_KNOP2       ;
;Блок, следующий за вызовом подпрограммы опроса кнопки (уст факта ее наж)
            MOV         ACC,@R0             ;
            JNB         ACC_1,La_300        ;

            CLR         ACC_1               ;кн была нажата, сброс флага нажат кн
            MOV         @R0,ACC             ;

            MOV         SLUG_I2C_R,#EE_I2C_K ;подготовка служебной константы I2C
            MOV         DATA_I2C_R,R5       ;подготовка данных для зап в EEPROM I2C
            LCALL       Pod_WRITE_I2C       ;запись в EEPROM I2C

            MOV         DATA_IND_R,DATA_I2C_R;-----------------------------------
            MOV         ADR_IND_R,#70       ; индикация записанного байта (символа)
            ACALL       Pod_PER_DAT_LCD     ;-----------------------------------

;Блок, предшествующий вызову подпрограммы опроса кнопки 3.
La_300:     MOV         R0,#KNOP3_R         ;
            MOV         R1,#NAKOPL3_R       ;
            LCALL       Pod_OPR_KNOP3       ;
;Блок, следующий за вызовом подпрограммы опроса кнопки (уст факта ее наж)
            MOV         ACC,@R0             ;
            JNB         ACC_1,La_400        ;

            CLR         ACC_1               ;кн была нажата, сброс флага нажат кн
            MOV         @R0,ACC             ;

            MOV         SLUG_I2C_R,#EE_I2C_K ;подготовка служебной константы I2C
            MOV         DATA_I2C_R,#0FFh    ;контрольная порча РОНа данных I2C
            LCALL       Pod_READ_I2C        ;чтение из EEPROM I2C

            MOV         DATA_IND_R,DATA_I2C_R;-----------------------------------
            MOV         ADR_IND_R,#72       ; индикация прочит байта (символа)
            ACALL       Pod_PER_DAT_LCD     ;-----------------------------------

La_400:     LJMP        La_OSN              ;закрыть основной цикл
```

```
;Подпрограммы-----------------------------------------------------------------

;Подпрограммы организации обмена (записи и чтения по произвольному адресу)
;по шине I2C для мастер-устройства.
;-------------------------------------------------------------------------------
;Подпрограмма базового интервала (полупериода сигнала SCL) I2C
;-------------------------------------------------------------------------------
Pod_DEL_I2C:
          NOP                           ;тело задержки
          RET
;-------------------------------------------------------------------------------
;Подпрограмма отрицательного перепада на линии SCL I2C
;-------------------------------------------------------------------------------
Pod_LOW_SCL:
          CLR         _SCL              ;спад SCL
          LCALL       Pod_DEL_I2C       ;пауза
          RET

;-------------------------------------------------------------------------------
;Подпрограмма отрицательного перепада на линии SDA I2C
;-------------------------------------------------------------------------------
Pod_LOW_SDA:
          CLR         _SDA              ;спад SDA
          LCALL       Pod_DEL_I2C       ;пауза
          RET

;-------------------------------------------------------------------------------
;Подпрограмма положительного перепада на линии SCL I2C
;-------------------------------------------------------------------------------
Pod_HIGH_SCL:
          SETB        _SCL              ;фронт SCL
          LCALL       Pod_DEL_I2C       ;пауза
          RET

;-------------------------------------------------------------------------------
;Подпрограмма положительного перепада на линии SDA I2C
;-------------------------------------------------------------------------------
Pod_HIGH_SDA:
          SETB        _SDA              ;фронт SDA
          LCALL       Pod_DEL_I2C       ;пауза
          RET

;-------------------------------------------------------------------------------
;Подпрограмма положительного импульса на линии SCL I2C
;-------------------------------------------------------------------------------
Pod_CLOCK_PULSE:
          LCALL       Pod_HIGH_SCL      ;фронт SCL
          JNB         _SCL,$            ;дожидаемся перехода SCL в выс уров
          LCALL       Pod_LOW_SCL       ;спад SCL
          RET

;-------------------------------------------------------------------------------
;Подпрограмма генерации условия START I2C
;-------------------------------------------------------------------------------
Pod_START:
          LCALL       Pod_LOW_SCL       ;
          LCALL       Pod_HIGH_SDA      ;
          LCALL       Pod_HIGH_SCL      ;
          LCALL       Pod_LOW_SDA       ;START
          LCALL       Pod_LOW_SCL       ;
          RET

;-------------------------------------------------------------------------------
;Подпрограмма генерации условия STOP I2C
;-------------------------------------------------------------------------------
Pod_STOP: LCALL       Pod_LOW_SCL       ;
```

```
          LCALL       Pod_LOW_SDA         ;
          LCALL       Pod_HIGH_SCL        ;
          LCALL       Pod_HIGH_SDA        ;STOP
          RET

;-------------------------------------------------------------------------------
;Подпрограмма опроса подтверждения на линии SDA I2C.
;В случае отсутствия подтверждения (выс уровень на линии SDA) устанавливается
;флаг ошибки шины I2C - _ERR_I2C.
;-------------------------------------------------------------------------------
Pod_ZACK: LCALL       Pod_HIGH_SDA        ;SDA - вход
          LCALL       Pod_HIGH_SCL        ;фронт SCL
          JNB         _SCL,$              ;дожидаемся перехода SCL в выс уров
          JNB         _SDA,Zack_0         ;есть ли подтверждение ?
          SETB        _ERR_I2C           ;подтверждения нет - уст флаг ошибк I2C
Zack_0:   LCALL       Pod_LOW_SCL         ;спад SCL
          RET

;-------------------------------------------------------------------------------
;Подпрограмма выдачи неподтверждения на линию SDA I2C.
;-------------------------------------------------------------------------------
Pod_NACK: LCALL       Pod_HIGH_SDA        ;фронт SDA
          LCALL       Pod_CLOCK_PULSE     ;импульс SCL
          RET

;-------------------------------------------------------------------------------
;Подпрограмма передачи байта по шине I2C. Передаваемый байт следует
;предварительно поместить в аккумулятор.  Используется счетчик циклов R3.
;Аккумулятор портится.
;-------------------------------------------------------------------------------
Pod_OUT_BYTE:
          MOV         R3,#8               ;нач уст ст циклов
Out_0:    RLC         A                   ;сдвиг А влево через С, бит 7 идет в С
          JC          Out_10              ;в зависимости от значения С:
          LCALL       Pod_LOW_SDA         ;передача 0
          SJMP        Out_20              ;
Out_10:   LCALL       Pod_HIGH_SDA        ;передача 1
Out_20:   LCALL       Pod_CLOCK_PULSE     ;импульс SCL
          DJNZ        R3,Out_0            ;
          RET

;-------------------------------------------------------------------------------
;Подпрограмма приема байта по шине I2C. Принятый байт оказывается в аккумуляторе.
;Используется счетчик циклов R3.
;-------------------------------------------------------------------------------
Pod_IN_BYTE:
          MOV         R3,#8               ;нач уст ст циклов
          LCALL       Pod_HIGH_SDA        ;SDA - вход
In_0:     LCALL       Pod_HIGH_SCL        ;фронт SCL
          JNB         _SCL,$              ;дожидаемся перехода SCL в выс уров
          MOV         C,_SDA              ; прием бита
          LCALL       Pod_LOW_SCL         ;спад SCL
          RLC         A                   ;сдвиг А влево через С, С идет в бит 0
          DJNZ        R3,In_0             ;
          RET

;-------------------------------------------------------------------------------
;Подпрограмма фиктивной записи (обращение с признаком записи по
;некоторому адресу ячейки без собственно записи) по шине I2C.
;-------------------------------------------------------------------------------
Pod_PWRITE_I2C:
          LCALL       Pod_START           ;START
          MOV         A,SLUG_I2C_R        ;  передача адр устройства
          LCALL       Pod_OUT_BYTE        ;  и признака записи
          LCALL       Pod_ZACK            ;запрос подтверждения
          MOV         A,ADRH_I2C_R        ;  передача ст байта адреса ячейки
          LCALL       Pod_OUT_BYTE        ;  в устройстве
          LCALL       Pod_ZACK            ;запрос подтверждения
```

```
          MOV         A,ADRL_I2C_R        ;  передача мл байта адреса ячейки
          LCALL       Pod_OUT_BYTE        ;  в устройстве
          LCALL       Pod_ZACK            ;запрос подтверждения
          RET

;-------------------------------------------------------------------------------
;Подпрограмма записи байта в ведомое устройство EEPROM по шине I2C.
;Предварительно следует поместить в РОНы ADRH_I2C_R, ADRL_I2C_R адрес ячейки,
;в РОН DATA_I2C_R - данные записи, в РОН SLUG_I2C_R - адрес устройства и признак
;операции (запись).
;-------------------------------------------------------------------------------
Pod_WRITE_I2C:
          LCALL       Pod_PWRITE_I2C      ;фиктивная запись - указание на ячейку
          MOV         A,DATA_I2C_R        ;  передача данных записи
          LCALL       Pod_OUT_BYTE        ;
          LCALL       Pod_ZACK            ;запрос подтверждения
          LCALL       Pod_STOP            ;STOP
          RET

;-------------------------------------------------------------------------------
;Подпрограмма чтения байта из ведомого устройства EEPROM по шине I2C.
;Предварит следует поместить в РОНы ADRH_I2C_R, ADRL_I2C_R адрес ячейки, а в
;РОН SLUG_I2C_R - адрес устройства и признак операции (запись).
;Прочитанный байт данных возвращается в РОНе DATA_I2C_R.
;-------------------------------------------------------------------------------
Pod_READ_I2C:
          LCALL       Pod_PWRITE_I2C      ;фиктивная запись - указание на ячейку
          LCALL       Pod_START           ;повторный START
          MOV         A,SLUG_I2C_R        ;
          SETB        ACC_0               ;уст признак чтения данных
          LCALL       Pod_OUT_BYTE      ;передача адреса устр и признака чтения
          LCALL       Pod_ZACK            ;запрос подтверждения
          LCALL       Pod_IN_BYTE         ;чтение байта данных из устройства
          LCALL       Pod_NACK            ;не даем подтверждения
          LCALL       Pod_STOP            ;STOP
          MOV         DATA_I2C_R,A        ;
          RET

;-------------------------------------------------------------------------------
;Подпрограмма инициализации PCH.
;-------------------------------------------------------------------------------
Pod_INIT_RSN:
          MOV        PORT_KNOP,#11111111b ;сделать вх все линии порта кн
          MOV        PORT_IND,#00000000b  ;сделать вых все линии порта индик
          RET

;-------------------------------------------------------------------------------
;Подпрограмма инициализации РОНов. Обнуляются все РОНы с адресами
;от NACH_ADR до KON_ADR при помощи косвенной адресации.
;-------------------------------------------------------------------------------
Pod_INIT_RON:
          MOV        R0,#NACH_ADR         ;установка начального адреса
Lk_0:     MOV        @R0,#0               ;обнуление очередного РОНа
          INC        R0                   ;переход к следующему адресу
          CJNE       R0,#KON_ADR,Lk_0     ;не достигли ли последнего адреса ?
          MOV        @R0,#0               ;обнуление последнего РОНа
          RET                             ;да, выход

;-------------------------------------------------------------------------------
;Подпрограмма вывода на ЖКИ содержимого буфера из 10 РОНов в виде 10 знакомест.
;R0 должен указывать на РОН в буфере, содержимое которого индицируется первым.
;R1 должен содержать нач адрес знакоместа в ОЗУ ЖКИ.
;-------------------------------------------------------------------------------
Pod_IND_10ZN:
           MOV        R2,#0                ;нач уст ст цикла
Ii_0:      MOV        DATA_IND_R,@R0       ;
```

```
            MOV         ADR_IND_R,R1          ;
            LCALL       Pod_PER_DAT_LCD       ; индикация очередного символа
            INC         R0                    ;
            INC         R1                    ;
            INC         R2                    ;
            CJNE        R2,#10,Ii_0           ;
            RET

;Подключение модулей опроса кнопок, вывода данных на ЖКИ (с опросом ЖКИ)
;и преобразования представления чисел
           $INCLUDE   (C:\PR_ADUC\I2C_EEP\knop.asm)
           $INCLUDE   (C:\PR_ADUC\I2C_EEP\lcd_opr.asm)
           $INCLUDE   (C:\PR_ADUC\I2C_EEP\preobr.asm)

;Конец исполняемого кода
           END
```

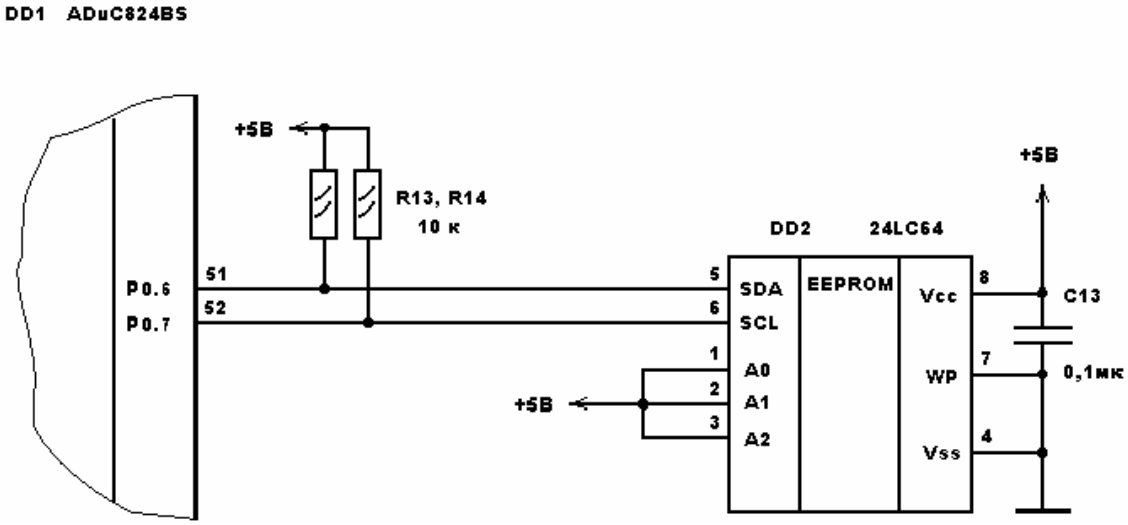


Рис. 3.15. Программно реализуемый интерфейс I^2C

Вывод SDA 24LC64 подключается к выводу 51 ADuC824 (P 0.6), вывод SCL 24LC64 – к выводу 52 (P 0.7). Слежение за состоянием линии SCL реализовано программно. В качестве линий SDA и SCL в микроконверторе можно использовать и любые другие линии ввода-вывода общего назначения, соответствующим образом описав их в заголовке исходного текста программы, следовательно, предложенный интерфейс I2C может быть использован в любом 51-совместимом микроконтроллере. К недостаткам интерфейса можно отнести несколько меньшую, чем в предыдущих примерах скорость обмена по шине.

В случае необходимости реализации блочной записи или чтения I^2C EEPROM (по текущему адресу) читатели могут попробовать самостоятельно написать соответствующий пользовательский код, что совсем несложно сделать, пользуясь предложенными подпрограммами и описаниями производителей I^2C EEPROM.

Использование ADuC824 как «ведомого» устройства I^2C, очевидно, подразумевает применение микроконвертора в качестве «интеллектуального» датчика при каком-то центральном процессоре обработки собранных данных. Практическая реализация режима ADuC824 «аппаратный ведомый» I^2C иллю-

стрируется программой, исходный текст которой содержится в файле i2c_slav.asm (листинг 3.18).

Листинг 3.18. Реализация режима «аппаратный ведомый»

```
;-------------------------------------------------------------------------
;Демонстрационная программа организации обмена между ADuC824 и центральным
;процессором (AT89C2051) по шине I2C.
;
;ADuC824 является ведомым устройством I2C.
;Протокол обмена по шине I2C реализован аппаратно.
;(Используется встроенный аппаратный модуль I2C в режиме "аппаратный ведомый").
;
;Передача данных в центральный процессор и прием данных из него производятся
;только по инициативе центрального процессора и тактируются им.
;
;При нажатии на кнопку 0 производится выбор байта данных для передачи в
;центральный процессор. Выбранный байт данных в ASCII коде  выводится на ЖКИ
;по адресу 0 ОЗУ ЖКИ.
;
;Принятый из центрального процессора байт данных в ASCII коде отображается на
;ЖКИ по адресу 8 ОЗУ ЖКИ.
;-------------------------------------------------------------------------------
          $INCLUDE  (C:\ADuC\mod824)
          $INCLUDE  (C:\PR_ADUC\I2C_SLAV\824.inc)
;-------------------------------------------------------------------------
;Описание битов, регистров и констант
;-------------------------------------------------------------------------
;Порты и линии ввода-вывода
          PORT_KNOP    EQU      P0      ;порт кнопок

          PORT_IND     EQU      P2      ;порт индикации

          _IN_KNOP0    EQU      P0_0    ;-----------------------------
          _IN_KNOP1    EQU      P0_1    ;
          _IN_KNOP2    EQU      P0_2    ;  входы кнопок
          _IN_KNOP3    EQU      P0_3    ;
          _IN_KNOP4    EQU      P0_4    ;
          _IN_KNOP5    EQU      P0_5    ;
          _IN_KNOP6    EQU      P0_6    ;
          _IN_KNOP7    EQU      P0_7    ;-----------------------------

          PORT_IND_0   EQU      P2_0    ;-----------------------------
          PORT_IND_1   EQU      P2_1    ;
          PORT_IND_2   EQU      P2_2    ; выходы индикации
          PORT_IND_3   EQU      P2_3    ;
          PORT_IND_4   EQU      P2_4    ;
          PORT_IND_5   EQU      P2_5    ;
          PORT_IND_6   EQU      P2_6    ;
          PORT_IND_7   EQU      P2_7    ;-----------------------------

          RW           EQU      PORT_IND_1 ;-------------------------
          RS           EQU      PORT_IND_2 ; линии управления ЖКИ
          E            EQU      PORT_IND_3 ;-------------------------

;РОНы обслуживания ЖКИ и кнопок
          ADR_IND_R    DATA     030h    ;-----------------------------
          COM_IND_R    DATA     030h    ; РОНы обслуживания ЖКИ
          DATA_IND_R   DATA     031h    ;-----------------------------

          KNOP0_R      DATA     032h    ;-----------------------------
          KNOP1_R      DATA     033h    ;
          KNOP2_R      DATA     034h    ; регистры, содержащие
          KNOP3_R      DATA     035h    ;
          KNOP4_R      DATA     036h    ; флаги нажатия и удержания
          KNOP5_R      DATA     037h    ;
          KNOP6_R      DATA     038h    ; каждой кнопки
          KNOP7_R      DATA     039h    ;-----------------------------
```

```
          NAKOPL0_R    DATA     03Ah    ;----------------------------
          NAKOPL1_R    DATA     03Bh    ;
          NAKOPL2_R    DATA     03Ch    ; регистры накопления
          NAKOPL3_R    DATA     03Dh    ;
          NAKOPL4_R    DATA     03Eh    ; значения подавления дребезга
          NAKOPL5_R    DATA     03Fh    ;
          NAKOPL6_R    DATA     040h    ; каждой кнопки
          NAKOPL7_R    DATA     041h    ;----------------------------

;РОНы обслуживания интерфейса программный ведущий-I2C
        OUT_I2C_R    DATA     060h    ;РОН хранения байта для передачи ведущему
                                      ;по I2C
        IN_I2C_R     DATA     061h    ;РОН хранения байта, принятого от ведущего
                                      ;по I2C
;Флаги
          _DAT_I2C     BIT      08h     ;флаг "принят байт данных по I2C"
          _ADR_I2C     BIT      09h     ;флаг "принят байт адр устр по I2C"

;Константа - адрес ведомого устройства I2C
          ADRSL_I2C_K  EQU      01000100b ;

;Прочие константы
          NACH_ADR     EQU      000h    ;начальный адрес обнуления РОНов
          KON_ADR      EQU      07Fh    ;конечный адрес обнуления РОНов
          POROG_K      EQU      50      ;порог подавления дребезга кнопок

;Начало исполняемого кода-------------------------------------------------------
                ORG 0h
            AJMP       Lab_START            ;идти на начало осн программы

                ORG 03Bh
            AJMP       Lab_I2C              ;идти на обработку прерываний от I2C

;Блок обработки прерываний от модуля I2C.
;Путем опроса бита I2CTX ведомый устанавливает, какой режим выбран ведущим -
;прием или передача. Если выбран режим приема и правильный адрес ведомого уже
;принят, ведомый ожидает следующего прерывания от I2C, принимает байт данных
;и устанавливает флаг "принят байт данных по I2C" _DAT_I2C. Обработка (в данном
;случае - индикация принятого байта)  и сброс этого флага возлагаются на
;основную программу.
;Если выбран режим передачи, ведомый передает ведущему
;байт данных, значение которого можно предварительно определить в основной
;программе. После этого интерфейс I2C сбрасывается в состояние ожидания адреса.
;Прерывания должны быть разрешены.
Lab_I2C:    CLR        EA                   ;----------------------------------
            PUSH       PSW                  ; глоб запрет прер и сохр контекста
            PUSH       ACC                  ;----------------------------------

            JB         I2CTX,Lab_Trans      ;какой режим выбран ведущим -
                                            ;прием или передача ?

;Блок обработки приема байта от ведущего по I2C
            JNB         _ADR_I2C,Lab_R1     ;принят ли адрес ведомого ?

            MOV         IN_I2C_R,I2CDAT     ;да, сохр принятого от ведущего байта
            SETB        _DAT_I2C            ;уст флаг "принят байт данных по I2C"
            CLR         _ADR_I2C           ;сбр флаг "принят байт адр устр по I2C"
            JMP        Lab_RETI             ;идти на возврат из прерывания

Lab_R1:     SETB       _ADR_I2C            ;уст флаг "принят байт адр устр по I2C"
            JMP        Lab_RETI             ;идти на возврат из прерывания

;Блок обработки передачи байта ведущему по I2C
Lab_Trans: MOV         I2CDAT,OUT_I2C_R     ;передать ведущему байт

;Блок возврата из прерываний----------------------------------------------------
Lab_RETI:  POP         ACC                  ;----------------------------------
```

```
           POP        PSW                  ; восст контекста и глоб разр прер
           SETB       EA                   ;---------------------------------
           RETI                            ;возврат из блока обраб прерываний

;Начало осн программы-------------------------------------------------------
                ORG 100h
Lab_START: MOV        SP,#080h             ;определить указатель стека
           MOV        PLLCON,#00000000b    ;уст макс частоту ядра (12,58 МГц)
           NOP                             ;
           LCALL      Pod_INIT_RSN         ;иниц РСН
           LCALL      Pod_INIT_RON         ;иниц РОН

           LCALL      Pod_INIT_LCD         ;иниц ЖКИ

           LCALL      Pod_CLEAR_LCD        ;стирание ЖКИ

           MOV        R5,#021h             ; задание нач значения байта для
           MOV        OUT_I2C_R,R5         ; передачи ведущему по I2C

           MOV        DATA_IND_R,R5        ;---------------------------------
           MOV        ADR_IND_R,#0         ; индикация  байта (символа)
           ACALL      Pod_PER_DAT_LCD      ;---------------------------------

;Начало основного цикла-----------------------------------------------------
La_OSN:    NOP                             ;метка возврата в осн цикле

;Блок, предшествующий вызову подпрограммы опроса кнопки 0.
           MOV        R0,#KNOP0_R          ;
           MOV        R1,#NAKOPL0_R        ;
           LCALL      Pod_OPR_KNOP0        ;
;Блок, следующий за вызовом подпрограммы опроса кнопки (уст факта ее наж)
           MOV        ACC,@R0              ;
           JNB        ACC_1,La_100         ;

           CLR        ACC_1                ;кн была нажата, сброс флага нажат кн
           MOV        @R0,ACC              ;

           INC        R5                   ;---------------------------------
           CJNE       R5,#06Fh,La_00       ; перебор возможн знач индицир байта
           MOV        R5,#021h             ;---------------------------------

La_00:     MOV        OUT_I2C_R,R5        ;подгот байта для перед в ведущ по I2C

           MOV        DATA_IND_R,R5        ;---------------------------------
           MOV        ADR_IND_R,#0         ; индикация  байта (символа)
           ACALL      Pod_PER_DAT_LCD      ;---------------------------------

La_100:    JNB        _DAT_I2C,La_OSN   ;не было ли приема байта от ведущего I2C

           CLR        _DAT_I2C         ;да, сбр фл "принят байт адр устр по I2C"
           MOV        DATA_IND_R,IN_I2C_R  ;-------------------------------
           MOV        ADR_IND_R,#8         ; индикация принятого байта (символа)
           ACALL      Pod_PER_DAT_LCD      ;---------------------------------

           LJMP       La_OSN               ;закрыть основной цикл


;Подпрограммы---------------------------------------------------------------

;-------------------------------------------------------------------------
;Подпрограмма инициализации РСН.
;-------------------------------------------------------------------------
Pod_INIT_RSN:
          MOV        PORT_KNOP,#11111111b ;сделать вх все линии порта кн
          MOV        PORT_IND,#00000000b  ;сделать вых все линии порта индик

;настройка модуля I2C
          MOV        IEIP2,#00000001b     ;разрешить прерывания от I2C
```

```
          MOV       IE,#10000000b       ;разрешить прерывания глобально

          MOV       I2CADD,#ADRSL_I2C_K ;определить адрес ведомого устр I2C
          MOV       I2CCON,#00000000b   ;уст режим ведомого I2C
          RET

;-------------------------------------------------------------------------------
;Подпрограмма инициализации РОНов. Обнуляются все РОНы с адресами
;от NACH_ADR до KON_ADR при помощи косвенной адресации.
;-------------------------------------------------------------------------------
Pod_INIT_RON:
          MOV       R0,#NACH_ADR        ;установка начального адреса
Lk_0:     MOV       @R0,#0              ;обнуление очередного РОНа
          INC       R0                  ;переход к следующему адресу
          CJNE      R0,#KON_ADR,Lk_0    ;не достигли ли последнего адреса ?
          MOV       @R0,#0              ;обнуление последнего РОНа
          RET                           ;да, выход

;Подключение модулей опроса кнопок и вывода данных на ЖКИ (с опросом ЖКИ)
          $INCLUDE  (C:\PR_ADUC\I2C_SLAV\knop.asm)
          $INCLUDE  (C:\PR_ADUC\I2C_SLAV\lcd_opr.asm)

;Конец исполняемого кода
          END
```

В отличие от предыдущих примеров, программа использует прерывания от модуля I2C ADuC824. Пользовательский интерфейс программы позволяет путем нажатий на кнопку «0» выбрать из некоторого множества и индицировать по адресу 0 ОЗУ ЖКИ, подключенный к микроконвертору, байт данных, предназначенный для передачи по шине I^2C в центральный процессор (но не осуществить саму передачу). По адресу 8 ОЗУ ЖКИ индицируется последний байт данных, поступивший из центрального процессора. Обмен данными ADuC824 и центрального процессора производится только по инициативе и тактовым сигналам последнего. Как и в случае реализации режима «программный ведущий», ADuC824 подключается к шине I^2C как это показано на рис. 3.12. Собственный адрес «ведомого» устройства на шине задается в данном случае не аппаратно, а программно – путем записи константы адреса (в программе она имеет имя ADRSL_I2C_K) в специальный регистр хранения адреса «ведомого» I2CADD в подпрограмме Pod_INIT_RSN.

В качестве «ведущего» устройства I^2C (центрального процессора) использовался 51-совместимый микроконтроллер фирмы Atmel AT89C2051 [15]. На основе AT89C2051 необходимо собрать макет (рис. 3.16).

Питание макета можно осуществлять от источника питания эволюционной платы ADuC824. Как можно видеть из схемы макета, к центральному процессору, также как и к ADuC824, подключен ЖКИ и кнопки управления, что дает возможность пользователю выбирать подлежащий передаче в ADuC824 байт данных, инициировать передачу и прием с индикацией переданного в ADuC824 и принятого из ADuC824 байтов. Исходный текст программы, реализующей обмен по шине I^2C со стороны центрального процессора AT89C2051, содержится в файле i2c_mast.asm (листинг 3.19).

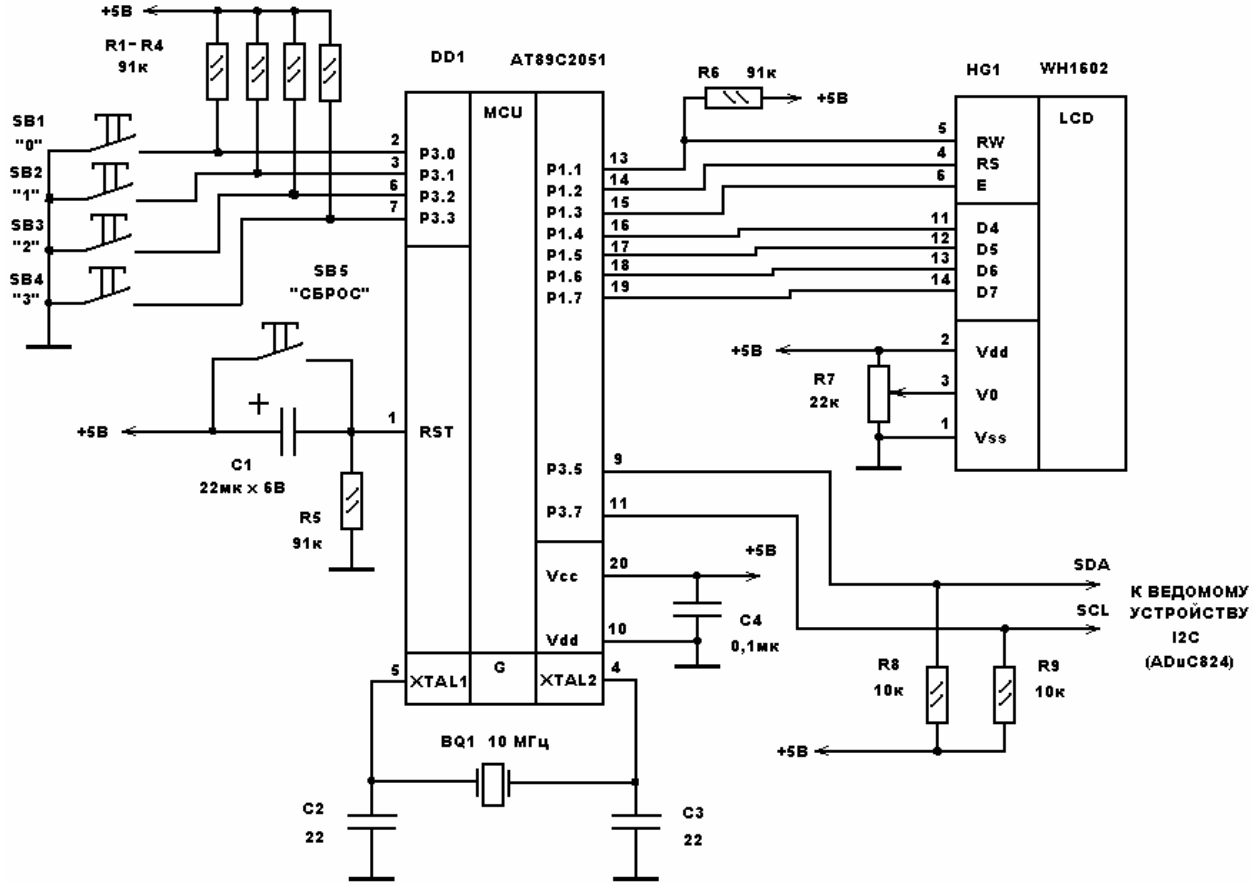


Рис. 3.16. Макет для исследования режима «аппаратный ведомый»

Листинг 3.19. Реализация интерфейса I²C для микроконтроллера AT89C2051

```
;-------------------------------------------------------------------------------
;Демонстрационная программа организации обмена  данными между микроконтроллером
;AT89C2051 и микроконвертером ADuC824 по шине I2C.
;
;AT89C2051 является мастер-устройством I2C.
;Обмен по шине I2C производится только по инициативе AT89C2051 и тактируется им.
;Протокол обмена по шине I2C в AT89C2051 реализован чисто программно.
;В качестве линий SDA и SCL можно использовать любые линии ввода-вывода общего
;назначения. Производится программное слежение за состоянием линии SCL.
;
;При нажатии на кнопку 0 производится выбор байта данных для передачи в ведомое
;устройство I2C. Выбранный байт данных в ASCII коде  выводится на ЖКИ по адресу
;0 ОЗУ ЖКИ.
;
;При нажатии на кнопку 1 производится передача выбранного байта данных в
;ведомое устройство I2C. Переданный байт данных выводится на ЖКИ по  адресу 4
;ОЗУ ЖКИ.
;
;При нажатии на кнопку 2 производится чтение (прием) байта данных из ведомого
;устройства I2C. Прочитанный байт данных в ASCII коде отображается на  ЖКИ по
;адресу 8 ОЗУ ЖКИ.
;-------------------------------------------------------------------------------
          $INCLUDE  (C:\PR_2051\I2C_MAST\2051.inc)
;-------------------------------------------------------------------------------
;Описание битов, регистров и констант
;-------------------------------------------------------------------------------
;Порты и линии ввода-вывода
          PORT_KNOP    EQU      P3       ;порт кнопок

          PORT_IND     EQU      P1       ;порт индикации
```

```
          _IN_KNOP0    EQU      P3_0    ;----------------------------
          _IN_KNOP1    EQU      P3_1    ;
          _IN_KNOP2    EQU      P3_2    ;  входы кнопок
          _IN_KNOP3    EQU      P3_3    ;
          _IN_KNOP4    EQU      P3_4    ;
          _IN_KNOP5    EQU      P3_5    ;

          _IN_KNOP7    EQU      P3_7    ;----------------------------

          PORT_IND_0   EQU      P1_0    ;----------------------------
          PORT_IND_1   EQU      P1_1    ;
          PORT_IND_2   EQU      P1_2    ; выходы индикации
          PORT_IND_3   EQU      P1_3    ;
          PORT_IND_4   EQU      P1_4    ;
          PORT_IND_5   EQU      P1_5    ;
          PORT_IND_6   EQU      P1_6    ;
          PORT_IND_7   EQU      P1_7    ;----------------------------

          RW           EQU      PORT_IND_1 ;-------------------------
          RS           EQU      PORT_IND_2 ; линии управления ЖКИ
          E            EQU      PORT_IND_3 ;-------------------------

          _SDA         EQU      P3_5    ;линия данных  шины I2C
          _SCL         EQU      P3_7    ;линия синхронизации шины I2C

;РОНы обслуживания ЖКИ и кнопок
          ADR_IND_R    DATA     030h    ;----------------------------
          COM_IND_R    DATA     030h    ; РОНы обслуживания ЖКИ
          DATA_IND_R   DATA     031h    ;----------------------------

          KNOP0_R      DATA     032h    ;----------------------------
          KNOP1_R      DATA     033h    ;
          KNOP2_R      DATA     034h    ; регистры, содержащие
          KNOP3_R      DATA     035h    ;
          KNOP4_R      DATA     036h    ; флаги нажатия и удержания
          KNOP5_R      DATA     037h    ;
          KNOP6_R      DATA     038h    ; каждой кнопки
          KNOP7_R      DATA     039h    ;----------------------------

          NAKOPL0_R    DATA     03Ah    ;----------------------------
          NAKOPL1_R    DATA     03Bh    ;
          NAKOPL2_R    DATA     03Ch    ; регистры накопления
          NAKOPL3_R    DATA     03Dh    ;
          NAKOPL4_R    DATA     03Eh    ; значения подавления дребезга
          NAKOPL5_R    DATA     03Fh    ;
          NAKOPL6_R    DATA     040h    ; каждой кнопки
          NAKOPL7_R    DATA     041h    ;----------------------------

;РОНы обслуживания интерфейса программный ведущий-I2C
          DATA_I2C_R   DATA     060h    ;РОН данных I2C
          SLUG_I2C_R   DATA     061h    ;РОН служ инф I2C (адр устр и призн зап)

;Флаги
          _ERR_I2C     BIT      07h     ;флаг "ошибка I2C"

;Константа адреса ведомого устройства I2C, сдвинутого влево на 1
          ADRSL_I2C_K  EQU    10001000b ;служ слово обращ к ведомомоу устр I2C,
                                        ;содержит адр ведом устр, включающий:
                                        ;тип устройства I2C (1000), номер устр
                                        ;I2C на шине (100) и признак записи (0)
;Прочие константы
          NACH_ADR     EQU      000h    ;начальный адрес обнуления РОНов
          KON_ADR      EQU      06Fh    ;конечный адрес обнуления РОНов
          POROG_K      EQU      100     ;порог подавления дребезга кнопок
;Начало исполняемого кода-------------------------------------------------------
                ORG 0h
           AJMP       Lab_START            ;идти на начало осн программы
```

```
;Начало осн программы-----------------------------------------------------
                ORG 050h
Lab_START: MOV        SP,#070h              ;определить указатель стека

           LCALL      Pod_INIT_RSN          ;иниц РСН
           LCALL      Pod_INIT_RON          ;иниц РОН

           LCALL      Pod_INIT_LCD          ;иниц ЖКИ

           LCALL      Pod_CLEAR_LCD         ;стирание ЖКИ

           MOV        R5,#021h          ;задание нач значен индицируемого байта

           MOV        DATA_IND_R,R5         ;-----------------------------------
           MOV        ADR_IND_R,#0          ; индикация  байта (символа)
           ACALL      Pod_PER_DAT_LCD       ;-----------------------------------

;Начало основного цикла---------------------------------------------------
La_OSN:    NOP                              ;метка возврата в осн цикле

;Блок, предшествующий вызову подпрограммы опроса кнопки 0.
           MOV        R0,#KNOP0_R           ;
           MOV        R1,#NAKOPL0_R         ;
           LCALL      Pod_OPR_KNOP0         ;
;Блок, следующий за вызовом подпрограммы опроса кнопки (уст факта ее наж)
           MOV        ACC,@R0               ;
           JNB        ACC_1,La_100          ;

           CLR        ACC_1                 ;кн была нажата, сброс флага нажат кн
           MOV        @R0,ACC               ;

           INC        R5                    ;-----------------------------------
           CJNE       R5,#06Fh,La_00        ; перебор возможн знач индицир байта
           MOV        R5,#021h              ;-----------------------------------

La_00:     MOV        DATA_IND_R,R5         ;-----------------------------------
           MOV        ADR_IND_R,#0          ; индикация  байта (символа)
           ACALL      Pod_PER_DAT_LCD       ;-----------------------------------

;Блок, предшествующий вызову подпрограммы опроса кнопки 1.
La_100:    MOV        R0,#KNOP1_R           ;
           MOV        R1,#NAKOPL1_R         ;
           LCALL      Pod_OPR_KNOP1         ;
;Блок, следующий за вызовом подпрограммы опроса кнопки (уст факта ее наж)
           MOV        ACC,@R0               ;
           JNB        ACC_1,La_200          ;

           CLR        ACC_1                 ;кн была нажата, сброс флага нажат кн
           MOV        @R0,ACC               ;

           MOV        SLUG_I2C_R,#ADRSL_I2C_K ;подгот служебной константы I2C
           MOV        DATA_I2C_R,R5    ;подгот данных для перед в ведом устр I2C
           LCALL      Pod_WRITE_I2C         ;передача в ведомое устр I2C

           MOV        DATA_IND_R,DATA_I2C_R;---------------------------------
           MOV        ADR_IND_R,#4       ; индикация записанного байта (символа)
           ACALL      Pod_PER_DAT_LCD       ;-----------------------------------

;Блок, предшествующий вызову подпрограммы опроса кнопки 2.
La_200:    MOV        R0,#KNOP2_R           ;
           MOV        R1,#NAKOPL2_R         ;
           LCALL      Pod_OPR_KNOP2         ;
;Блок, следующий за вызовом подпрограммы опроса кнопки (уст факта ее наж)
           MOV        ACC,@R0               ;
           JNB        ACC_1,La_300          ;
           CLR        ACC_1                 ;кн была нажата, сброс флага нажат кн
           MOV        @R0,ACC               ;
```

```
            MOV       SLUG_I2C_R,#ADRSL_I2C_K ;подготовка служебной константы I2C
            MOV        DATA_I2C_R,#0FFh     ;контрольная порча РОНа данных I2C
            LCALL      Pod_READ_I2C         ;чтение из ведомого устр I2C

            MOV        DATA_IND_R,DATA_I2C_R;-------------------------------------
            MOV        ADR_IND_R,#8       ;индикация прочитанного байта (символа)
            ACALL      Pod_PER_DAT_LCD      ;-------------------------------------

La_300:     LJMP       La_OSN               ;закрыть основной цикл


;Подпрограммы-----------------------------------------------------------------------

;Подпрограммы организации обмена байтами данных по шине I2C для
;мастер-устройства.
;-------------------------------------------------------------------------------
;Подпрограмма базового интервала (полупериода сигнала SCL) I2C
;-------------------------------------------------------------------------------
Pod_DEL_I2C:
          NOP                            ;тело задержки
          RET
;-------------------------------------------------------------------------------
;Подпрограмма отрицательного перепада на линии SCL I2C
;-------------------------------------------------------------------------------
Pod_LOW_SCL:
          CLR          _SCL              ;спад SCL
          LCALL        Pod_DEL_I2C       ;пауза
          RET

;-------------------------------------------------------------------------------
;Подпрограмма отрицательного перепада на линии SDA I2C
;-------------------------------------------------------------------------------
Pod_LOW_SDA:
          CLR          _SDA              ;спад SDA
          LCALL        Pod_DEL_I2C       ;пауза
          RET

;-------------------------------------------------------------------------------
;Подпрограмма положительного перепада на линии SCL I2C
;-------------------------------------------------------------------------------
Pod_HIGH_SCL:
          SETB         _SCL              ;фронт SCL
          LCALL        Pod_DEL_I2C       ;пауза
          RET

;-------------------------------------------------------------------------------
;Подпрограмма положительного перепада на линии SDA I2C
;-------------------------------------------------------------------------------
Pod_HIGH_SDA:
          SETB         _SDA              ;фронт SDA
          LCALL        Pod_DEL_I2C       ;пауза
          RET

;-------------------------------------------------------------------------------
;Подпрограмма положительного импульса на линии SCL I2C
;-------------------------------------------------------------------------------
Pod_CLOCK_PULSE:
          LCALL        Pod_HIGH_SCL      ;фронт SCL
          JNB          _SCL,$            ;дожидаемся перехода SCL в выс уров
          LCALL        Pod_LOW_SCL       ;спад SCL
          RET

;-------------------------------------------------------------------------------
;Подпрограмма генерации условия START I2C
;-------------------------------------------------------------------------------
Pod_START:
          LCALL        Pod_LOW_SCL       ;
          LCALL        Pod_HIGH_SDA      ;
```

```
          LCALL       Pod_HIGH_SCL        ;
          LCALL       Pod_LOW_SDA         ;START
          LCALL       Pod_LOW_SCL         ;
          RET

;-------------------------------------------------------------------------------
;Подпрограмма генерации условия STOP I2C
;-------------------------------------------------------------------------------
Pod_STOP: LCALL       Pod_LOW_SCL         ;
          LCALL       Pod_LOW_SDA         ;
          LCALL       Pod_HIGH_SCL        ;
          LCALL       Pod_HIGH_SDA        ;STOP
          RET

;-------------------------------------------------------------------------------
;Подпрограмма опроса подтверждения на линии SDA I2C.
;В случае отсутствия подтверждения (выс уровень на линии SDA) устанавливается
;флаг ошибки шины I2C - _ERR_I2C.
;-------------------------------------------------------------------------------
Pod_ZACK: LCALL       Pod_HIGH_SDA        ;SDA - вход
          LCALL       Pod_HIGH_SCL        ;фронт SCL
          JNB         _SCL,$              ;дожидаемся перехода SCL в выс уров
          JNB         _SDA,Zack_0         ;есть ли подтверждение ?
          SETB        _ERR_I2C          ;подтверждения нет - уст флаг ошибк I2C
Zack_0:   LCALL       Pod_LOW_SCL         ;спад SCL
          RET

;-------------------------------------------------------------------------------
;Подпрограмма выдачи неподтверждения на линию SDA I2C.
;-------------------------------------------------------------------------------
Pod_NACK: LCALL       Pod_HIGH_SDA        ;фронт SDA
          LCALL       Pod_CLOCK_PULSE     ;импульс SCL
          RET

;-------------------------------------------------------------------------------
;Подпрограмма передачи байта по шине I2C. Передаваемый байт следует
;предварительно поместить в аккумулятор.  Используется счетчик циклов R3.
;Аккумулятор портится.
;-------------------------------------------------------------------------------
Pod_OUT_BYTE:
          MOV         R3,#8               ;нач уст ст циклов
Out_0:    RLC         A                   ;сдвиг А влево через С, бит 7 идет в С
          JC          Out_10              ;в зависимости от значения С:
          LCALL       Pod_LOW_SDA         ;передача 0
          SJMP        Out_20              ;
Out_10:   LCALL       Pod_HIGH_SDA        ;передача 1
Out_20:   LCALL       Pod_CLOCK_PULSE     ;импульс SCL
          DJNZ        R3,Out_0            ;
          RET

;-------------------------------------------------------------------------------
;Подпрограмма приема байта по шине I2C. Принятый байт оказывается в аккумуляторе.
;Используется счетчик циклов R3.
;-------------------------------------------------------------------------------
Pod_IN_BYTE:
          MOV         R3,#8               ;нач уст ст циклов
          LCALL       Pod_HIGH_SDA        ;SDA - вход
In_0:     LCALL       Pod_HIGH_SCL        ;фронт SCL
          JNB         _SCL,$              ;дожидаемся перехода SCL в выс уров
          MOV         C,_SDA              ; прием бита
          LCALL       Pod_LOW_SCL         ;спад SCL
          RLC         A                   ;сдвиг А влево через С, С идет в бит 0
          DJNZ        R3,In_0             ;
          RET

;-------------------------------------------------------------------------------
;Подпрограмма передачи байта в ведомое устройство  по шине I2C.
;Предварительно следует поместить в РОН DATA_I2C_R - данные передачи,
```

```
;в РОН SLUG_I2C_R - адрес устройства и признак операции (запись).
;-------------------------------------------------------------------------
Pod_WRITE_I2C:
          LCALL       Pod_START           ;START
          MOV         A,SLUG_I2C_R        ;  передача адр устройства
          LCALL       Pod_OUT_BYTE        ;  и признака записи
          LCALL       Pod_ZACK            ;запрос подтверждения
          MOV         A,DATA_I2C_R        ;  передача данных записи
          LCALL       Pod_OUT_BYTE        ;
          LCALL       Pod_ZACK            ;запрос подтверждения
          LCALL       Pod_STOP            ;STOP
          RET

;-------------------------------------------------------------------------
;Подпрограмма чтения байта из ведомого устройства  по шине I2C.
;Предварительно следует поместить в РОН SLUG_I2C_R - адрес устройства и признак
;операции (запись). Прочитанный байт данных возвращается в РОНе DATA_I2C_R.
;-------------------------------------------------------------------------
Pod_READ_I2C:
          LCALL       Pod_START           ;START
          MOV         A,SLUG_I2C_R        ;
          SETB        ACC_0               ;уст признак чтения данных
          LCALL       Pod_OUT_BYTE       ;передача адреса устр и признака чтения
          LCALL       Pod_ZACK            ;запрос подтверждения
          LCALL       Pod_IN_BYTE         ;чтение байта данных из устройства
          LCALL       Pod_NACK            ;не даем подтверждение
          LCALL       Pod_STOP            ;STOP
          MOV         DATA_I2C_R,A        ;
          RET

;-------------------------------------------------------------------------
;Подпрограмма инициализации РСН.
;-------------------------------------------------------------------------
Pod_INIT_RSN:
          MOV        PORT_KNOP,#11111111b ;сделать вх все линии порта кн
          MOV        PORT_IND,#00000000b  ;сделать вых все линии порта индик
          RET

;-------------------------------------------------------------------------
;Подпрограмма инициализации РОНов. Обнуляются все РОНы с адресами
;от NACH_ADR до KON_ADR при помощи косвенной адресации.
;-------------------------------------------------------------------------
Pod_INIT_RON:
          MOV        R0,#NACH_ADR         ;установка начального адреса
Lk_0:     MOV        @R0,#0               ;обнуление очередного РОНа
          INC        R0                   ;переход к следующему адресу
          CJNE       R0,#KON_ADR,Lk_0     ;не достигли ли последнего адреса ?
          MOV        @R0,#0               ;обнуление последнего РОНа
          RET                             ;да, выход

;Подключение модулей опроса кнопок и вывода данных на ЖКИ (с опросом ЖКИ)
          $INCLUDE   (C:\PR_2051\I2C_MAST\knop.asm)
          $INCLUDE   (C:\PR_2051\I2C_MAST\lcd_opr.asm)

;Конец исполняемого кода
          END
```

Трансляция этого файла в объектный файл с расширением .hex для AT89C2051 осуществляется без каких-либо отличий от трансляции файлов с исходными текстами для ADuC824. К сожалению, AT89C2051 не поддерживает режима последовательного внутрисхемного программирования через UART, поэтому читателям, желающим повторить предложенный макет, придется вос-

пользоваться для программирования AT89C2051 каким-либо программатором, реализующим параллельное программирование этого микроконтроллера.

При нажатии на кнопку «0» производится выбор байта данных для передачи в «ведомое» устройство I^2C. ASCII-код последнего выбранного байта индицируется по адресу 0 ОЗУ ЖКИ. При нажатии на кнопку «1» производится передача выбранного ранее байта в «ведомое» устройство I^2C. ASCII-код переданного байта индицируется по адресу 4 ОЗУ ЖКИ. При нажатии на кнопку «2» производится прием байта данных из «ведомого» устройства I^2C. ASCII-код последнего принятого байта индицируется по адресу 8 ОЗУ ЖКИ.

Как можно видеть из текста файла i2c_mast.asm, для реализации режима «мастер» I^2C центрального процессора использованы подпрограммы описанного выше программного интерфейса i2c_eep.asm с поправками на то, что в этом случае между «ведущим» и «ведомым» устройствами происходит обмен данными без выдачи на шину I^2C адресов ячеек памяти, ввиду отсутствия последней. Соответственно, подпрограммы Pod_WRITE_I2C и Pod_READ_I2C во встроенном программном обеспечении центрального процессора модифицированы таким образом, чтобы производить передачу только байта «адрес устройства и признак операции» и передачу или прием одного байта данных. Временные диаграммы, иллюстрирующие процесс передачи одного байта данных из центрального процессора в ADuC824 и прием одного байта данных центральным процессором из ADuC824, приведены соответственно на рис. 3.17, а и рис. 3.17, в. Временные диаграммы, показанные на рис. 3.17, б и рис. 3.17, г, иллюстрируют соответственно порядок передачи и приема последовательности из N байтов данных. Программное обеспечение для этих диаграмм легко разработать, пользуясь подпрограммами из файлов i2c_slav.asm и i2c_mast.asm. Заинтересованным лицам предлагается сделать это самостоятельно.

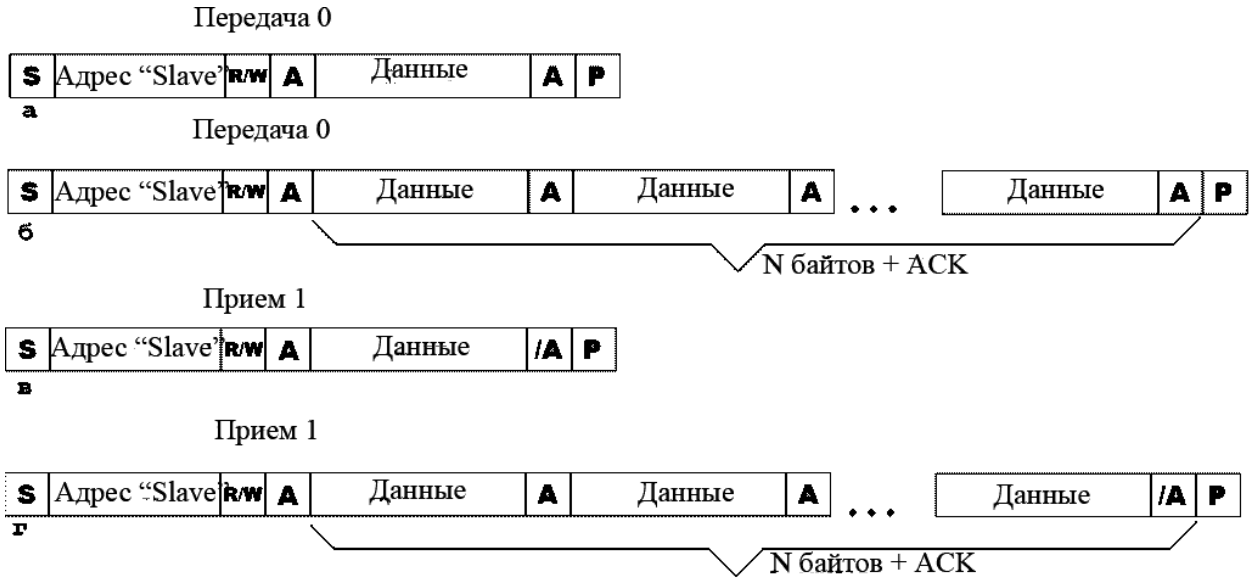


Рис. 3.17. Передача данных по интерфейсу I^2C

3.8. Модуль TIC как часы реального времени

Аппаратный модуль счетчика временных интервалов производителем ADuC824 рекомендуется использовать в проектах для построения часов реального времени. Программа, исходный текст которой содержится в файле tic24.asm (листинг 3.20), реализует часы реального времени с 24-часовым отсчетом (0…23) и возможностью ручной установки показаний.

Листинг 3.20. Реализация часов реального времени

```
;-------------------------------------------------------------------------------
;Демонстрационная программа использования модуля счетчика временных
;интервалов (TIC) ADuC824.
;
;Программа реализует часы реального времени с возможностью ручной установки
;показаний. Отсчет времени  24-часовой.
;При нажатии на кнопку 0 производится установка часов.
;При нажатии на кнопку 1 производится установка минут.
;При нажатии на кнопку 2 производится обнуление счетчиков секунд и сотых долей
;секунд (установка точного времени).
;Текущие значения счетчиков часов, минут и секунд индицируются на ЖКИ.
;Прерывания от TIC не используются.
;
;-------------------------------------------------------------------------------
          $INCLUDE  (C:\ADuC\mod824)
          $INCLUDE  (C:\PR_ADUC\TIC24\824.inc)
;-------------------------------------------------------------------------------
;Описание битов, регистров и констант
;-------------------------------------------------------------------------------
;Порты и линии ввода-вывода
          PORT_KNOP    EQU      P0      ;порт кнопок

          PORT_IND     EQU      P2      ;порт индикации

          _IN_KNOP0    EQU      P0_0    ;-----------------------------
          _IN_KNOP1    EQU      P0_1    ;
          _IN_KNOP2    EQU      P0_2    ;  входы кнопок
          _IN_KNOP3    EQU      P0_3    ;
```

```
          _IN_KNOP4    EQU      P0_4    ;
          _IN_KNOP5    EQU      P0_5    ;
          _IN_KNOP6    EQU      P0_6    ;
          _IN_KNOP7    EQU      P0_7    ;-----------------------------

          PORT_IND_0   EQU      P2_0    ;-----------------------------
          PORT_IND_1   EQU      P2_1    ;
          PORT_IND_2   EQU      P2_2    ; выходы индикации
          PORT_IND_3   EQU      P2_3    ;
          PORT_IND_4   EQU      P2_4    ;
          PORT_IND_5   EQU      P2_5    ;
          PORT_IND_6   EQU      P2_6    ;
          PORT_IND_7   EQU      P2_7    ;-----------------------------

          RW           EQU      PORT_IND_1 ;-------------------------
          RS           EQU      PORT_IND_2 ; линии управления ЖКИ
          E            EQU      PORT_IND_3 ;-------------------------

;РОНы обслуживания ЖКИ и кнопок
          ADR_IND_R    DATA     030h    ;-----------------------------
          COM_IND_R    DATA     030h    ; РОНы обслуживания ЖКИ
          DATA_IND_R   DATA     031h    ;-----------------------------

          KNOP0_R      DATA     032h    ;-----------------------------
          KNOP1_R      DATA     033h    ;
          KNOP2_R      DATA     034h    ; регистры, содержащие
          KNOP3_R      DATA     035h    ;
          KNOP4_R      DATA     036h    ; флаги нажатия и удержания
          KNOP5_R      DATA     037h    ;
          KNOP6_R      DATA     038h    ; каждой кнопки
          KNOP7_R      DATA     039h    ;-----------------------------

          NAKOPL0_R    DATA     03Ah    ;-----------------------------
          NAKOPL1_R    DATA     03Bh    ;
          NAKOPL2_R    DATA     03Ch    ; регистры накопления
          NAKOPL3_R    DATA     03Dh    ;
          NAKOPL4_R    DATA     03Eh    ; значения подавления дребезга
          NAKOPL5_R    DATA     03Fh    ;
          NAKOPL6_R    DATA     040h    ; каждой кнопки
          NAKOPL7_R    DATA     041h    ;-----------------------------

;РОНы обслуживания подпрограмм преобразования формы представления чисел
          IND_MILL_R   DATA     04Eh    ;РОН миллиардов дес числа
          IND_SOTMIL_R DATA     04Fh    ;РОН сотен миллионов дес числа
          IND_DESMIL_R DATA     050h    ;РОН десятков миллионов дес числа
          IND_MIL_R    DATA     051h    ;РОН миллионов дес числа
          IND_SOTTIS_R DATA     052h    ;РОН сотен тысяч дес числа
          IND_DESTIS_R DATA     053h    ;РОН десятков тысяч дес числа
          IND_TIS_R    DATA     054h    ;РОН тысяч дес числа
          IND_SOT_R    DATA     055h    ;РОН сотен дес числа
          IND_DES_R    DATA     056h    ;РОН десятков дес числа
          IND_ED_R     DATA     057h    ;РОН единиц дес числа

          BYTE_0_R     DATA     05Bh    ;байт 0 упакованного двоичн-дес числа
          BYTE_1_R     DATA     05Ch    ;байт 1 упакованного двоичн-дес числа
          BYTE_2_R     DATA     05Dh    ;байт 2 упакованного двоичн-дес числа
          BYTE_3_R     DATA     05Eh    ;байт 3 упакованного двоичн-дес числа
          BYTE_4_R     DATA     05Fh    ;байт 4 упакованного двоичн-дес числа

          RAB_R        DATA     060h    ;рабочий РОН

;Константы
          NACH_ADR     EQU      000h    ;начальный адрес обнуления РОНов
          KON_ADR      EQU      07Fh    ;конечный адрес обнуления РОНов
          POROG_K      EQU      5       ;порог подавления дребезга кнопок
;Начало исполняемого кода-----------------------------------------------------
                ORG  0h
           AJMP       Lab_START            ;идти на начало осн программы
```

```
;Начало осн программы-----------------------------------------------------
                ORG 100h
Lab_START: MOV        SP,#080h             ;определить указатель стека
           MOV        PLLCON,#00000000b    ;уст макс частоту ядра (12,58 МГц)
           NOP                             ;
           LCALL      Pod_INIT_RSN         ;иниц РСН
           LCALL      Pod_INIT_RON         ;иниц РОН

           LCALL      Pod_INIT_LCD         ;иниц ЖКИ

           LCALL      Pod_CLEAR_LCD        ;стирание ЖКИ

            MOV        DATA_IND_R,#3Ah      ;-----------------------------------
           MOV        ADR_IND_R,#2         ; индикация разделительных двоеточий
           LCALL      Pod_PER_DAT_LCD      ;
            MOV        ADR_IND_R,#5         ; по адресам 2 и 5 ОЗУ ЖКИ
           LCALL      Pod_PER_DAT_LCD      ;-----------------------------------


           MOV        TIMECON,#00110000b   ;разрешить модификацию регистров TIC
            MOV        MIN,#55              ;нач уст регистра минут
           MOV        HOUR,#23             ;нач установка регистра часов
           MOV        TIMECON,#00110011b   ;запретить модификацию регистров TIC

;Начало основного цикла---------------------------------------------------
La_OSN:    NOP                             ;метка возврата в осн цикле

;Блок, предшествующий вызову подпрограммы опроса кнопки 0.
           MOV        R0,#KNOP0_R          ;
           MOV        R1,#NAKOPL0_R        ;
           LCALL      Pod_OPR_KNOP0        ;
;Блок, следующий за вызовом подпрограммы опроса кнопки (уст факта ее наж)
           MOV        ACC,@R0              ;
           JNB        ACC_1,La_30          ;

           CLR        ACC_1                ;кн была нажата, сброс флага нажат кн
           MOV        @R0,ACC              ;

           MOV        R1,HTHSEC            ;-----------------------------------
           MOV        R2,SEC               ; сохранение регистров счета
            MOV        R3,MIN               ;  времени TIC
           MOV        A,HOUR               ;-----------------------------------
            INC        A                    ;
           CJNE       A,#24,La_10          ;
            MOV        A,#0                 ;
La_10:     MOV        TIMECON,#00110000b   ;разрешить модификацию регистров TIC
           MOV        HTHSEC,R1            ;-----------------------------------
            MOV        SEC,R2               ; восст регистров счета времени TIC
            MOV        MIN,R3               ;-----------------------------------
            MOV        HOUR,A               ;установка часов
           MOV        TIMECON,#00110011b   ;запретить модификацию регистров TIC


           LJMP       La_OSN               ;закрыть основной цикл

;Блок, предшествующий вызову подпрограммы опроса кнопки 1.
La_30:     MOV        R0,#KNOP1_R          ;
           MOV        R1,#NAKOPL1_R        ;
           LCALL      Pod_OPR_KNOP1        ;
;Блок, следующий за вызовом подпрограммы опроса кнопки (уст факта ее наж)
           MOV        ACC,@R0              ;
           JNB        ACC_1,La_60         ;

           CLR        ACC_1                ;кн была нажата, сброс флага нажат кн
           MOV        @R0,ACC              ;

            MOV        R1,HTHSEC            ;-----------------------------------
```

```
            MOV         R2,SEC                ; сохранение регистров счета
             MOV         R3,HOUR               ;  времени TIC
            MOV         A,MIN                 ;---------------------------------
             INC         A                     ;
            CJNE        A,#60,La_50           ;
             MOV         A,#0                  ;
La_50:      MOV         TIMECON,#00110000b    ;разрешить модификацию регистров TIC
            MOV         MIN,A                 ;установка минут
            MOV         HTHSEC,R1             ;---------------------------------
             MOV         SEC,R2                ; восст регистров счета времени TIC
             MOV         HOUR,R3               ;---------------------------------
             MOV         TIMECON,#00110011b    ;запретить модификацию регистров TIC

            LJMP        La_OSN                ;закрыть основной цикл

;Блок, предшествующий вызову подпрограммы опроса кнопки 2.
La_60:      MOV         R0,#KNOP2_R           ;
            MOV         R1,#NAKOPL2_R         ;
            LCALL       Pod_OPR_KNOP2         ;
;Блок, следующий за вызовом подпрограммы опроса кнопки (уст факта ее наж)
            MOV         ACC,@R0               ;
            JNB         ACC_1,La_100          ;

            CLR         ACC_1                 ;кн была нажата, сброс флага нажат кн
            MOV         @R0,ACC               ;

            MOV         R1,MIN                ;  сохранение регистров счета
             MOV         R2,HOUR               ;  времени TIC
            MOV         TIMECON,#00110000b    ;разрешить модификацию регистров TIC
            MOV         MIN,R1                ;---------------------------------
             MOV         HOUR,R2               ; восст регистров счета времени TIC
             MOV         TIMECON,#00110011b    ;запретить модификацию регистров TIC

            LJMP        La_OSN                ;закрыть основной цикл

;Блок преобразования и индикации текущего времени
La_100:     MOV         RAB_R,HOUR            ;
            MOV         R0,#BYTE_0_R          ;
            MOV         R1,#RAB_R             ;
            MOV         RAB_R+1,#0            ;---------------------------------
            MOV         RAB_R+2,#0            ; обнуление ст незначащих РОНов
             MOV         RAB_R+3,#0            ;---------------------------------
            LCALL       B32BCD                ;преобр часов из двоичн в двоичн-дес

            MOV         R0,#BYTE_0_R          ;
            MOV         R1,#IND_MILL_R        ;
            LCALL       BCD10BCD             ;преобр часов из уп дв-дес в неуп дес

            MOV         DATA_IND_R,IND_DES_R ;
            MOV         ADR_IND_R,#0          ;
            LCALL       Pod_PER_DAT_LCD       ; индикация дес часов
            MOV         DATA_IND_R,IND_ED_R   ;
            MOV         ADR_IND_R,#1          ;
            LCALL       Pod_PER_DAT_LCD       ; индикация единиц часов


            MOV         RAB_R,MIN             ;
            MOV         R0,#BYTE_0_R          ;
            MOV         R1,#RAB_R             ;
            MOV         RAB_R+1,#0            ;---------------------------------
            MOV         RAB_R+2,#0            ; обнуление ст незначащих РОНов
             MOV         RAB_R+3,#0            ;---------------------------------
            LCALL       B32BCD                ;преобр минут из двоичн в двоичн-дес

            MOV         R0,#BYTE_0_R          ;
            MOV         R1,#IND_MILL_R        ;
            LCALL       BCD10BCD             ;преобр минут из уп дв-дес в неуп дес
```

```
            MOV         DATA_IND_R,IND_DES_R ;
            MOV         ADR_IND_R,#3         ;
            LCALL       Pod_PER_DAT_LCD      ; индикация дес минут
            MOV         DATA_IND_R,IND_ED_R  ;
            MOV         ADR_IND_R,#4         ;
            LCALL       Pod_PER_DAT_LCD      ; индикация единиц минут


            MOV         RAB_R,SEC            ;
            MOV         R0,#BYTE_0_R         ;
            MOV         R1,#RAB_R            ;
            MOV         RAB_R+1,#0           ;-------------------------------
            MOV         RAB_R+2,#0           ; обнуление ст незначащих РОНов
             MOV         RAB_R+3,#0           ;-------------------------------
            LCALL       B32BCD               ;преобр секунд из двоичн в двоичн-дес

            MOV         R0,#BYTE_0_R         ;
            MOV         R1,#IND_MILL_R       ;
            LCALL       BCD10BCD           ;преобр мсекунд из уп дв-дес в неуп дес

            MOV         DATA_IND_R,IND_DES_R ;
            MOV         ADR_IND_R,#6         ;
            LCALL       Pod_PER_DAT_LCD      ; индикация десятков секунд
            MOV         DATA_IND_R,IND_ED_R  ;
            MOV         ADR_IND_R,#7         ;
            LCALL       Pod_PER_DAT_LCD      ; индикация единиц секунд

            LJMP        La_OSN               ;закрыть основной цикл

;Подпрограммы-----------------------------------------------------------------

;-------------------------------------------------------------------------------
;Подпрограмма инициализации РСН.
;-------------------------------------------------------------------------------
Pod_INIT_RSN:
            MOV         PORT_KNOP,#11111111b ;сделать вх все линии порта кн
            MOV         PORT_IND,#00000000b  ;сделать вых все линии порта индик

;Блок настройки модуля TIC
            MOV         TIMECON,#00110011b   ;базовый интервал - часы, многократно
                                              ;генерировать интервал, разреш счет
                                                ;разрешить счетчик врем интервала
            MOV         IEIP2,#00000000b     ;запретить прерывания от TIC
             RET

;-------------------------------------------------------------------------------
;Подпрограмма инициализации РОНов. Обнуляются все РОНы с адресами
;от NACH_ADR до KON_ADR при помощи косвенной адресации.
;-------------------------------------------------------------------------------
Pod_INIT_RON:
            MOV         R0,#NACH_ADR         ;установка начального адреса
Lk_0:       MOV         @R0,#0               ;обнуление очередного РОНа
            INC         R0                   ;переход к следующему адресу
            CJNE        R0,#KON_ADR,Lk_0     ;не достигли ли последнего адреса ?
            MOV         @R0,#0               ;обнуление последнего РОНа
            RET                              ;да, выход

;Подключение модулей опроса кнопок, вывода данных на ЖКИ (с опросом ЖКИ)
;и преобразования представления чисел
            $INCLUDE   (C:\PR_ADUC\TIC24\knop.asm)
            $INCLUDE   (C:\PR_ADUC\TIC24\lcd_opr.asm)
             $INCLUDE   (C:\PR_ADUC\TIC24\preobr.asm)

;Конец исполняемого кода
            END
```

Для работы этой программы, как и в случае экспериментов с Flash/EE-памятью данных, требуется макет, содержащий только подключенные к ADuC824 кнопки управления и ЖКИ. Программа выводит на ЖКИ значение текущего времени в десятичном представлении и формате {часы : минуты : секунды}. Ручная установка значения часов производится путем нажатия на кнопку «0», значения минут – путем нажатия на кнопку «1». При нажатии на кнопку «2» производится обнуление регистров счета секунд и сотых долей секунд, что позволяет синхронизировать часы по внешнему эталонному источнику времени. После сброса программа начинает счет времени со значения {23:55:00}. Поскольку заданные аппаратно диапазоны величин специальных регистров отсчета времени SEC, MIN, HOUR соответствуют требуемым диапазонам для 24-часового счета, то прерывания от TIC в этой программе использовать нет необходимости. Программная организация процесса отсчета времени сводится к инициализации специального регистра управления модулем TIC TIMECON, производимой в подпрограмме Pod_INIT_RSN. Константа инициализации задает базовый временной интервал (часы) и разрешает прохождение счетных импульсов на вход предварительного делителя TIC. Однако, помимо счета и индикации программа должна обеспечивать пользователю возможность ручной установки содержимого счетчиков времени. Поскольку запись в любой из счетчиков HTHSEC, SEC, MIN, HOUR возможна только при сброшенном бите TCEN регистра TIMECON, а его сброс автоматически вызывает очистку всех счетчиков (табл. 1.19), то для проведения корректной установки счетчиков часов и минут в программе производится предварительное сохранение всех счетчиков с последующим их восстановлением.

24-часовой отсчет времени является самым простым случаем использования TIC, однако зачастую требуется иметь счетчик времени с другим значением главного цикла счета. Программа, исходный текст которой содержится в файле tic12.asm (листинг 3.21), в качестве примера реализует часы реального времени с 12-часовым отсчетом (1…12) и возможностью ручной установки показаний.

Листинг 3.21. Часы реального времени с 12-часовым отсчетом

```
;-------------------------------------------------------------------------
;Демонстрационная программа использования модуля счетчика временных
;интервалов (TIC) ADuC824.
;
;Программа реализует часы реального времени с возможностью ручной установки
;показаний. Отсчет времени  12-часовой.
;При нажатии на кнопку 0 производится установка часов.
;При нажатии на кнопку 1 производится установка минут.
;При нажатии на кнопку 2 производится обнуление счетчиков секунд и сотых долей
;секунд (установка точного времени).
;Текущие значения счетчиков часов, минут и секунд индицируются на ЖКИ.
;Используется прерывание от TIC.
;-------------------------------------------------------------------------
          $INCLUDE  (C:\ADuC\mod824)
          $INCLUDE  (C:\PR_ADUC\TIC12\824.inc)
;-------------------------------------------------------------------------
;Описание битов, регистров и констант
;-------------------------------------------------------------------------
;Порты и линии ввода-вывода
```

```
          PORT_KNOP    EQU      P0      ;порт кнопок

          PORT_IND     EQU      P2      ;порт индикации

          _IN_KNOP0    EQU      P0_0    ;-----------------------------
          _IN_KNOP1    EQU      P0_1    ;
          _IN_KNOP2    EQU      P0_2    ;  входы кнопок
          _IN_KNOP3    EQU      P0_3    ;
          _IN_KNOP4    EQU      P0_4    ;
          _IN_KNOP5    EQU      P0_5    ;
          _IN_KNOP6    EQU      P0_6    ;
          _IN_KNOP7    EQU      P0_7    ;-----------------------------

          PORT_IND_0   EQU      P2_0    ;-----------------------------
          PORT_IND_1   EQU      P2_1    ;
          PORT_IND_2   EQU      P2_2    ; выходы индикации
          PORT_IND_3   EQU      P2_3    ;
          PORT_IND_4   EQU      P2_4    ;
          PORT_IND_5   EQU      P2_5    ;
          PORT_IND_6   EQU      P2_6    ;
          PORT_IND_7   EQU      P2_7    ;-----------------------------

          RW           EQU      PORT_IND_1 ;--------------------------
          RS           EQU      PORT_IND_2 ; линии управления ЖКИ
          E            EQU      PORT_IND_3 ;--------------------------

;РОНы обслуживания ЖКИ и кнопок
          ADR_IND_R    DATA     030h    ;-----------------------------
          COM_IND_R    DATA     030h    ; РОНы обслуживания ЖКИ
          DATA_IND_R   DATA     031h    ;-----------------------------

          KNOP0_R      DATA     032h    ;-----------------------------
          KNOP1_R      DATA     033h    ;
          KNOP2_R      DATA     034h    ; регистры, содержащие
          KNOP3_R      DATA     035h    ;
          KNOP4_R      DATA     036h    ; флаги нажатия и удержания
          KNOP5_R      DATA     037h    ;
          KNOP6_R      DATA     038h    ; каждой кнопки
          KNOP7_R      DATA     039h    ;-----------------------------

          NAKOPL0_R    DATA     03Ah    ;-----------------------------
          NAKOPL1_R    DATA     03Bh    ;
          NAKOPL2_R    DATA     03Ch    ; регистры накопления
          NAKOPL3_R    DATA     03Dh    ;
          NAKOPL4_R    DATA     03Eh    ; значения подавления дребезга
          NAKOPL5_R    DATA     03Fh    ;
          NAKOPL6_R    DATA     040h    ; каждой кнопки
          NAKOPL7_R    DATA     041h    ;-----------------------------

;РОНы обслуживания подпрограмм преобразования формы представления чисел
          IND_MILL_R   DATA     04Eh    ;РОН миллиардов дес числа
          IND_SOTMIL_R DATA     04Fh    ;РОН сотен миллионов дес числа
          IND_DESMIL_R DATA     050h    ;РОН десятков миллионов дес числа
          IND_MIL_R    DATA     051h    ;РОН миллионов дес числа
          IND_SOTTIS_R DATA     052h    ;РОН сотен тысяч дес числа
          IND_DESTIS_R DATA     053h    ;РОН десятков тысяч дес числа
          IND_TIS_R    DATA     054h    ;РОН тысяч дес числа
          IND_SOT_R    DATA     055h    ;РОН сотен дес числа
          IND_DES_R    DATA     056h    ;РОН десятков дес числа
          IND_ED_R     DATA     057h    ;РОН единиц дес числа

          BYTE_0_R     DATA     05Bh    ;байт 0 упакованного двоичн-дес числа
          BYTE_1_R     DATA     05Ch    ;байт 1 упакованного двоичн-дес числа
          BYTE_2_R     DATA     05Dh    ;байт 2 упакованного двоичн-дес числа
          BYTE_3_R     DATA     05Eh    ;байт 3 упакованного двоичн-дес числа
          BYTE_4_R     DATA     05Fh    ;байт 4 упакованного двоичн-дес числа

          RAB_R        DATA     060h    ;рабочий РОН
```

```
;Константы
          NACH_ADR     EQU      000h    ;начальный адрес обнуления РОНов
          KON_ADR      EQU      07Fh    ;конечный адрес обнуления РОНов
          POROG_K      EQU      5       ;порог подавления дребезга кнопок
;Начало исполняемого кода-------------------------------------------------------
                ORG  0h
           AJMP       Lab_START            ;идти на начало осн программы

                ORG  053h
           AJMP       Lab_TIC              ;идти  на начало блока прерыв от TIC

          ORG  060h
;Блок обработки прерываний от TIC. Прерывание по совпадению содержимого
;регистра HOUR  с содержимым INTVAL.
;Прерывания должны быть разрешены.
Lab_TIC:   CLR        EA                   ;---------------------------------
           PUSH       PSW                  ; глоб запрет прер и сохр контекста
           PUSH       ACC                  ;---------------------------------

           MOV        TIMECON,#00110000b   ;разрешить модификацию регистров TIC
                                           ;сбросить флаг прерывания от TIC
           MOV        HOUR,#1              ;установка регистра часов
           MOV        INTVAL,#12           ; уст содержимого INTVAL,
                                           ; соответственно содержимому
                                               ; регистра часов: (13-HOUR)=12
            MOV        TIMECON,#00110011b   ;запретить модификацию регистров TIC
                                           ;разреш счет и счетчик врем интервала
;Блок возврата из прерываний----------------------------------------------------
Lab_RETI:  POP        ACC                  ;---------------------------------
           POP        PSW                  ; восст контекста и глоб разр прер
           SETB       EA                   ;---------------------------------
           RETI                            ;возврат из блока обраб прерываний

;Начало осн программы-----------------------------------------------------------
                ORG 100h
Lab_START: MOV        SP,#080h             ;определить указатель стека
           MOV        PLLCON,#00000000b    ;уст макс частоту ядра (12,58 МГц)
           NOP                             ;
           LCALL      Pod_INIT_RSN         ;иниц РСН
           LCALL      Pod_INIT_RON         ;иниц РОН

           LCALL      Pod_INIT_LCD         ;иниц ЖКИ

           LCALL      Pod_CLEAR_LCD        ;стирание ЖКИ

            MOV        DATA_IND_R,#3Ah      ;-------------------------------------
           MOV        ADR_IND_R,#2         ; индикация разделительных двоеточий
           LCALL      Pod_PER_DAT_LCD      ;
            MOV        ADR_IND_R,#5         ; по адресам 2 и 5 ОЗУ ЖКИ
           LCALL      Pod_PER_DAT_LCD      ;-------------------------------------


           MOV        TIMECON,#00110000b   ;разрешить модификацию регистров TIC
            MOV        MIN,#55              ;нач уст регистра минут
           MOV        HOUR,#12             ;нач установка регистра часов
           MOV        INTVAL,#1            ; уст содержимого INTVAL,
; соответственно содержимому
                                          ; регистра часов: (13-HOUR)=1
            MOV        TIMECON,#00110011b   ;запретить модификацию регистров TIC

            SETB       EA                   ;разрешить прерывания глобально


;Начало основного цикла---------------------------------------------------------
La_OSN:    NOP                             ;метка возврата в осн цикле

;Блок, предшествующий вызову подпрограммы опроса кнопки 0.
```

```
            MOV         R0,#KNOP0_R          ;
            MOV         R1,#NAKOPL0_R        ;
            LCALL       Pod_OPR_KNOP0        ;
;Блок, следующий за вызовом подпрограммы опроса кнопки (уст факта ее наж)
            MOV         ACC,@R0              ;
            JNB         ACC_1,La_30          ;

            CLR         ACC_1                ;кн была нажата, сброс флага нажат кн
            MOV         @R0,ACC              ;

            CLR         EA                   ;глоб запрещение прерываний
            MOV         R1,HTHSEC            ;---------------------------------
            MOV         R2,SEC               ; сохранение регистров счета
             MOV         R3,MIN               ;  времени TIC
            MOV         A,HOUR               ;---------------------------------
             INC         A                   ;
            CJNE        A,#13,La_10          ;
             MOV         A,#1                ;
La_10:      MOV         TIMECON,#00110000b   ;разрешить модификацию регистров TIC
            MOV         HTHSEC,R1            ;---------------------------------
             MOV         SEC,R2              ; восст регистров счета времени TIC
             MOV         MIN,R3              ;---------------------------------
             MOV         HOUR,A              ;установка часов
             MOV         RAB_R,A             ;---------------------------------
             MOV         A,#13               ; коррекция содержимого  INTVAL
             CLR         C                   ;
            SUBB        A,RAB_R              ; после уст часов (INTVAL=(13-HOUR))
            MOV         INTVAL,A             ;---------------------------------
            MOV         TIMECON,#00110011b   ;запретить модификацию регистров TIC


            SETB        EA                   ;разрешить прерывания глобально

            LJMP        La_OSN               ;закрыть основной цикл

;Блок, предшествующий вызову подпрограммы опроса кнопки 1.
La_30:      MOV         R0,#KNOP1_R          ;
            MOV         R1,#NAKOPL1_R        ;
            LCALL       Pod_OPR_KNOP1        ;
;Блок, следующий за вызовом подпрограммы опроса кнопки (уст факта ее наж)
            MOV         ACC,@R0              ;
            JNB         ACC_1,La_60         ;

            CLR         ACC_1                ;кн была нажата, сброс флага нажат кн
            MOV         @R0,ACC              ;

            CLR         EA                   ;глоб запрещение прерываний
             MOV         R1,HTHSEC           ;---------------------------------
            MOV         R2,SEC               ; сохранение регистров счета
             MOV         R3,HOUR             ;  времени TIC
            MOV         A,MIN                ;---------------------------------
             INC         A                   ;
            CJNE        A,#60,La_50          ;
             MOV         A,#0                ;
La_50:      MOV         TIMECON,#00110000b   ;разрешить модификацию регистров TIC
            MOV         MIN,A                ;установка минут
            MOV         HTHSEC,R1            ;---------------------------------
             MOV         SEC,R2              ; восст регистров счета времени TIC
             MOV         HOUR,R3             ;---------------------------------
             MOV         TIMECON,#00110011b   ;запретить модификацию регистров TIC
            SETB        EA                   ;разрешить прерывания глобально

            LJMP        La_OSN               ;закрыть основной цикл

;Блок, предшествующий вызову подпрограммы опроса кнопки 2.
La_60:      MOV         R0,#KNOP2_R          ;
            MOV         R1,#NAKOPL2_R        ;
            LCALL       Pod_OPR_KNOP2        ;
```

```
;Блок, следующий за вызовом подпрограммы опроса кнопки (уст факта ее наж)
            MOV         ACC,@R0             ;
            JNB         ACC_1,La_100        ;

            CLR         ACC_1               ;кн была нажата, сброс флага нажат кн
            MOV         @R0,ACC             ;

            CLR         EA                  ;глоб запрещение прерываний
            MOV         R1,MIN              ;  сохранение регистров счета
             MOV         R2,HOUR             ;  времени TIC
            MOV         TIMECON,#00110000b  ;разрешить модификацию регистров TIC
            MOV         MIN,R1              ;-------------------------------
             MOV         HOUR,R2             ; восст регистров счета времени TIC
             MOV         TIMECON,#00110011b  ;запретить модификацию регистров TIC
            SETB        EA                  ;разрешить прерывания глобально

            LJMP        La_OSN              ;закрыть основной цикл

;Блок преобразования и индикации текущего времени
La_100:
            MOV         RAB_R,HOUR          ;
            MOV         R0,#BYTE_0_R        ;
            MOV         R1,#RAB_R           ;
            MOV         RAB_R+1,#0          ;-------------------------------
            MOV         RAB_R+2,#0          ; обнуление ст незначащих РОНов
             MOV         RAB_R+3,#0          ;-------------------------------
            LCALL       B32BCD              ;преобр часов из двоичн в двоичн-дес

            MOV         R0,#BYTE_0_R        ;
            MOV         R1,#IND_MILL_R      ;
            LCALL       BCD10BCD            ;преобр часов из уп дв-дес в неуп дес

            MOV         DATA_IND_R,IND_DES_R ;
            MOV         ADR_IND_R,#0        ;
            LCALL       Pod_PER_DAT_LCD     ; индикация дес часов
            MOV         DATA_IND_R,IND_ED_R  ;
            MOV         ADR_IND_R,#1        ;
            LCALL       Pod_PER_DAT_LCD     ; индикация единиц часов


            MOV         RAB_R,MIN           ;
            MOV         R0,#BYTE_0_R        ;
            MOV         R1,#RAB_R           ;
            MOV         RAB_R+1,#0          ;-------------------------------
            MOV         RAB_R+2,#0          ; обнуление ст незначащих РОНов
             MOV         RAB_R+3,#0          ;-------------------------------
            LCALL       B32BCD              ;преобр минут из двоичн в двоичн-дес

            MOV         R0,#BYTE_0_R        ;
            MOV         R1,#IND_MILL_R      ;
            LCALL       BCD10BCD           ;преобр минут из уп дв-дес в неуп дес

            MOV         DATA_IND_R,IND_DES_R ;
            MOV         ADR_IND_R,#3        ;
            LCALL       Pod_PER_DAT_LCD     ; индикация дес минут
            MOV         DATA_IND_R,IND_ED_R  ;
            MOV         ADR_IND_R,#4        ;
            LCALL       Pod_PER_DAT_LCD     ; индикация единиц минут


            MOV         RAB_R,SEC           ;
            MOV         R0,#BYTE_0_R        ;
            MOV         R1,#RAB_R           ;
            MOV         RAB_R+1,#0          ;-------------------------------
            MOV         RAB_R+2,#0          ; обнуление ст незначащих РОНов
             MOV         RAB_R+3,#0          ;-------------------------------
            LCALL       B32BCD              ;преобр секунд из двоичн в двоичн-дес
```

```
            MOV         R0,#BYTE_0_R          ;
            MOV         R1,#IND_MILL_R        ;
            LCALL       BCD10BCD            ;преобр секунд из уп дв-дес в неуп дес

            MOV         DATA_IND_R,IND_DES_R ;
            MOV         ADR_IND_R,#6          ;
            LCALL       Pod_PER_DAT_LCD       ; индикация десятков секунд
            MOV         DATA_IND_R,IND_ED_R   ;
            MOV         ADR_IND_R,#7          ;
            LCALL       Pod_PER_DAT_LCD       ; индикация единиц секунд

            LJMP        La_OSN                ;закрыть основной цикл

;Подпрограммы-----------------------------------------------------------------

;-------------------------------------------------------------------------------
;Подпрограмма инициализации PCH.
;-------------------------------------------------------------------------------
Pod_INIT_RSN:
            MOV         PORT_KNOP,#11111111b ;сделать вх все линии порта кн
            MOV         PORT_IND,#00000000b  ;сделать вых все линии порта индик

;Блок настройки модуля TIC
            MOV         TIMECON,#00110011b   ;базовый интервал - часы, многократно
                                              ;генерировать интервал, разрешить счет
                                                 ;разрешить счетчик врем интервала
            MOV         IEIP2,#01000100b     ;разреш прерыв от TIC и уст им высокий
                                              ;приоритет
            RET

;-------------------------------------------------------------------------------
;Подпрограмма инициализации РОНов. Обнуляются все РОНы с адресами
;от NACH_ADR до KON_ADR при помощи косвенной адресации.
;-------------------------------------------------------------------------------
Pod_INIT_RON:
            MOV         R0,#NACH_ADR          ;установка начального адреса
Lk_0:       MOV         @R0,#0                ;обнуление очередного РОНа
            INC         R0                    ;переход к следующему адресу
            CJNE        R0,#KON_ADR,Lk_0      ;не достигли ли последнего адреса ?
            MOV         @R0,#0                ;обнуление последнего РОНа
            RET                               ;да, выход

;Подключение модулей опроса кнопок, вывода данных на ЖКИ (с опросом ЖКИ)
;и преобразования представления чисел
            $INCLUDE   (C:\PR_ADUC\TIC12\knop.asm)
            $INCLUDE   (C:\PR_ADUC\TIC12\lcd_opr.asm)
             $INCLUDE   (C:\PR_ADUC\TIC12\preobr.asm)

;Конец исполняемого кода
            END
```

В этой программе для организации счета используется прерывание от TIC по совпадению содержимого счетчика часов HOUR с содержимым специального регистра выбора значения интервала INTVAL. При написании и отладке программ, подобных этой, необходимо учитывать следующий важный момент. Число разрешенных состояний N, принимаемых счетчиком часов HOUR в нашем случае (при 12-часовом отсчете времени) равно 12. При HOUR=13 должно генерироваться прерывание, в блоке обработки которого в регистр HOUR программно записывается значение 1 (наименьшее разрешенное состояние), после чего начинается новый аппаратный цикл счета времени. Однако на текущем состоянии 8-разрядного счетчика временного интервала программное изменение

HOUR никак не скажется, поскольку счетчик временного интервала инкрементируется только при переполнении счетчика минут MIN (рис. 1.21). Поэтому, чтобы устранить потенциальную ошибку в счете часов, в регистр выбора значения интервала INTVAL программно записывается число, равное разности значения (N+1) и нового значения HOUR. Таким образом, если содержимое HOUR изменяется программно, а не аппаратно от переполнения счетчика минут MIN, то для нового значения HOUR всегда следует сразу же программно корректировать значение INTVAL. Коррекция производится по формуле: INTVAL = (N+1 – HOUR), а для нашего случая – по формуле: INTVAL = (13 – HOUR). Программное изменение содержимого HOUR в файле tic12.asm имеет место в трех местах: в блоке обработки прерываний от TIC, при установке начального значения счета времени {12:55:00} до начала основного цикла программы и при ручной модификации значения часов в блоке обработки нажатий на кнопку «0». Как можно видеть из программы, во всех этих случаях содержимое регистра INTVAL немедленно подвергается коррекции.

3.9. Использование модуля ЦАП

Использование встроенного модуля ЦАП ADuC824 иллюстрируется демонстрационной программой, исходный текст которой содержится в файле dac.asm (листинг 3.22). Для экспериментов с модулем ЦАП, как и в предыдущем случае, требуется макет, содержащий только подключенные к ADuC824 кнопки управления и ЖКИ. Программа генерирует на одном из двух альтернативных выходов ЦАП ADuC824 (P1.7) последовательность импульсов ступенчато возрастающего напряжения, которую можно наблюдать на осциллографе.

Листинг 3.22. Использование встроенного ЦАП

```
;-------------------------------------------------------------------------------
;Демонстрационная программа использования ЦАП ADuC824.
;
;Данные в ЖКИ передаются по 4-битной шине, состояние ЖКИ опрашивается.
;
;На выходе ЦАП P1.7 формируется ступенчато возрастающее напряжение от нуля до
;некоторого верхнего значения. Верхнее значение определяется десятичным числом,
;индицируемым с адреса 0 ОЗУ ЖКИ.
;При нажатии на кнопку 0 производится перебор возможных значений этого числа
;с некоторым шагом.
;
;Прерывания не используются.
;-------------------------------------------------------------------------------
          $INCLUDE  (C:\ADuC\mod824)
          $INCLUDE  (C:\PR_ADUC\DAC\824.inc)
;-------------------------------------------------------------------------------
;Описание битов, регистров и констант
;-------------------------------------------------------------------------------
;Порты и линии ввода-вывода
          PORT_KNOP    EQU      P0       ;порт кнопок

          PORT_IND     EQU      P2       ;порт индикации

          PORT_IND_0   EQU      P2_0     ;----------------------------
```

```
          PORT_IND_1   EQU      P2_1    ;
          PORT_IND_2   EQU      P2_2    ; выходы индикации
          PORT_IND_3   EQU      P2_3    ;
          PORT_IND_4   EQU      P2_4    ;
          PORT_IND_5   EQU      P2_5    ;
          PORT_IND_6   EQU      P2_6    ;
          PORT_IND_7   EQU      P2_7    ;-----------------------------

          RW           EQU      PORT_IND_1 ;-------------------------
          RS           EQU      PORT_IND_2 ; линии управления ЖКИ
          E            EQU      PORT_IND_3 ;-------------------------

          _IN_KNOP0    EQU      P0_0    ;-----------------------------
          _IN_KNOP1    EQU      P0_1    ;
          _IN_KNOP2    EQU      P0_2    ;  входы кнопок
          _IN_KNOP3    EQU      P0_3    ;
          _IN_KNOP4    EQU      P0_4    ;
          _IN_KNOP5    EQU      P0_5    ;
          _IN_KNOP6    EQU      P0_6    ;
          _IN_KNOP7    EQU      P0_7    ;-----------------------------

;РОНы обслуживания ЖКИ и кнопок
          ADR_IND_R    DATA     030h    ;-----------------------------
          COM_IND_R    DATA     030h    ; РОНы обслуживания ЖКИ
          DATA_IND_R   DATA     031h    ;-----------------------------

          KNOP0_R      DATA     032h    ;-----------------------------
          KNOP1_R      DATA     033h    ;
          KNOP2_R      DATA     034h    ; РОНы, содержащие
          KNOP3_R      DATA     035h    ;
          KNOP4_R      DATA     036h    ; флаги нажатия и удержания
          KNOP5_R      DATA     037h    ;
          KNOP6_R      DATA     038h    ; каждой кнопки
          KNOP7_R      DATA     039h    ;-----------------------------

          NAKOPL0_R    DATA     03Ah    ;-----------------------------
          NAKOPL1_R    DATA     03Bh    ;
          NAKOPL2_R    DATA     03Ch    ; РОНы накопления
          NAKOPL3_R    DATA     03Dh    ;
          NAKOPL4_R    DATA     03Eh    ; значения подавления дребезга
          NAKOPL5_R    DATA     03Fh    ;
          NAKOPL6_R    DATA     040h    ; каждой кнопки
          NAKOPL7_R    DATA     041h    ;-----------------------------

;РОНы обслуживания подпрограмм преобразования формы представления чисел
          IND_MILL_R   DATA     04Eh    ;РОН миллиардов дес числа
          IND_SOTMIL_R DATA     04Fh    ;РОН сотен миллионов дес числа
          IND_DESMIL_R DATA     050h    ;РОН десятков миллионов дес числа
          IND_MIL_R    DATA     051h    ;РОН миллионов дес числа
          IND_SOTTIS_R DATA     052h    ;РОН сотен тысяч дес числа
          IND_DESTIS_R DATA     053h    ;РОН десятков тысяч дес числа
          IND_TIS_R    DATA     054h    ;РОН тысяч дес числа
          IND_SOT_R    DATA     055h    ;РОН сотен дес числа
          IND_DES_R    DATA     056h    ;РОН десятков дес числа
          IND_ED_R     DATA     057h    ;РОН единиц дес числа

          BYTE_0_R     DATA     05Bh    ;байт 0 упакованного двоичн-дес числа
          BYTE_1_R     DATA     05Ch    ;байт 1 упакованного двоичн-дес числа
          BYTE_2_R     DATA     05Dh    ;байт 2 упакованного двоичн-дес числа
          BYTE_3_R     DATA     05Eh    ;байт 3 упакованного двоичн-дес числа
          BYTE_4_R     DATA     05Fh    ;байт 4 упакованного двоичн-дес числа

          DAC_L_R      DATA     068h    ; РОНы предельного значения числа
          DAC_H_R      DATA     069h    ; дискрет выхода ЦАП

          ST_L_R       DATA     06Ah    ; РОНы текущего значения числа
          ST_H_R       DATA     06Bh    ; дискрет выхода ЦАП
```

```
;Константы
          NACH_ADR     EQU      000h    ;начальный адрес обнуления РОНов
          KON_ADR      EQU      07Fh    ;конечный адрес обнуления РОНов
          POROG_K      EQU      10      ;порог подавления дребезга кнопок

;Начало исполняемого кода-------------------------------------------------------
               ORG 0h
           AJMP       Lab_START           ;идти на начало осн программы

;Начало осн программы-----------------------------------------------------------
               ORG 100h
Lab_START: MOV        SP,#080h            ;определить указатель стека
           MOV        PLLCON,#00000000b   ;уст макс частоту ядра (12,58 МГц)
           NOP                            ;
           LCALL      Pod_INIT_RSN        ;иниц РСН
           LCALL      Pod_INIT_RON        ;иниц РОН

           LCALL      Pod_INIT_LCD        ;иниц ЖКИ

           LCALL      Pod_CLEAR_LCD       ;стирание ЖКИ

           MOV        DACH,#0             ;
           MOV        DACL,#0             ;момент уст в 0 напряж ЦАП
           MOV        DAC_H_R,#HIGH(64)   ;
           MOV        DAC_L_R,#LOW(64)    ;нач уст предельного знач на вых ЦАП

           MOV        R0,#BYTE_0_R        ;
           MOV        R1,#DAC_L_R         ;
           MOV        DAC_H_R+1,#0        ; обнуление ст незначащих РОНов
           MOV        DAC_H_R+2,#0        ;------------------------------
           LCALL      B32BCD              ;преобр числа дискрет из дв в дв-дес

           MOV        R0,#BYTE_0_R        ;
           MOV        R1,#IND_MILL_R      ;
           LCALL      BCD10BCD           ;пр числ дискр из уп дв-дес в неуп дес

           MOV        R0,#IND_MILL_R      ; индицировать с РОНа IND_MILL_R
           MOV        R1,#0               ; индицировать с адр 0 ЖКИ
           LCALL      Pod_IND_10ZN        ;индикация числа дискрет
;Начало основного цикла---------------------------------------------------------
La_OSN:    NOP                            ;метка возврата в осн цикле

           MOV        ACC,ST_L_R          ;-----------------------------------
           ADD        A,#1                ;
           JNC        La_1                ;
           INC        ST_H_R              ; блок организации ступенчатого
La_1:      MOV        ST_L_R,ACC          ; возрастания
           CJNE       A,DAC_L_R,La_5      ; выходного напряжения ЦАП
           MOV        ACC,ST_H_R          ; до предельн значения, заложенного
           CJNE       A,DAC_H_R,La_5      ;
           MOV        ST_L_R,#0           ;   в  DAC_H_R, DAC_L_R
           MOV        ST_H_R,#0           ;-----------------------------------

La_5:      MOV        DACH,ST_H_R         ;
           MOV        DACL,ST_L_R         ;момент модификации вых напряж ЦАП

;Блок, предшествующий вызову подпрограммы опроса кнопки 0.
           MOV        R0,#KNOP0_R         ;
           MOV        R1,#NAKOPL0_R       ;
           LCALL      Pod_OPR_KNOP0       ;
;Блок, следующий за вызовом подпрограммы опроса кнопки (уст факта ее наж)
           MOV        ACC,@R0             ;
           JNB        ACC_1,La_OSN        ;

           CLR        ACC_1               ;кн была нажата, сброс флага нажат кн
           MOV        @R0,ACC             ;

           MOV        ACC,DAC_L_R         ;---------------------------------
```

```
            ADD         A,#64                ; Блок перебора числа дискрет внутри
            JC          La_10                ; пространства предельных значений
            MOV         DAC_L_R,ACC          ; 64 - 4032   с шагом 64
            SJMP        La_50                ;
La_10:      MOV         DAC_L_R,ACC          ;
            INC         DAC_H_R              ;
            MOV         ACC,DAC_H_R          ;
            CJNE        A,#010h,La_50        ;
            MOV         DAC_L_R,#LOW(64)     ;
            MOV         DAC_H_R,#HIGH(64)    ;
La_50:      NOP                              ;---------------------------------

            MOV         DACH,#0              ;
            MOV         DACL,#0              ;момент обнуления напряжения ЦАП

            MOV         R0,#BYTE_0_R         ;
            MOV         R1,#DAC_L_R          ;
            MOV         DAC_H_R+1,#0         ; обнуление ст незначащих РОНов
            MOV         DAC_H_R+2,#0         ;-----------------------------
            LCALL       B32BCD              ;пр числа дискр из двоичн в двоичн-дес

            MOV         R0,#BYTE_0_R         ;
            MOV         R1,#IND_MILL_R       ;
            LCALL       BCD10BCD            ;пр числа дискр из уп дв-дес в неуп дес

            MOV         R0,#IND_MILL_R       ; индицировать с РОНа IND_MILL_R
            MOV         R1,#0                ; индицировать с адр 0 ЖКИ
            LCALL       Pod_IND_10ZN         ;индикация числа дискрет

            LJMP        La_OSN               ;закрыть основной цикл


;Подпрограммы-----------------------------------------------------------------

;-------------------------------------------------------------------------------
;Подпрограмма инициализации РСН.
;-------------------------------------------------------------------------------
Pod_INIT_RSN:
            MOV         PORT_KNOP,#11111111b ;сделать вх все линии порта кн
            MOV         PORT_IND,#00000000b  ;сделать вых все линии порта инд

;Блок настройки ЦАП
            MOV         DACCON,#00010111b    ;вых ЦАП - P1.7, 12-битный ЦАП,
                                             ;диапазон ЦАП - 0-AVdd, ЦАП разрешен,
            RET

;-------------------------------------------------------------------------------
;Подпрограмма инициализации РОНов. Обнуляются все РОНы с адресами
;от NACH_ADR до KON_ADR при помощи косвенной адресации.
;-------------------------------------------------------------------------------
Pod_INIT_RON:
            MOV         R0,#NACH_ADR         ;установка начального адреса
Lk_0:       MOV         @R0,#0               ;обнуление очередного РОНа
            INC         R0                   ;переход к следующему адресу
            CJNE        R0,#KON_ADR,Lk_0     ;не достигли ли последнего адреса ?
            MOV         @R0,#0               ;обнуление последнего РОНа
            RET                              ;да, выход

;-------------------------------------------------------------------------------
;Подпрограмма вывода на ЖКИ содержимого буфера из 10 РОНов в виде 10 знакомест.
;R0 должен указывать на РОН в буфере, содержимое которого индицируется первым.
;R1 должен содержать нач адрес знакоместа в ОЗУ ЖКИ.
;-------------------------------------------------------------------------------
Pod_IND_10ZN:
            MOV         R2,#0                ;нач уст ст цикла
Ii_0:       MOV         DATA_IND_R,@R0       ;
            MOV         ADR_IND_R,R1         ;
            LCALL       Pod_PER_DAT_LCD      ; индикация очередного символа
```

```
            INC         R0                  ;
            INC         R1                  ;
            INC         R2                  ;
            CJNE        R2,#10,Ii_0         ;
            RET

;Подключение модулей опроса кнопок, вывода данных на ЖКИ (с опросом ЖКИ)
;и преобразования представления чисел
          $INCLUDE   (C:\PR_ADUC\DAC\knop.asm)
          $INCLUDE   (C:\PR_ADUC\DAC\lcd_opr.asm)
          $INCLUDE   (C:\PR_ADUC\DAC\preobr.asm)
;Конец исполняемого кода
          END
```

Путем нажатий на кнопку «0» можно менять количество ступенек в одном импульсе и, как следствие, размах и длительность импульсов в последовательности. Максимальный размах импульсов равен AVDD. Временная длительность одной ступеньки – около 30 мкс. Это время, затрачиваемое программой на прохождение одного своего основного цикла, в течение которого происходит приращение числа, помещаемого затем в регистры данных ЦАП DACH, DACL на единицу, сравнение этого числа с некоторой константой (порогом), а также опрос кнопки «0». В каждом программном цикле выходное напряжение ЦАП возрастает на величину, соответствующую одной дискрете ЦАП в случае, если записываемое в DACH, DACL число еще не достигло порога. При достижении порога в DACH, DACL записываются нули, что вызывает спад импульса, и со следующего основного цикла программы начинается ступенчатый фронт нового импульса. При 12-разрядном режиме работы ЦАП значение напряжения, соответствующее уровню одной дискреты ЦАП, определяется как AVDD/4096, а максимальное количество ступенек в одном импульсе равно 4095. При таком большом количестве ступенек генерируемые программой импульсы выглядят на экране осциллографа как пилообразные с линейно возрастающим фронтом и почти мгновенным спадом.

В блоке программной обработки обнаружения нажатия на кнопку «0» производится изменение константы (порога) сравнения. Каждое нажатие на кнопку увеличивает эту константу на 64 в пределах от 64 до 4032. Текущее значение порога в дискретах индицируется на ЖКИ в десятичном представлении начиная с адреса 0 ОЗУ ЖКИ. Настойку модуля ЦАП производит подпрограмма Pod_INIT_RSN.

Поскольку количество выполняемых инструкций в разных проходах основного цикла программы будет неодинаковым из-за наличия в алгоритме ветвлений, частота такой импульсной последовательности будет нестабильной. Задача достижения стабильности частоты последовательности генерируемых импульсов легко решается с использованием механизма прерываний от какого-либо из системных таймеров ADuC824. Такое решение иллюстрируется программой, исходный текст которой содержится в файле dac_tim.asm (листинг 3.23).

Листинг 3.23. Формирование напряжения с помощью прерываний

```
;-------------------------------------------------------------------------------
;Демонстрационная программа использования ЦАП  и таймера-счетчика T0  ADuC824.
;
;Данные в ЖКИ передаются по 4-битной шине, состояние ЖКИ опрашивается.
;На выходе ЦАП P1.7 формируется ступенчато возрастающее напряжение от нуля до
;некоторого верхнего значения. Верхнее значение определяется десятичным числом,
;индицируемым с адреса 0 ОЗУ ЖКИ.
;При нажатии на кнопку 0 производится перебор возможных значений этого числа
;с некоторым шагом.
;Используются прерывания при переполнении системного таймера T0.
;Они синхронизируют ступенчатое возрастание напряжения на выходе ЦАП.
;-------------------------------------------------------------------------------
          $INCLUDE  (C:\ADuC\mod824)
          $INCLUDE  (C:\PR_ADUC\DAC_TIM\824.inc)
;-------------------------------------------------------------------------------
;Описание битов, регистров и констант
;-------------------------------------------------------------------------------
;Порты и линии ввода-вывода
          PORT_KNOP    EQU      P0      ;порт кнопок

          PORT_IND     EQU      P2      ;порт индикации

          PORT_IND_0   EQU      P2_0    ;-----------------------------
          PORT_IND_1   EQU      P2_1    ;
          PORT_IND_2   EQU      P2_2    ; выходы индикации
          PORT_IND_3   EQU      P2_3    ;
          PORT_IND_4   EQU      P2_4    ;
          PORT_IND_5   EQU      P2_5    ;
          PORT_IND_6   EQU      P2_6    ;
          PORT_IND_7   EQU      P2_7    ;-----------------------------

          RW           EQU      PORT_IND_1 ;--------------------------
          RS           EQU      PORT_IND_2 ; линии управления ЖКИ
          E            EQU      PORT_IND_3 ;--------------------------

          _IN_KNOP0    EQU      P0_0    ;-----------------------------
          _IN_KNOP1    EQU      P0_1    ;
          _IN_KNOP2    EQU      P0_2    ;  входы кнопок
          _IN_KNOP3    EQU      P0_3    ;
          _IN_KNOP4    EQU      P0_4    ;
          _IN_KNOP5    EQU      P0_5    ;
          _IN_KNOP6    EQU      P0_6    ;
          _IN_KNOP7    EQU      P0_7    ;-----------------------------

;РОНы обслуживания ЖКИ и кнопок
          ADR_IND_R    DATA     030h    ;-----------------------------
          COM_IND_R    DATA     030h    ; РОНы обслуживания ЖКИ
          DATA_IND_R   DATA     031h    ;-----------------------------

          KNOP0_R      DATA     032h    ;-----------------------------
          KNOP1_R      DATA     033h    ;
          KNOP2_R      DATA     034h    ; РОНы, содержащие
          KNOP3_R      DATA     035h    ;
          KNOP4_R      DATA     036h    ; флаги нажатия и удержания
          KNOP5_R      DATA     037h    ;
          KNOP6_R      DATA     038h    ; каждой кнопки
          KNOP7_R      DATA     039h    ;-----------------------------

          NAKOPL0_R    DATA     03Ah    ;-----------------------------
          NAKOPL1_R    DATA     03Bh    ;
          NAKOPL2_R    DATA     03Ch    ; РОНы накопления
          NAKOPL3_R    DATA     03Dh    ;
          NAKOPL4_R    DATA     03Eh    ; значения подавления дребезга
          NAKOPL5_R    DATA     03Fh    ;
          NAKOPL6_R    DATA     040h    ; каждой кнопки
          NAKOPL7_R    DATA     041h    ;-----------------------------
```

```
;РОНы обслуживания подпрограмм преобразования формы представления чисел
          IND_MILL_R   DATA     04Eh    ;РОН миллиардов дес числа
          IND_SOTMIL_R DATA     04Fh    ;РОН сотен миллионов дес числа
          IND_DESMIL_R DATA     050h    ;РОН десятков миллионов дес числа
          IND_MIL_R    DATA     051h    ;РОН миллионов дес числа
          IND_SOTTIS_R DATA     052h    ;РОН сотен тысяч дес числа
          IND_DESTIS_R DATA     053h    ;РОН десятков тысяч дес числа
          IND_TIS_R    DATA     054h    ;РОН тысяч дес числа
          IND_SOT_R    DATA     055h    ;РОН сотен дес числа
          IND_DES_R    DATA     056h    ;РОН десятков дес числа
          IND_ED_R     DATA     057h    ;РОН единиц дес числа

          BYTE_0_R     DATA     05Bh    ;байт 0 упакованного двоичн-дес числа
          BYTE_1_R     DATA     05Ch    ;байт 1 упакованного двоичн-дес числа
          BYTE_2_R     DATA     05Dh    ;байт 2 упакованного двоичн-дес числа
          BYTE_3_R     DATA     05Eh    ;байт 3 упакованного двоичн-дес числа
          BYTE_4_R     DATA     05Fh    ;байт 4 упакованного двоичн-дес числа

          DAC_L_R      DATA     068h    ; РОНы предельного значения числа
          DAC_H_R      DATA     069h    ; дискрет выхода ЦАП

          ST_L_R       DATA     06Ah    ; РОНы текущего значения числа
          ST_H_R       DATA     06Bh    ; дискрет выхода ЦАП

;Константы
          NACH_ADR     EQU      000h    ;начальный адрес обнуления РОНов
          KON_ADR      EQU      07Fh    ;конечный адрес обнуления РОНов
          POROG_K      EQU      10      ;порог подавления дребезга кнопок

;Начало исполняемого кода-----------------------------------------------------
                ORG 0h
           AJMP       Lab_START           ;идти на начало осн программы

                ORG 000Bh
           AJMP       Lab_TIM0            ;идти на обр прерыв от T0
;Блок обработки прерываний от T0  ---------------------------------------------
;Прерывания обрабатываются по переполнению T0.
;Прерывания должны быть разрешены.
Lab_TIM0:  CLR        EA                  ;----------------------------------
           PUSH       PSW                 ; глоб запрет прер и сохр контекста
           PUSH       ACC                 ;----------------------------------

           MOV        TH0,#HIGH(65505)    ;Загрузка в T0 константы
           MOV        TL0,#LOW(65505)     ;(уст периодичности прерываний от T0)

           MOV        ACC,ST_L_R          ;----------------------------------
           ADD        A,#1                ;
           JNC        La_1                ;
           INC        ST_H_R              ; блок организации ступенчатого
La_1:      MOV        ST_L_R,ACC          ; возрастания
           CJNE       A,DAC_L_R,La_5      ; выходного напряжения ЦАП
           MOV        ACC,ST_H_R          ; до предельн значения, заложенного
           CJNE       A,DAC_H_R,La_5      ;
           MOV        ST_L_R,#0           ;   в  DAC_H_R, DAC_L_R
           MOV        ST_H_R,#0           ;----------------------------------

La_5:      MOV        DACH,ST_H_R         ;
           MOV        DACL,ST_L_R         ;момент модификации вых напряж ЦАП

;Блок возврата из прерываний-------------------------------------------------
Lab_RETI:  POP        ACC                 ;----------------------------------
           POP        PSW                 ; восст контекста и глоб разр прер
           SETB       EA                  ;----------------------------------
           RETI                           ;возврат из блока обраб прерываний


;Начало осн программы---------------------------------------------------------
                ORG 100h
Lab_START: MOV        SP,#080h            ;определить указатель стека
```

```
            MOV         PLLCON,#00000000b    ;уст макс частоту ядра (12,58 МГц)
            NOP                              ;
            LCALL       Pod_INIT_RSN         ;иниц РСН

            LCALL       Pod_INIT_RON         ;иниц РОН

            LCALL       Pod_INIT_LCD         ;иниц ЖКИ

            LCALL       Pod_CLEAR_LCD        ;стирание ЖКИ

            MOV         DACH,#0              ;
            MOV         DACL,#0              ;момент уст в 0 напряж ЦАП
            MOV         DAC_H_R,#HIGH(64)    ;
            MOV         DAC_L_R,#LOW(64)     ;нач уст предельн знач на вых ЦАП

            MOV         R0,#BYTE_0_R         ;
            MOV         R1,#DAC_L_R          ;
            MOV         DAC_H_R+1,#0         ; обнуление ст незначащих РОНов
            MOV         DAC_H_R+2,#0         ;------------------------------
            LCALL       B32BCD               ;преобр числа дискрет из дв в дв-дес

            MOV         R0,#BYTE_0_R         ;
            MOV         R1,#IND_MILL_R       ;
            LCALL       BCD10BCD           ;пр числ дискр из уп дв-дес в неуп дес

            MOV         R0,#IND_MILL_R       ; индицировать с РОНа IND_MILL_R
            MOV         R1,#0                ; индицировать с адр 0 ЖКИ
            LCALL       Pod_IND_10ZN         ;индикация числа дискрет

            SETB        EA                   ;разрешить прерывания глобально

;Начало основного цикла-------------------------------------------------------
La_OSN:     NOP                              ;метка возврата в осн цикле

;Блок, предшествующий вызову подпрограммы опроса кнопки 0.
            MOV         R0,#KNOP0_R          ;
            MOV         R1,#NAKOPL0_R        ;
            LCALL       Pod_OPR_KNOP0        ;
;Блок, следующий за вызовом подпрограммы опроса кнопки (уст факта ее наж)
            MOV         ACC,@R0              ;
            JNB         ACC_1,La_OSN         ;
            CLR         ACC_1                ;кн была нажата, сброс флага нажат кн
            MOV         @R0,ACC              ;
            CLR         EA                   ;запретить прерывания глобально

            MOV         ACC,DAC_L_R          ;---------------------------------
            ADD         A,#64                ; Блок перебора числа дискрет внутри
            JC          La_10                ; пространства предельных значений
            MOV         DAC_L_R,ACC          ; 64 - 4032   с шагом 64
            SJMP        La_50                ;
La_10:      MOV         DAC_L_R,ACC          ;
            INC         DAC_H_R              ;
            MOV         ACC,DAC_H_R          ;
            CJNE        A,#010h,La_50        ;
            MOV         DAC_L_R,#LOW(64)     ;
            MOV         DAC_H_R,#HIGH(64)    ;
La_50:      NOP                              ;---------------------------------

            MOV         DACH,#0              ;
            MOV         DACL,#0              ;момент обнуления напряжения ЦАП

            MOV         R0,#BYTE_0_R         ;
            MOV         R1,#DAC_L_R          ;
            MOV         DAC_H_R+1,#0         ; обнуление ст незначащих РОНов
            MOV         DAC_H_R+2,#0         ;------------------------------
            LCALL       B32BCD             ;пр числа дискр из двоичн в двоичн-дес

            MOV         R0,#BYTE_0_R         ;
            MOV         R1,#IND_MILL_R       ;
            LCALL       BCD10BCD           ;пр числа дискр из уп дв-дес в неуп дес
```

```
          MOV       R0,#IND_MILL_R       ; индицировать с РОНа IND_MILL_R
          MOV       R1,#0                ; индицировать с адр 0 ЖКИ
          LCALL     Pod_IND_10ZN         ;индикация числа дискрет

          SETB      EA                   ;разрешить прерывания глобально
          LJMP      La_OSN               ;закрыть основной цикл


;Подпрограммы-----------------------------------------------------------------

;-------------------------------------------------------------------------------
;Подпрограмма инициализации РСН.
;-------------------------------------------------------------------------------
Pod_INIT_RSN:
          MOV       PORT_KNOP,#11111111b ;сделать вх все линии порта кн
          MOV       PORT_IND,#00000000b  ;сделать вых все линии порта индик

;Блок настройки ЦАП
          MOV       DACCON,#00010111b    ;вых ЦАП - P1.7, 12-битный ЦАП,
                                         ;диапазон ЦАП - 0-AVdd, ЦАП разрешен.
;Блок настройки T0
          MOV       TMOD,#00110001b      ;T1 остановлен, T0 - 16-битный таймер.
          MOV       TCON,#00010000b      ;T1 отключен, T0 включен
          MOV       IE,#00000010b        ;разреш прерыв от T0 и запретить глоб
          MOV       TH0,#0               ; Нач загрузка в T0 константы
          MOV       TL0,#0               ;
          RET

;-------------------------------------------------------------------------------
;Подпрограмма инициализации РОНов. Обнуляются все РОНы с адресами
;от NACH_ADR до KON_ADR при помощи косвенной адресации.
;-------------------------------------------------------------------------------
Pod_INIT_RON:
          MOV       R0,#NACH_ADR         ;установка начального адреса
Lk_0:     MOV       @R0,#0               ;обнуление очередного РОНа
          INC       R0                   ;переход к следующему адресу
          CJNE      R0,#KON_ADR,Lk_0     ;не достигли ли последнего адреса ?
          MOV       @R0,#0               ;обнуление последнего РОНа
          RET                            ;да, выход

;-------------------------------------------------------------------------------
;Подпрограмма вывода на ЖКИ содержимого буфера из 10 РОНов в виде 10 знакомест.
;R0 должен указывать на РОН в буфере, содержимое которого индицируется первым.
;R1 должен содержать нач адрес знакоместа в ОЗУ ЖКИ.
;-------------------------------------------------------------------------------
Pod_IND_10ZN:
          MOV       R2,#0                ;нач уст ст цикла
Ii_0:     MOV       DATA_IND_R,@R0       ;
          MOV       ADR_IND_R,R1         ;
          LCALL     Pod_PER_DAT_LCD      ; индикация очередного символа
          INC       R0                   ;
          INC       R1                   ;
          INC       R2                   ;
          CJNE      R2,#10,Ii_0          ;
          RET

;Подключение модулей опроса кнопок, вывода данных на ЖКИ (с опросом ЖКИ)
;и преобразования представления чисел
          $INCLUDE  (C:\PR_ADUC\DAC_TIM\knop.asm)
          $INCLUDE  (C:\PR_ADUC\DAC_TIM\lcd_opr.asm)
          $INCLUDE  (C:\PR_ADUC\DAC_TIM\preobr.asm)

;Конец исполняемого кода
          END
```

Ступенчатое приращение выходного напряжения ЦАП здесь производится не в основной программе, а в блоке обработки прерываний от переполнения таймера-счетчика 0, который программно сконфигурирован как 16-разрядный таймер. Частота генерируемых программой импульсов будет стабильной, если суммарное время, затрачиваемое на передачу управления по адресу вектора прерывания, обработку запроса прерывания и возврат в основную программу, меньше, чем интервал времени между переполнениями таймера-счетчика. Другими словами, очередной запрос прерывания должен поступать, когда предыдущие прерывание уже обработано. В предлагаемой программе периодичность переполнения T0 задается путем записи в его старшую и младшую половины TH0, TL0 некоторой константы N. Запись производится сразу после очередного переполнения (в начале блока обработки прерываний). Период прерываний по переполнению T0 таким образом будет равен (65536 – N)/12 тактов частоты ядра (частоты с выхода системы ФАПЧ). Основной цикл программы теперь включает в себя только опрос кнопки «0».

3.10. Использование модуля UART

Использование встроенного порта UART ADuC824 иллюстрируется демонстрационной программой, исходный текст которой содержится в файле uart.asm (листинг 3.24). Для экспериментов с UART, как и в предыдущем случае, требуется макет, содержащий только подключенные к ADuC824 кнопки управления и ЖКИ. Программа позволяет организовать обмен данными в асинхронном (старт-стопном) режиме между ADuC824 и каким-либо другим устройством, имеющим UART.

Листинг 3.24. Использование интерфейса UART

```
;-------------------------------------------------------------------------------
;Демонстрационная программа обслуживания UART для ADuC824.
;
;Данные в ЖКИ передаются по 4-битной шине, состояние ЖКИ опрашивается,
;подпрограмма опроса возвращает управление, когда ЖКИ готов к приему данных.
;
;При нажатии на кн 0 производится перебор кодов символов для передачи через ;UART.
;При нажатии на кн 1 производится передача через UART одного байта.
;Параметры настройки UART: скорость - 9600 бит/с, N-8-1.
;
;Прием байта через UART от внешнего устройства (ПЭВМ) может происходить
;в любой момент времени (определяется внешним устройством).
;
;Передаваемый байт (символ) отображается на ЖКИ по адресу ОЗУ ЖКИ  65,
;Принятый байт (символ)  отображается на ЖКИ по адресу ОЗУ ЖКИ  70.
;-------------------------------------------------------------------------------
          $INCLUDE  (C:\ADuC\mod824)
          $INCLUDE  (C:\PR_ADUC\UART\824.inc)
;-------------------------------------------------------------------------------
;Описание битов, регистров и констант
;-------------------------------------------------------------------------------
```

```
          PORT_KNOP    EQU      P0      ;порт кнопок

          PORT_IND     EQU      P2      ;порт индикации

          _IN_KNOP0    EQU      P0_0    ;-----------------------------
          _IN_KNOP1    EQU      P0_1    ;
          _IN_KNOP2    EQU      P0_2    ;  входы кнопок
          _IN_KNOP3    EQU      P0_3    ;
          _IN_KNOP4    EQU      P0_4    ;
          _IN_KNOP5    EQU      P0_5    ;
          _IN_KNOP6    EQU      P0_6    ;
          _IN_KNOP7    EQU      P0_7    ;-----------------------------

          POROG_K      EQU      50      ;порог подавления дребезга
          NACH_ADR     EQU      000h    ;начальный адрес обнуления РОНов
          KON_ADR      EQU      07Fh    ;конечный адрес обнуления РОНов

          KNOP0_R      DATA     032h    ;-----------------------------
          KNOP1_R      DATA     033h    ;
          KNOP2_R      DATA     034h    ; регистры, содержащие
          KNOP3_R      DATA     035h    ;
          KNOP4_R      DATA     036h    ; флаги нажатия и удержания
          KNOP5_R      DATA     037h    ;
          KNOP6_R      DATA     038h    ; каждой кнопки
          KNOP7_R      DATA     039h    ;-----------------------------

          NAKOPL0_R    DATA     03Ah    ;-----------------------------
          NAKOPL1_R    DATA     03Bh    ;
          NAKOPL2_R    DATA     03Ch    ; регистры накопления
          NAKOPL3_R    DATA     03Dh    ;
          NAKOPL4_R    DATA     03Eh    ; значения подавления дребезга
          NAKOPL5_R    DATA     03Fh    ;
          NAKOPL6_R    DATA     040h    ; каждой кнопки
          NAKOPL7_R    DATA     041h    ;-----------------------------

          PORT_IND_0   EQU      P2_0    ;-----------------------------
          PORT_IND_1   EQU      P2_1    ;
          PORT_IND_2   EQU      P2_2    ; выходы индикации
          PORT_IND_3   EQU      P2_3    ;
          PORT_IND_4   EQU      P2_4    ;
          PORT_IND_5   EQU      P2_5    ;
          PORT_IND_6   EQU      P2_6    ;
          PORT_IND_7   EQU      P2_7    ;-----------------------------

          RW           EQU      PORT_IND_1 ;--------------------------
          RS           EQU      PORT_IND_2 ; линии обслуживания ЖКИ
          E            EQU      PORT_IND_3 ;--------------------------

          ADR_IND_R    DATA     030h    ;-----------------------------
          COM_IND_R    DATA     030h    ; РОНы обслуживания ЖКИ
          DATA_IND_R   DATA     031h    ;-----------------------------

          RX_UART_R    DATA     042h    ;РОН сохр принятого по UART байта
          TX_UART_R    DATA     043h    ;РОН сохр передаваемого по UART байта

;Флаги
          _ID_PER      BIT      00h     ;флаг "идет передача через UART"
          _PR_BYTE     BIT      01h     ;флаг "принят байт через UART"

;Начало исполняемого кода-------------------------------------------------------
               ORG 0h
           LJMP       Lab_START            ;идти на начало осн программы

               ORG 023h
           LJMP       Lab_UART             ;идти на обр прерыв от UART

;Блок обработки прерываний от UART----------------------------------------------
;Прерывания обрабатываются по завершению передачи и по завершению приема байта.
```

```
;Основная программа может судить об идущей в данный момент передаче байта по
;состоянию флага _ID_PER "идет передача", а о состоявшемся приеме байта - по
;установленному флагу _PR_BYTE "принят байт".  Сброс _PR_BYTE возлагается на
;основную программу. Установка _ID_PER (т. е. инициация передачи) возлагается
;на основную программу.
;Прерывания должны быть разрешены.
Lab_UART:  CLR        EA                  ;---------------------------------
           PUSH       PSW                 ; глоб запрет прер и сохр контекста
           PUSH       ACC                 ;---------------------------------

           JNB        TI,La_U0            ;окончилась ли передача ?
           CLR        TI                  ;да, сбросить флаг окончания передачи
           CLR        _ID_PER             ;    сбросить флаг "идет передача"
La_U0:     JBC        RI,La_U1            ;окончился ли прием ?
           AJMP       Lab_RETI            ;нет (или его вообще не было), выйти
La_U1:     SETB       _PR_BYTE            ;да, уст флаг "принят байт"
           MOV        RX_UART_R,SBUF      ;сохранить принятый байт
           AJMP       Lab_RETI            ;выйти

;Блок возврата из прерываний-----------------------------------------------------
Lab_RETI:  POP        ACC                 ;---------------------------------
           POP        PSW                 ; восст контекста и глоб разр прер
           SETB       EA                  ;---------------------------------
           RETI                           ;возврат из блока обраб прерываний

;Начало основной программы-------------------------------------------------------
                ORG 100h
Lab_START: MOV        SP,#080h            ;определить указатель стека
           MOV        PLLCON,#00000000b   ;уст макс частоту ядра (12,58 МГц)
           NOP                            ;

           LCALL      Pod_INIT_RSN        ;иниц РСН
           LCALL      Pod_INIT_RON        ;иниц РОН

          MOV        R3,#021h          ;задание нач значения индицируемого байта

           LCALL      Pod_INIT_LCD        ;иниц ЖКИ

           LCALL      Pod_CLEAR_LCD       ;стирание ЖКИ

           LCALL      Pod_IND_PRIVET_R    ;индикация заставки "Привет"

;Начало основного цикла----------------------------------------------------------
La_OSN:    NOP                            ;метка возврата осн цикла

       ;Блок, предшествующий вызову подпрограммы опроса кнопки 0.
           MOV        R0,#KNOP0_R         ;
           MOV        R1,#NAKOPL0_R       ;
           LCALL      Pod_OPR_KNOP0       ;
       ;Блок, следующий за вызовом подпрограммы опроса кнопки (уст факта ее наж)
           MOV        ACC,@R0             ;
           JNB        ACC_1,La_0          ;

           CLR        ACC_1               ;кн была нажата, сброс флага нажат кн
           MOV        @R0,ACC             ;

           INC        R3                  ;---------------------------------
           CJNE       R3,#080h,La_00      ; перебор возможных значений байта
           MOV        R3,#021h            ;---------------------------------
La_00:     MOV        DATA_IND_R,R3       ;---------------------------------
           MOV        ADR_IND_R,#65       ; индикация  байта (символа)
           LCALL      Pod_PER_DAT_LCD     ;---------------------------------

       ;Блок, предшествующий вызову подпрограммы опроса кнопки 1.
La_0:      MOV        R0,#KNOP1_R         ;
           MOV        R1,#NAKOPL1_R       ;
           LCALL      Pod_OPR_KNOP1       ;
```

```
       ;Блок, следующий за вызовом подпрограммы опроса кнопки (уст факта ее наж)
           MOV        ACC,@R0              ;
           JNB        ACC_1,La_1           ;

           CLR        ACC_1                ;кн была нажата, сброс флага нажат кн
           MOV        @R0,ACC              ;

           MOV        TX_UART_R,R3         ;-----------------------------------
           LCALL      Pod_TX_UART          ; команда на передачу байта

La_1:      JNB        _PR_BYTE,La_OSN      ;флаг "принят байт" уст ?
           CLR        _PR_BYTE             ;да, сбросить флаг "принят байт"

           MOV        DATA_IND_R,RX_UART_R ;-----------------------------------
           MOV        ADR_IND_R,#70        ; индикация принятого байта (символа)
           LCALL      Pod_PER_DAT_LCD      ;-----------------------------------

           AJMP       La_OSN               ;закрытие осн цикла


;Подпрограммы-----------------------------------------------------------------

;-------------------------------------------------------------------------
;Подпрограмма инициализации РСН.
;-------------------------------------------------------------------------
Pod_INIT_RSN:
          MOV        PORT_KNOP,#11111111b ;сделать вх все линии порта кн
          MOV        PORT_IND,#00000000b  ;сделать вых все линии порта индик

;блок инициализации UART
          MOV        SCON,#01010000b    ;настройка UART: режим 1, используем T2
          MOV        T2CON,#00110100b   ;настройка T2
          MOV        RCAP2H,#0FFh       ; настройка UART на скорость 9600 бит/с
          MOV        RCAP2L,#0D7h         ;
          MOV        TH2,#0               ; сброс T2
          MOV        TL2,#0               ;
          MOV        IE,#10010000b        ;разреш прерыв от UART и прерыв глоб
           RET

;-------------------------------------------------------------------------------
;Подпрограмма инициализации РОНов. Обнуляются все РОНы с адресами
;от NACH_ADR до KON_ADR при помощи косвенной адресации.
;-------------------------------------------------------------------------------
Pod_INIT_RON:
          MOV        R0,#NACH_ADR         ;установка начального адреса
Lk_0:     MOV        @R0,#0               ;обнуление очередного РОНа
          INC        R0                   ;переход к следующему адресу
          CJNE       R0,#KON_ADR,Lk_0     ;не достигли ли последнего адреса ?
          MOV        @R0,#0               ;обнуление последнего РОНа
          RET                             ;да, выход

;-------------------------------------------------------------------------------
;Подпрограмма передачи байта через UART. Передаваемый байт следует предварительно
;поместить в РОН TX_UART_R. Должны быть разрешены прерывания от UART.
;-------------------------------------------------------------------------------
Pod_TX_UART:
          JB          _ID_PER,$           ; ожидаем окончания предыдущ передачи,
          JB          TI,$                 ; если она идет
          MOV         SBUF,TX_UART_R       ;начать передачу
          SETB        _ID_PER              ;уст флаг "идет передача"
          RET

;-------------------------------------------------------------------------
;Подпрограмма заставки, выводимой на ЖКИ после сброса ("Привет")
;-------------------------------------------------------------------------
Pod_IND_PRIVET_R:
           MOV        DATA_IND_R,#0A8h;П
           MOV        ADR_IND_R,#0    ;
```

```
            LCALL       Pod_PER_DAT_LCD ;
            MOV         DATA_IND_R,#070h;p
            MOV         ADR_IND_R,#1    ;
            LCALL       Pod_PER_DAT_LCD ;
            MOV         DATA_IND_R,#0B8h;и
            MOV         ADR_IND_R,#2    ;
            LCALL       Pod_PER_DAT_LCD ;
            MOV         DATA_IND_R,#0B3h;в
            MOV         ADR_IND_R,#3    ;
            LCALL       Pod_PER_DAT_LCD ;
            MOV         DATA_IND_R,#065h;е
            MOV         ADR_IND_R,#4    ;
            LCALL       Pod_PER_DAT_LCD ;
            MOV         DATA_IND_R,#0BFh;т
            MOV         ADR_IND_R,#5    ;
            LCALL       Pod_PER_DAT_LCD ;
            RET

;Подключение модулей опроса кнопок и вывода данных на ЖКИ (с опросом ЖКИ)
          $INCLUDE   (C:\PR_ADUC\UART\knop.asm)
          $INCLUDE   (C:\PR_ADUC\UART\lcd_opr.asm)

;Конец исполняемого кода
          END
```

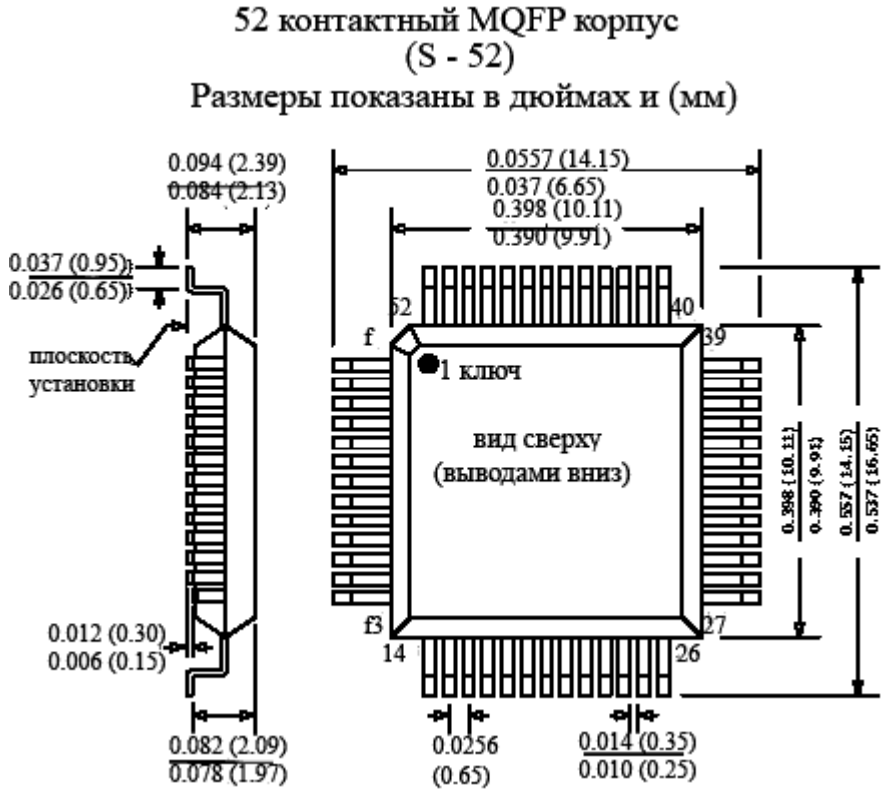
В качестве такого устройства использовался компьютер с запущенной на нем распространенной программой TELIX, эмулирующей режим простого терминала. Подобных программ существует великое множество (Terminal, «Удаленный доступ», «Одиссей» и т. п.). Используемый терминальной программой COM-порт компьютера должен быть настроен в ней следующим образом: скорость 9600 бит/с, контроль четности отсутствует, 8 бит в байте, 1 стоповый бит (N-8-1). В случае, если программа «видит» внешнее устройство, подключенное к COM-порту компьютера, в окне ее статуса должна появиться надпись «Online 00:00». Со стороны ADuC824 обмен с компьютером производится через тот же самый порт UART, через который осуществляется последовательная загрузка памяти (с использованием адаптера интерфейса RS-232 MAX232).

При нажатии на кнопку «0» на макете производится выбор кодов символов для передачи через UART, при этом ASCII-код текущего выбранного байта индицируется по адресу 65 ОЗУ ЖКИ. При нажатии на кнопку «1» производится передача этого байта через UART в ПК. ASCII-код переданного в компьютер байта сразу же отобразится в главном окне программы TELIX. Прием байта через UART от компьютера может происходить в любой момент времени. Каждый набранный на клавиатуре компьютера символ будет передаваться в виде байта через COM-порт в ADuC824. ASCII-код последнего принятого байта будет индицироваться по адресу 70 ОЗУ ЖКИ макета. Для контроля правильности приема в терминальной программе можно установить опцию «Local echo ON» («Локальное эхо включено»). Это даст возможность видеть набираемые (передаваемые через COM-порт) символы в главном окне программы.

3.11. Вопросы для самоконтроля

1. Что такое дребезг контактов и как с ним можно программно бороться?
2. Нарисуйте алгоритм программы обработки сигнала с механической кнопки.
3. Что такое «флаг» в программе и как его можно использовать?
4. Расскажите об интерфейсе ЖКИ со встроенным контроллером типа HD44780.
5. Можно ли передавать данные в ЖКИ на основе HD44780 не байтами, а данными другой размерности?
6. Можно ли использовать ОЗУ индикатора на основе HD44780 в качестве обыкновенного внешнего ОЗУ данных?
7. Каково назначение резистор R16 на рис. 3.2?
8. Как определить интегральную нелинейность встроенного в МК АЦП?
9. Нарисуйте схему подключения к встроенному АЦП внешнего сигнала.
10. Нарисуйте алгоритм измерения температуры встроенным датчиком.
11. Как проверить правильность записи данных в EEPROM МК?
12. Какие устройства имеют интерфейс SPI?
13. Как устроена микросхема DataFlash?
14. Нарисуйте структуру памяти микросхемы DataFlash.
15. Можно ли использовать микросхему DataFlash без подключения к выводу RDY/BUSY?
16. Как происходит запись и чтение страницы в DataFlash?
17. Нарисуйте схему подключения DataFlash к МК.
18. Расскажите основы реализации интерфейса I^2C в МК.
19. Какие в МК существуют ограничения в реализации интерфейса I^2C?
20. Нарисуйте алгоритм передачи и приема байта данных по интерфейсу I^2C.
21. Зачем нужны подтягивающие резисторы в интерфейсе I^2C?
22. Опишите последовательность сигналов на линиях SDA и SCL во время чтения и записи данных.
23. Как узнать адрес устройства, которое необходимо подключить к МК по интерфейсу I^2C?
24. Как организовать контроль состояния линии SCL ресурсами МК?
25. Какие вы знаете устройства, помимо микросхемы 24LC64, использующие интерфейс I^2C?
26. Что такое часы реального времени?
27. Как реализовать в МК часы реального времени?
28. Какова теоретическая погрешность хода часов реального времени в МК? Как можно уменьшить эту погрешность?
29. Как сконфигурировать ЦАП, встроенный в МК?
30. Как можно сформировать с помощью встроенного ЦАП синусоидальный сигнал? Какие достижимы предельные параметры такого сигнала?
31. Зачем нужно применять технику работы ЦАП по прерываниям? В каких задачах это может понадобиться?
32. Как с помощью модуля UART организовать взаимодействие с ПК?
33. Нарисуйте схему соединения МК с ПК по интерфейсу RS-232.

Приложение 1
Габаритные размеры и нумерация выводов корпуса ADuC824



Приложение 2

Предельно допустимые параметры ADuC824

(при t=25°C, если не оговаривается особо)

Напряжение AVDD относительно AGND – 0,3 ... +7 В.

Напряжение AVDD относительно DGND – 0,3 ... +7 В.

Напряжение DVDD от AGND – 0,3 ... +7 В.

Напряжение DVDD относительно DGND – 0,3 ... +7 В.

Напряжение AGND относительно DGND – 0,3 ... +0,3 В (контакты AGND и DGND замкнуты накоротко внутри корпуса ADuC824).

Напряжение AVDD относительно DVDD – 2 ... +5 В.

Аналоговое входное напряжение относительно AGND – 0,3 …AVDD +0,3В (применительно к контактам P1.2 – P1.7, работающим как аналоговые или цифровые входы).

Напряжение ИОН относительно AGND – 0,3 ... AVDD + 0,3 В.

Входной ток на входах AIN/REF 30 мА.

Напряжение цифрового входа относительно DGND – 0,3 ... DVDD +0,3 В.

Напряжение цифрового выхода относительно DGND – 0,3 ... DVDD +0,3 В.

Диапазон рабочих температур – 40 ... +85 °C.

Диапазон температур хранения – 65 ... +150 °C.

Температура перехода +150 °C.

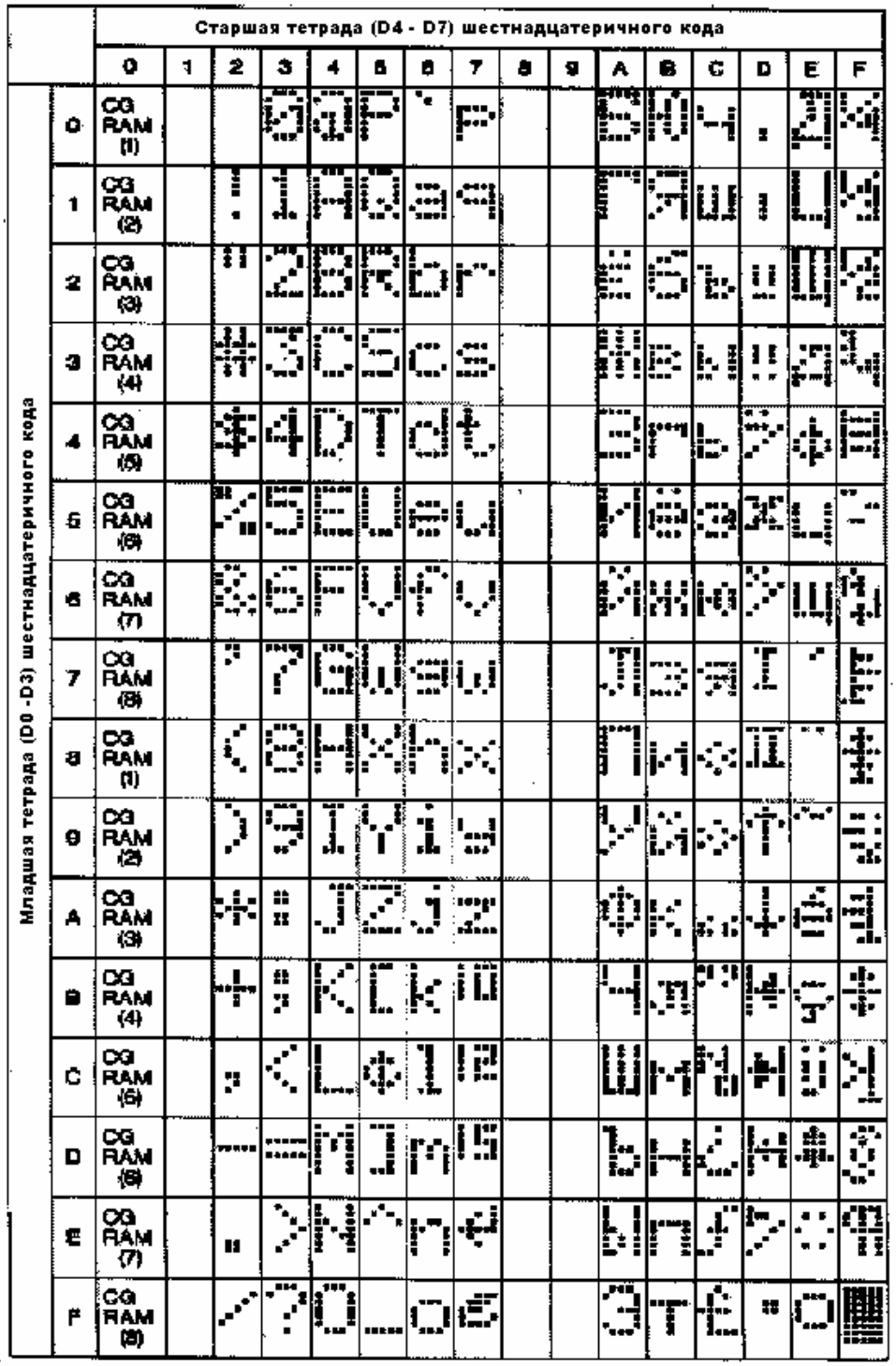
Температурное сопротивление θja 90 °C/Вт.

Температура выводов при пайке: в паровой фазе (60 с) +215 °C, инфракрасная (15 с) +220 °C.

Устройство чувствительно к электростатическим разрядам. Разряд напряжения до 4000 В может произойти неконтролируемым образом при простом прикосновении. Несмотря на то, что устройство имеет цепи защиты, для сохранения его работоспособности следует предпринять соответствующие меры защиты от статического электричества.

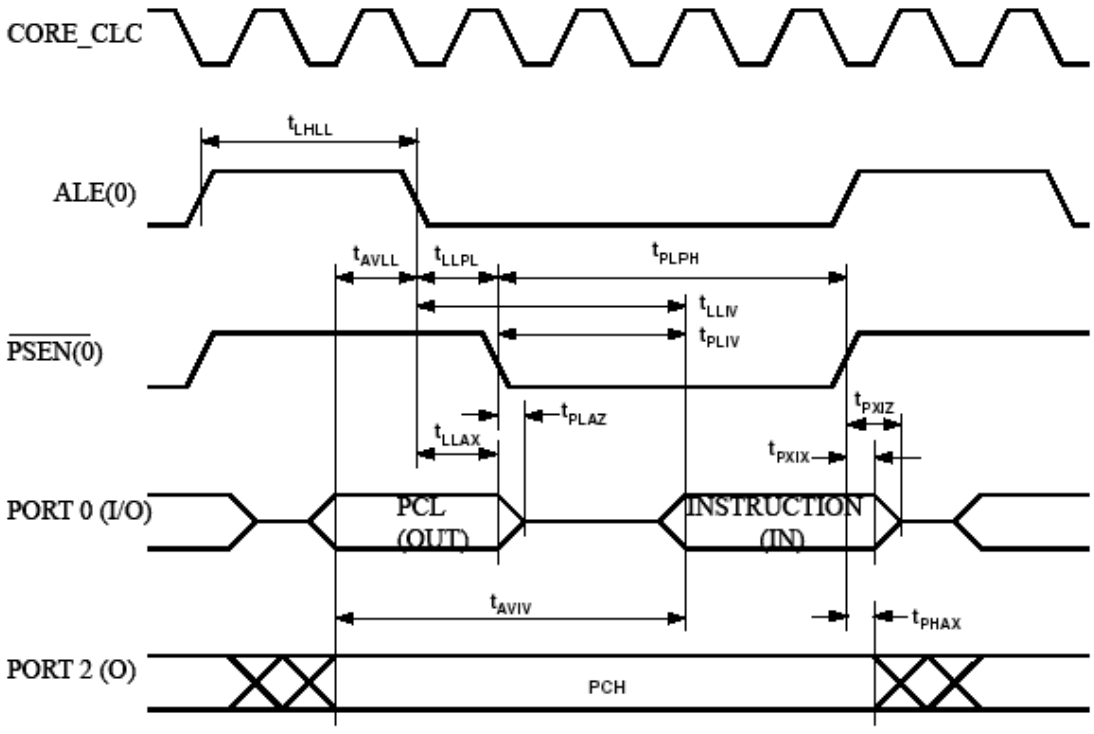
Приложение 3

Таблица кодов символов (фонтов) для русифицированного ЖКИ, с встроенным контроллером управления, совместимым с HD44780



Приложение 4
Временные параметры внешней памяти программ

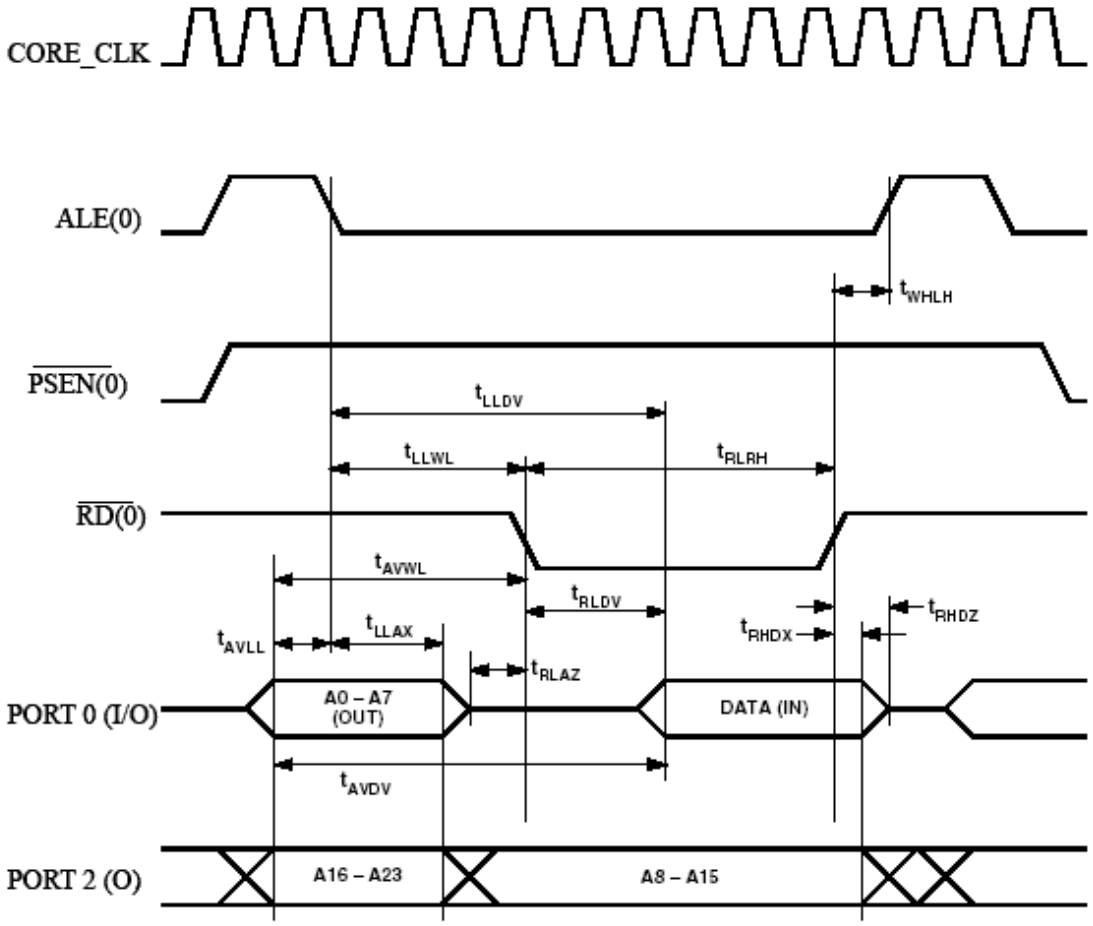
Параметр	12,58 МГц частота ядра		Переменная частота ядра		Единица
	Мин	Макс	Мин	Макс	
ВНЕШНЯЯ ПАМЯТЬ ПРОГРАММ					
t_{LHLL} Длительность активного уровня ALE	119		$2t_{CORE} - 40$		нс
t_{AVLL} Предустановка адреса до сброса ALE	39		$t_{CORE} - 40$		нс
t_{LLAX} Удержание адреса после сброса ALE	49		$t_{CORE} - 30$		нс
t_{LLIV} Сброс ALE до установки команды		218		$4t_{CORE} - 100$	
t_{LLPL} Сброс ALE до сброса/PSEN	49		$t_{CORE} - 30$		нс
t_{PLPH} Длительность активного уровня/PSEN	193		$3\ t_{CORE} - 45$		нс
t_{PLIV} Сброс/PSEN до установки команды		133		$3\ t_{CORE} - 105$	нс
t_{PXIX} Удержание команды после/PSEN	0		0		
t_{PXIZ} Переход команды в плавающее состояние после/PSEN		54		$t_{CORE} - 25$	нс
t_{AVIV} Адрес до установки команды		292		$5\ t_{CORE} - 105$	нс
t_{PLAZ} Сброс /PSEN до перехода адреса в плавающее состояние		25		25	нс
t_{PHAX} Удержание адреса после установки/PSEN	0		0		нс



Цикл чтения внешней памяти программ

Приложение 5
Временные параметры чтения внешней памяти данных

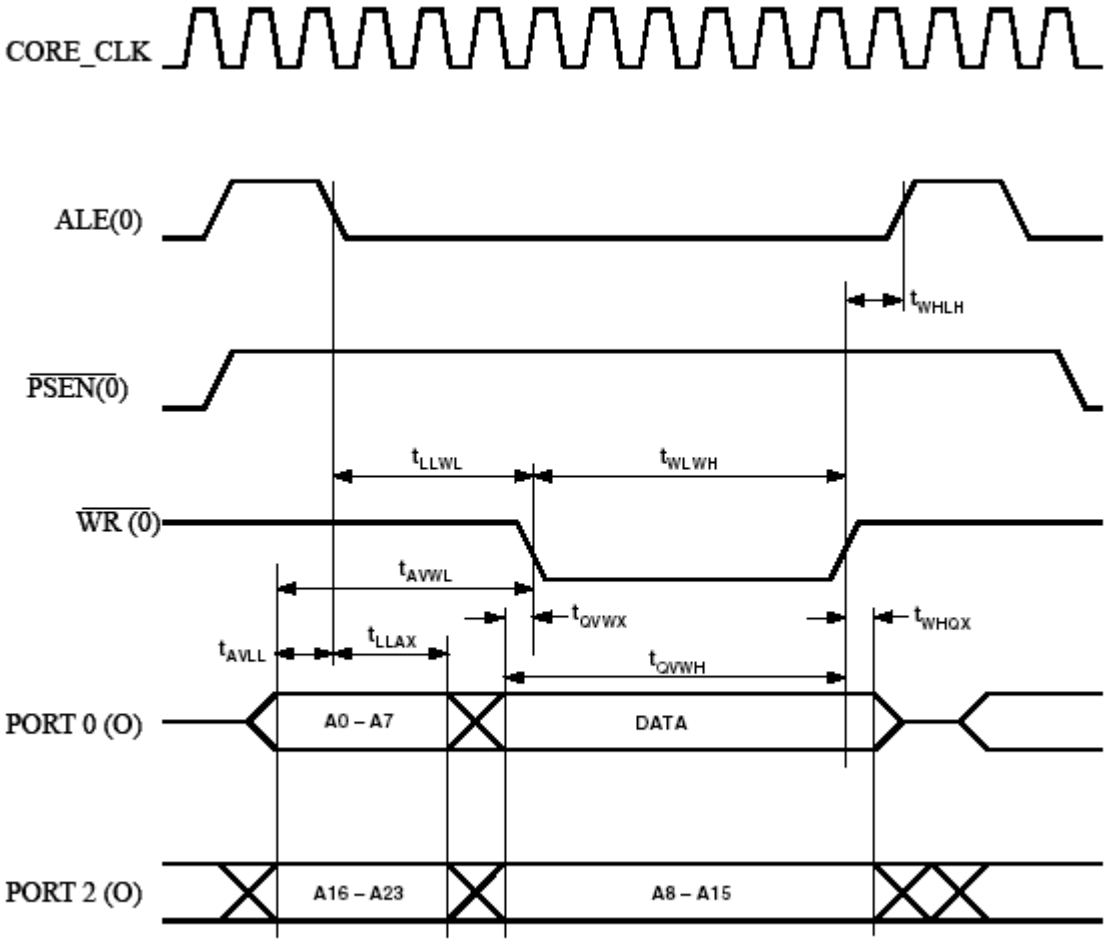
Параметр	12,58 МГц частота ядра Мин	Макс	Переменная частота ядра Мин	Макс	Единица
ЦИКЛ ЧТЕНИЯ ВНЕШНЕЙ ПАМЯТИ ДАННЫХ					
t_{RLRH} Длительность активного уровня/RD	377		$6t_{CORE} - 100$		нс
t_{AVLL} Предустановка адреса до сброса ALE	39		$t_{CORE} - 40$		нс
t_{LLAX} Удержание адреса после сброса ALE	44		$t_{CORE} - 35$		нс
t_{RLDV} Сброс/RD до установки данных		232		$5t_{CORE} - 165$	нс
t_{RHDX} Удержание данных и адреса после уст./RD	0		0		нс
t_{RHDZ} Переход данных в плавающ. сост.после уст./RD		89		$2\ t_{CORE} - 70$	нс
t_{LLDV} Сброс ALE до установки данных		486		$8\ t_{CORE} - 150$	нс
t_{AVDV} Адрес до установки данных		550		$9\ t_{CORE} - 165$	нс
t_{LLWL} Сброс ALE до сброса /RD	188	288	$3_{CORE} - 50$	$3\ t_{CORE} + 50$	нс
t_{AVWL} Переустановка адреса до сброса /RD	188		$4\ t_{CORE} - 130$		нс
t_{RLAZ} Сброс /RD после перехода адреса в плавающ. сост.		0		0	нс
t_{WHLH} Переустановка/RD до установки ALE	39	119	$t_{CORE} - 40$	$t_{CORE} + 40$	нс



Цикл чтения внешней памяти данных

Приложение 6
Временные параметры записи внешней памяти данных

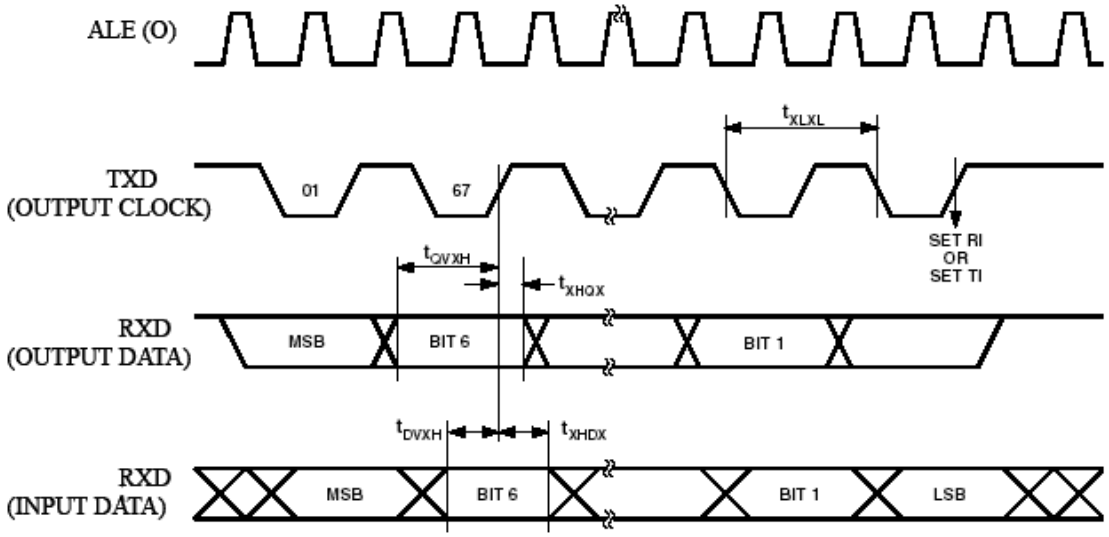
Параметр	12,58 МГц частота ядра Мин	Макс	Переменная частота ядра Мин	Макс	Единица
ЦИКЛ ЗАПИСИ ВНЕШНЕЙ ПАМЯТИ ДАННЫХ					
t_{WLWH} Длительность активного уровня/WR	377		$6t_{CORE} - 100$		нс
t_{AVLL} Предустановка адреса до сброса ALE	39		$t_{CORE} - 40$		нс
t_{LLAX} Удержание адреса после сброса ALE	44		$t_{CORE} - 35$		нс
t_{LLWL} Сброс ALE до сброса/WR	188	288	$3t_{CORE} - 50$	3tCORE +50	нс
t_{AVWL} Переустановка адреса до сброса /WR	188		$4t_{CORE} - 130$		нс
t_{QVWX} Переустановка данных до изменения/WR	29		$t_{CORE} - 50$		нс
t_{QVWH} Удержание данных до и во время/WR	406		$7\ t_{CORE} - 150$		нс
t_{WHQX} Удержание данных и адреса после/WR	29		$t_{CORE} - 50$		нс
t_{WHLH} Сброс/WR до сброса ALE	39	119	$t_{CORE} - 40$	$t_{CORE} + 40$	нс



Цикл записи внешней памяти данных

Приложение 7
Временные параметры интерфейса UART в режиме сдвигового регистра

Параметр	12,58 МГц частота ядра			Переменная частота ядра			Единица
	Мин	Тип	Макс	Мин	Тип	Макс	
ВРЕМЕННЫЕ ПАРАМЕНТЫ UART (режим сдвигового регистра)							
t_{XLXL} Период тактовой частоты последовательность порта	0,95			$12t_{CORE}$			мкс
t_{QVXH} Установка выходных данных до тактового сигнала	662			$10t_{CORE} – 133$			нс
t_{DVXH} Установка выходных данных до тактового сигнала	292			$2t_{CORE}$+–133			нс
t_{XHDX} Удержание выходных данных после тактового сигнала	0			0			нс
t_{XHQX} Удержание выходных данных после тактового сигнала	42			$2t_{CORE} –117$			нс

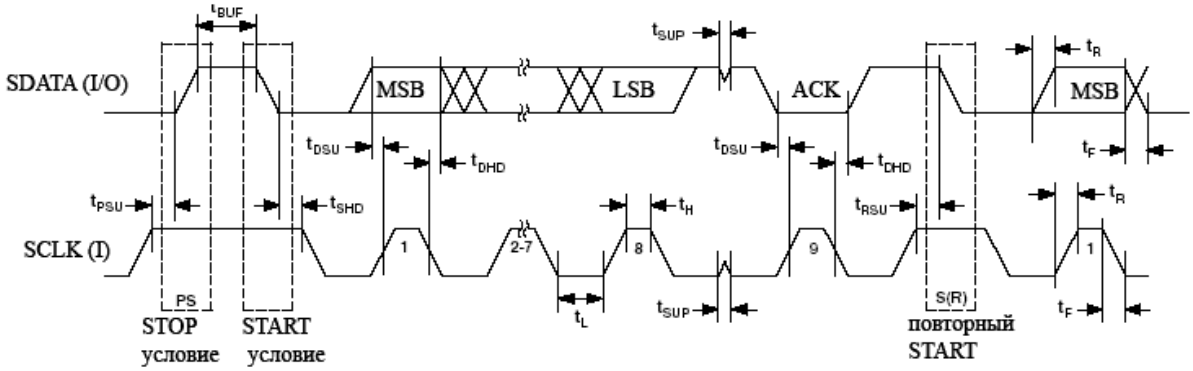


Временные параметры UART в режиме сдвигового регистра

Приложение 8
Временные параметры I²C-совместимого интерфейса

Параметр		Мин	Макс	Единица
	ВРЕМЕННЫЕ ПАРАМЕТРЫ I2C-СОВМЕСТИМОГО ИНТЕРФЕЙСА			
t_L	Длительность низкого уровня SCLOCK	4,7		мкс
t_H	Длительность высокого уровня SCLOCK	4,0		мкс
t_{SHD}	Время удержания в условиях START	0,6		мкс
t_{DSU}	Время установления данных	100		мкс
t_{DHD}	Время удержания данных		0,9	мкс
t_{RSU}	Время установки для повторного START	0,6		мкс
t_{PSU}	Время установки условия STOP	0,6		мкс
t_{BUF}	Время освобождения шины между условием STOP и условием START	1,3		мкс
t_R	Длительность фронта сигналов SCLOCK и SDATA		300	нс
t_F	Длительность спада сигналов SCLOCK и SDATA		300	нс
t_{SUP}*	Длительность подавляемого шумового выброса		50	нс

* На линиях SCLOCK и SDATA входные фильтры подавляют шумовые выбросы длительностью менее 50 нс



Временные параметры I2C-совместимого интерфейса

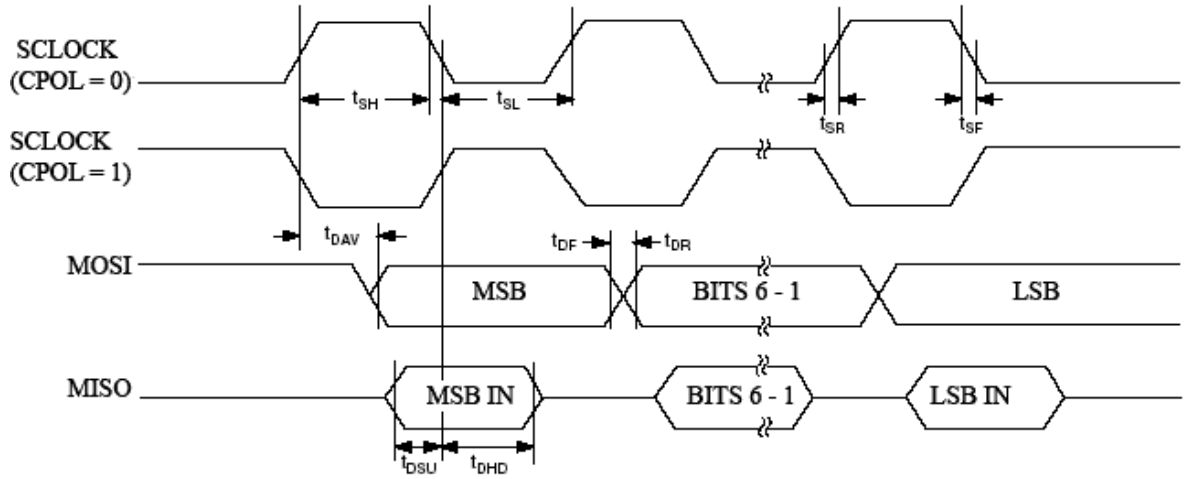
Приложение 9
Временные параметры интерфейса SPI в режиме Master (CPHA=1)

Параметр	Мин	Тип	Макс	Единица
ВРЕМЕННЫЕ ПАРАМЕТРЫ РЕЖИМА MASTER SPI(CPHA=1)				нс
t_{SL} Длительность низкого уровня SCLOCK		630		нс
t_{SH} Длительность высокого уровня SCLOCK		630		нс
t_{DAV} Выдача выходных данных после перепада SCLOCK			50	нс
t_{DSU} Время установления выходных данных до перепада SCLOCK	100			нс
t_{DHD} Время удержания выходных данных после перепада SCLOCK	100			
t_{DF} Длительность спада выходных данных		10	25	нс
t_{DR} Длительность фронта выходных данных		10	25	нс
t_{SR} Длительность фронта SCLOCK		10	25	нс
t_{SF} Длительность спада SCLOCK		10	25	нс

*Характеристики приводятся для следующих условий:

а. Биты выбора тактовой частоты ядра CD2, CD1, CD0 в специальном регистре PLLCON имеют значения соответственно 0, 1, 1, т. е. установлена тактовая частота ядра 1,57 МГц.

б. Биты выбора скорости SPI SPR1, SPR0 в специальном регистре SPICON имеют значения 0, 0.



Временные параметры режима Master SPI (CPHA=1)

Приложение 10

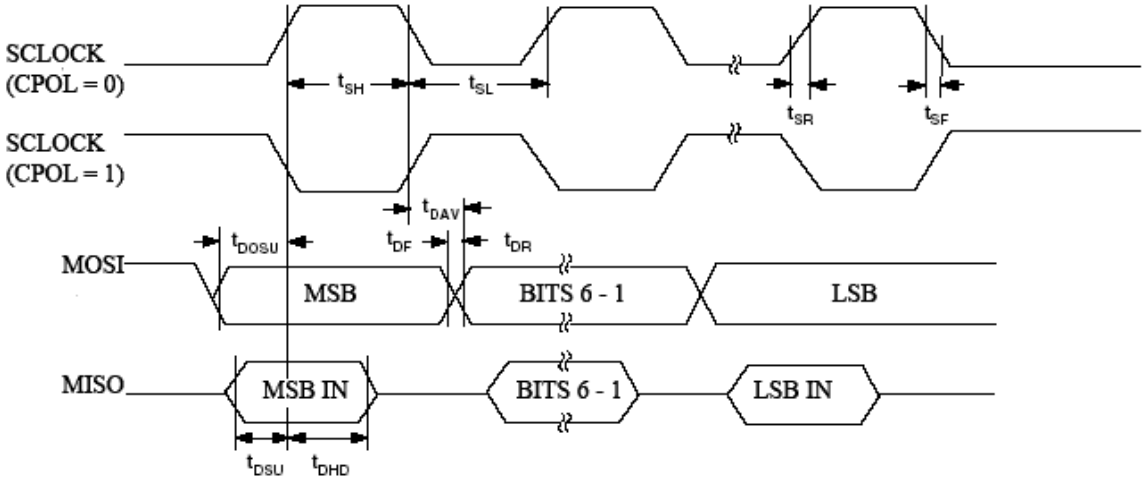
Временные параметры интерфейса SPI в режиме Master (CPHA=0)

Параметр	Мин	Тип	Макс	Единица
ВРЕМЕННЫЕ ПАРАМЕТРЫ РЕЖИМА MASTER SPI(CPHA=0)				
t_{SL} Длительность низкого уровня SCLOCK		630		нс
t_{SH} Длительность высокого уровня SCLOCK		630		нс
t_{DAV} Выдача выходных данных после перепада SCLOCK			50	нс
t_{DOSU} Установка выходных данных до перепада SCLOCK			150	нс
t_{DSU} Время установления выходных данных до перепада SCLOCK	100			нс
t_{DHD} Время удержания выходных данных после перепада SCLOCK	100			нс
t_{DF} Длительность спада выходных данных		10	25	нс
t_{DR} Длительность фронта выходных данных		10	25	нс
t_{SR} Длительность фронта SCLOCK		10	25	нс
t_{SF} Длительность спада SCLOCK		10	25	нс

*Характеристики приводятся для следующих условий:

а. Биты выбора тактовой частоты ядра CD2, CD1, CD0 в специальном регистре PLLCON имеют значения соответственно 0, 1, 1, т. е. установлена тактовая частота ядра 1,57 МГц.

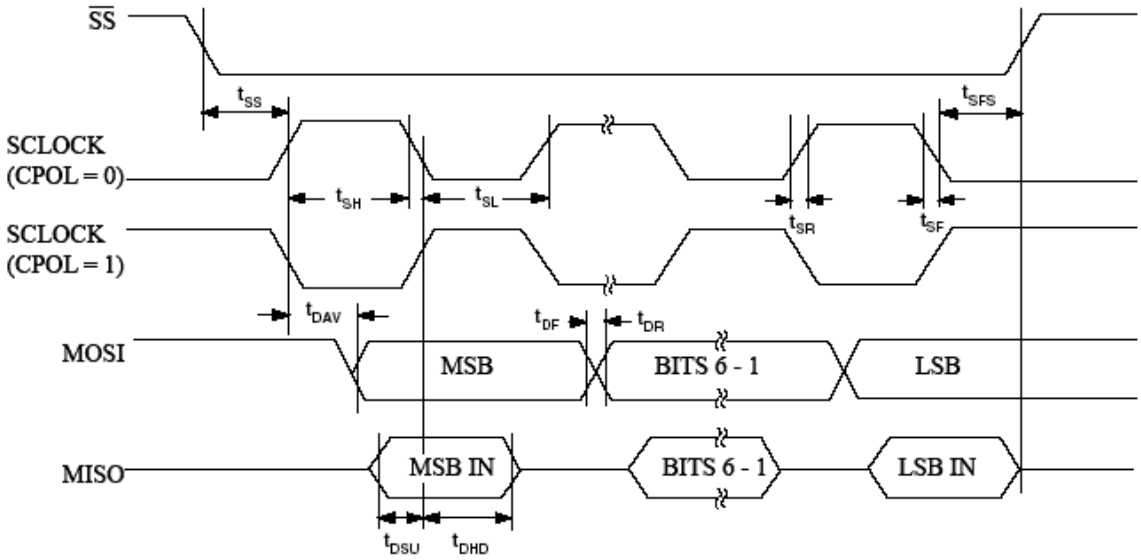
б. Биты выбора скорости SPI SPR1, SPR0 в специальном регистре SPICON имеют значения 0, 0.



Временные параметры режима Master SPI (CPHA=0)

Приложение 11
Временные параметры интерфейса SPI в режиме Slave (CPHA=1)

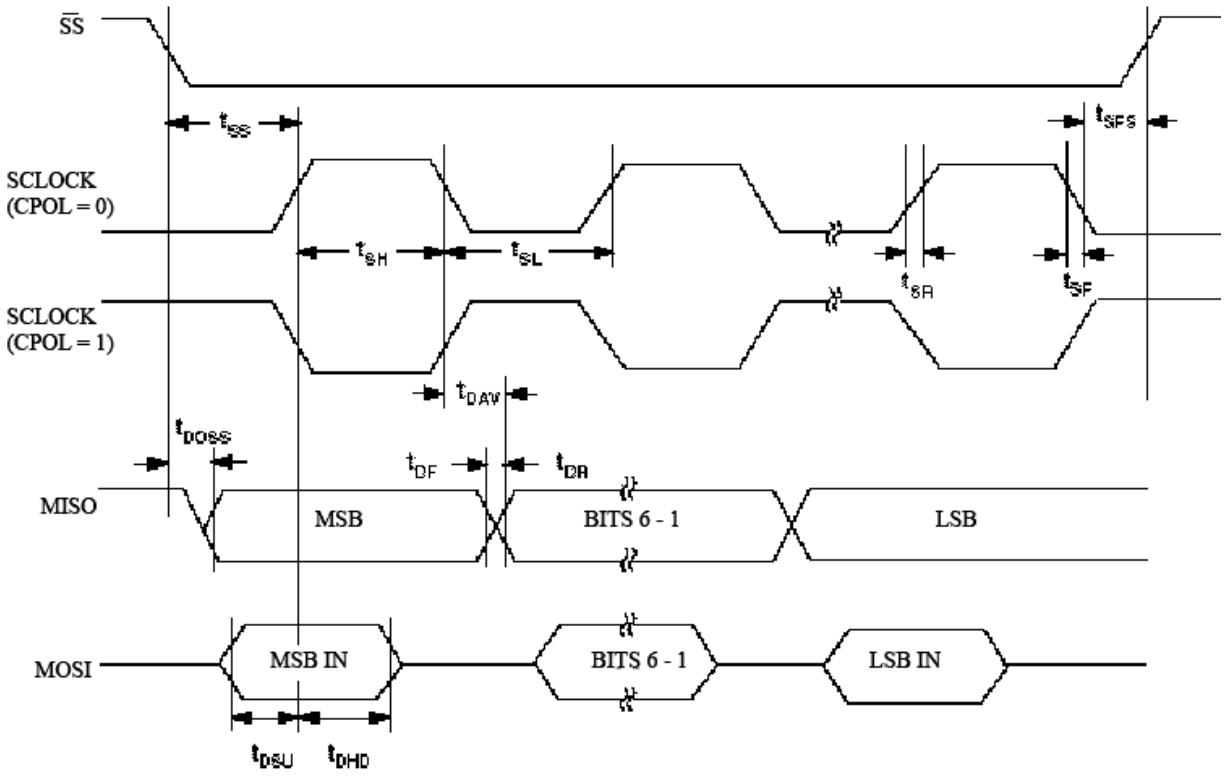
Параметр	Мин	Тип	Макс	Единица
ВРЕМЕННЫЕ ПАРАМЕТРЫ РЕЖИМА SLAVE SPI(CPHA=1)				
t_{SS} /SS до перепада SCLOCK	0			нс
t_{SL} Длительность низкого уровня SCLOCK		330		нс
t_{SH} Длительность высокого уровня SCLOCK		330		нс
t_{DAV} Выдача выходных данных после перепада SCLOCK			50	нс
t_{DSU} Время установления выходных данных до перепада SCLOCK	100			нс
t_{DHD} Время удержания выходных данных после перепада SCLOCK	100			нс
t_{DF} Длительность спада выходных данных		10	25	нс
t_{DR} Длительность фронта выходных данных		10	25	нс
t_{SR} Длительность фронта SCLOCK		10	25	нс
t_{SF} Длительность спада SCLOCK		10	25	нс
t_{SFS} Установка/SS после перепада SCLOCK	0			нс



Временные параметры режима Slave SPI (CPHA=1)

Приложение 12
Временные параметры интерфейса SPI в режиме Slave (CPHA=0)

Параметр	Мин	Тип	Макс	Единица
ВРЕМЕННЫЕ ПАРАМЕТРЫ РЕЖИМА SLAVE SPI(CPHA=0)				
t_{SS} /SS до перепада SCLOCK	0			нс
t_{SL} Длительность низкого уровня SCLOCK		330		нс
t_{SH} Длительность высокого уровня SCLOCK		330		нс
t_{DAV} Выдача выходных данных после перепада SCLOCK			50	нс
t_{DSU} Время установления выходных данных до перепада SCLOCK	100			нс
t_{DHD} Время удержания выходных данных после перепада SCLOCK	100			нс
t_{DF} Длительность спада выходных данных		10	25	нс
t_{DR} Длительность фронта выходных данных		10	25	нс
t_{SR} Длительность фронта SCLOCK		10	25	нс
t_{SF} Длительность спада SCLOCK		10	25	нс
t_{DOSS} Выдача выходных данных после перепада/SS			20	нс
t_{SFS} Установка/SS после перепада SCLOCK	0			нс



Временные параметры режима Slave SPI (CPHA=0)

Приложение 13
Список команд ассемблера Asm51

Группа команд передачи данных

Название команды	Мнемокод	КОП	Т	Б	Ц	Операция
Пересылка в аккумулятор из регистра (n = 0 – 7)	MOV A, Rn	11101rrr	1	1	1	(A) = (Rn)
Пересылка в аккумулятор прямоадресуемого байта	MOV A, ad	11100101	3	2	1	(A) = (ad)
Пересылка в аккумулятор байта из РДП (i = 0, 1)	MOV A, @Ri	1110011i	1	1	1	(A) = ((Ri))
Загрузка в аккумулятор константы	MOV A, #d	01110100	2	2	1	(A) = #d
Пересылка в регистр из аккумулятора	MOV Rn, A	11111rrr	1	1	1	(Rn) = (A)
Пересылка в регистр прямоадресуемого байта	MOV Rn, ad	10101rrr	3	2	2	(Rn) = (ad)
Загрузка в регистр константы	MOV Rn, #d	01111rrr	2	2	1	(Rn) = #d
Пересылка по прямому адресу аккумулятора	MOV ad, A	11110101	3	2	1	(ad) = (A)
Пересылка по прямому адресу регистра	MOV ad, Rn	10001rrr	3	2	2	(ad) = (Rn)
Пересылка прямоадресуемого байта по прямому адресу	MOV add, ads	10000101	9	3	2	(add) = (ads)
Пересылка байта из РДП по прямому адресу	MOV ad, @Ri	1000011i	3	2	2	(ad) = ((Ri))
Пересылка по прямому адресу константы	MOV ad, #d	01110101	7	3	2	(ad) = #d
Пересылка в РДП из аккумулятора	MOV @Ri, A	1111011i	1	1	1	((Ri)) = (A)
Пересылка в РДП прямоадресуемого байта	MOV @Ri, ad	0110011i	3	2	2	((Ri)) = (ad)
Пересылка в РДП константы	MOV @Ri, #d	0111011i	2	2	1	((Ri)) = #d
Загрузка указателя данных	MOV DPTR, #d16	10010000	13	3	2	(DPTR) = #d16
Пересылка в аккумулятор байта из ПП	MOVC A, @A + DPTR	10010011	1	1	2	(A) = ((A) + (DPTR))
Пересылка в аккумулятор байта из ПП	MOVC A, @A + PC	10000011	1	1	2	(PC) = (PC) + 1 (A) = ((A) + (PC))
Пересылка в аккумулятор байта из ВПД	MOVX A, @Ri	1110001i	1	1	2	(A) = ((Ri))
Пересылка в аккумулятор байта из расширенной ВПД	MOVX A, @DPTR	11100000	1	1	2	(A) = ((DPTR))
Пересылка в ВПД из аккумулятора	MOVX @Ri, A	1111001i	1	1	2	((Ri)) = (A)

Название команды	Мнемокод	КОП	Т	Б	Ц	Операция
Пересылка в расширенную ВПД из аккумулятора	MOVX @DPTR, A	11110000	1	1	2	((DPTR)) = (A)
Загрузка в стек	PUSH ad	11000000	3	2	2	(SP) = (SP) + 1 ((SP)) = (ad)
Извлечение из стека	POP ad	11010000	3	2	2	(ad) = (SP) (SP) = (SP) - 1
Обмен аккумулятора с регистром	XCH A, Rn	11001rrr	1	1	1	(A) <-> (Rn)
Обмен аккумулятора с прямоадресуемым байтом	XCH A, ad	11000101	3	2	1	(A) <-> (ad)
Обмен аккумулятора с байтом из РДП	XCH A, @Ri	1100011i	1	1	1	(A) <-> ((Ri))
Обмен младшей тетрады аккумулятора с младшей тетрадой байта РДП	XCHD A, @Ri	1101011i	1	1	1	(A_{0-3}) <-> $((Ri)_{0-3})$

Группа команд арифметических операций

Название команды	Мнемокод	КОП	Т	Б	Ц	Операция
Сложение аккумулятора с регистром (n = 0 - 7)	ADD A, Rn	00101rrr	1	1	1	= (A) + (Rn)
Сложение аккумулятора с прямоадресуемым байтом	ADD A, ad	00100101	3	2	1	(A) = (A) + (ad)
Сложение аккумулятора с байтом из РПД (i = 0, 1)	ADD A, @Ri	0010011i	1	1	1	(A) = (A) + ((Ri))
Сложение аккумулятора с константой	ADD A, #d	00100100	2	2	1	(A) = (A) + #d
Сложение аккумулятора с регистром и переносом	ADDC A, Rn	00111rrr	1	1	1	(A) = (A) + (Rn) + (C)
Сложение аккумулятора с прямоадресуемым байтом и переносом	ADDC A, ad	00110101	3	2	1	(A) = (A) + (ad) + (C)
Сложение аккумулятора с байтом из РПД и переносом	ADDC A, @Ri	0011011i	1	1	1	(A) = (A) + ((Ri)) + (C)
Сложение аккумулятора с константой и переносом	ADDC A, #d	00110100	2	2	1	(A) = (A) + #d + (C)
Десятичная коррекция аккумулятора	DA A	11010100	1	1	1	Если $(A_{0-3}) > 9 \vee ((AC) = 1)$, то $(A_{0-3}) = (A_{0-3}) + 6$, затем если $(A_{4-7}) > 9 \vee ((C) = 1)$, то $(A_{4-7}) = (A_{4-7}) + 6$
Вычитание из аккумулятора регистра и заема	SUBB A, Rn	10011rrr	1	1	1	(A) = (A) - (C) - (Rn)
Вычитание из аккумулятора прямоадресуемого байта и заема	SUBB A, ad	10010101	3	2	1	(A) = (A) - (C) - ((ad))
Вычитание из аккумулятора байта РПД и заема	SUBB A, @Ri	1001011i	1	1	1	(A) = (A) - (C) - ((Ri))

Название команды	Мнемокод	КОП	Т	Б	Ц	Операция
Вычитание из аккумулятора константы и заема	SUBB A, #d	10010100	2	2	1	(A) = (A) - (C) - #d
Инкремент аккумулятора	INC A	00000100	1	1	1	(A) = (A) + 1
Инкремент регистра	INC Rn	00001rrr	1	1	1	(Rn) = (Rn) + 1
Инкремент прямоадресуемого байта	INC ad	00000101	3	2	1	(ad) = (ad) + 1
Инкремент байта в РПД	INC @Ri	0000011i	1	1	1	((Ri)) = ((Ri)) +1
Инкремент указателя данных	INC DPTR	10100011	1	1	2	(DPTR) = (DPTR) + 1
Декремент аккумулятора	DEC A	00010100	1	1	1	(A) = (A) – 1
Декремент регистра	DEC Rn	00011rrr	1	1	1	(Rn) = (Rn) – 1
Декремент прямоадресуемого байта	DEC ad	00010101	3	2	1	(ad) = (ad) – 1
Декремент байта в РПД	DEC @Ri	0001011i	1	1	1	((Ri)) = ((Ri)) -1
Умножение аккумуллятора на регистр B	MUL AB	10100100	1	1	4	(B)(A) = (A)*(B)
Деление аккумулятора на регистр B	DIV AB	10000100	1	1	4	(A).(B) = (A)/(B)

Группа команд логических операций

Название команды	Мнемокод	КОП	Т	Б	Ц	Операция
Логическое И аккумулятора и регистра	ANL A, Rn	01011rrr	1	1	1	(A) = (A) /\ (Rn)
Логическое И аккумулятора и прямоадресуемого байта	ANL A, ad	01010101	3	2	1	(A) = (A) /\ (ad)
Логическое И аккумулятора и байта из РПД	ANL A, @Ri	0101011i	1	1	1	(A) = (A) /\ ((Ri))
Логическое И аккумулятора и константы	ANL A, #d	01010100	2	2	1	(A) = (A) /\ #d
Логическое И прямоадресуемого байта и аккумулятора	ANL ad, A	01010010	3	2	1	(ad) = (ad) /\ (A)
Логическое И прямоадресуемого байта и константы	ANL ad, #d	01010011	7	3	2	(ad) = (ad) /\ #d
Логическое ИЛИ аккумулятора и регистра	ORL A, Rn	01001rrr	1	1	1	(A) = (A) \/ (Rn)
Логическое ИЛИ аккумулятора и прямоадресуемого байта	ORL A, ad	01000101	3	2	1	(A) = (A) \/ (ad)
Логическое ИЛИ аккумулятора и байта из РПД	ORL A, @Ri	0100011i	1	1	1	(A) = (A) \/ ((Ri))
Логическое ИЛИ аккумулятора и константы	ORL A, #d	01000100	2	2	1	(A) = (A) \/ #d
Логическое ИЛИ прямоадресуемого байта и аккумулятора	ORL ad, A	01000010	3	2	1	(ad) = (ad) \/ (A)
Логическое ИЛИ прямоадресуемого байта и константы	ORL ad, #d	01000011	7	3	2	(ad) = (ad) \/ #d

Название команды	Мнемокод	КОП	Т	Б	Ц	Операция
Исключающее ИЛИ аккумулятора и регистра	XRL A, Rn	01101rrr	1	1	1	(A) = (A) ⩛ (Rn)
Исключающее ИЛИ аккумулятора и прямоадресуемого байта	XRL A, ad	01100101	3	2	1	(A) = (A) ⩛ (ad)
Исключающее ИЛИ аккумулятора и байта из РПД	XRL A, @Ri	0110011i	1	1	1	(A) = (A) ⩛ ((Ri))
Исключающее ИЛИ аккумулятора и константы	XRL A, #d	01100100	2	2	1	(A) = (A) ⩛ #d
Исключающее ИЛИ прямоадресуемого байта и аккумулятора	XRL ad, A	01100010	3	2	1	(ad) = (ad) ⩛ (A)
Исключающее ИЛИ прямоадресуемого байта и константы	XRL ad, #d	01100011	7	3	2	(ad) = (ad) ⩛ #d
Сброс аккумулятора	CLR A	11100100	1	1	1	(A) = 0
Инверсия аккумулятора	CPL A	11110100	1	1	1	(A) = (/A)
Сдвиг аккумулятора влево циклически	RL A	00100011	1	1	1	$(A_{n+1}) = (A_n)$, n = 0 - 6, $(A_0) = (A_7)$
Сдвиг аккумулятора влево через перенос	RLC A	00110011	1	1	1	$(A_{n+1}) = (A_n)$, n = 0 - 6, (A_0) = (C), (C) = (A_7)
Сдвиг аккумулятора вправо циклически	RR A	00000011	1	1	1	$(A_n) = (A_{n+1})$, n = 0 - 6, $(A_7) = (A_0)$
Сдвиг аккумулятора вправо через перенос	RRC A	00010011	1	1	1	$(A_n) = (A_{n+1})$, n = 0 - 6, (A_7) = (C), (C) = (A_0)
Обмен местами тетрад в аккумуляторе	SWAP A	11000100	1	1	1	(A_{0-3}) <-> (A_{4-7})

Группа команд операций с битами

Название команды	Мнемокод	КОП	Т	Б	Ц	Операция
Сброс переноса	CLR C	11000011	1	1	1	(C) = 0
Сброс бита	CLR bit	11000010	4	2	1	(b) = 0
Установка переноса	SETB C	11010011	1	1	1	(C) = 1
Установка бита	SETB bit	11010010	4	2	1	(b) = 1
Инверсия переноса	CPL C	10110011	1	1	1	(C) = (/C)
Инверсия бита	CPL bit	10110010	4	2	1	(b) = (/b)
Логическое И бита и переноса	ANL C, bit	10000010	4	2	2	(C) = (C) /\ (b)
Логическое И инверсии бита и переноса	ANL C, /bit	10110000	4	2	2	(C) = (C) /\ (/b)
Логическое ИЛИ бита и переноса	ORL C, bit	01110010	4	2	2	(C) = (C) \/ (b)
Логическое ИЛИ инверсии бита и переноса	ORL C, /bit	10100000	4	2	2	(C) = (C) \/ (/b)
Пересылка бита в перенос	MOV C, bit	10100010	4	2	1	(C) = (b)
Пересылка переноса в бит	MOV bit, C	10010010	4	2	2	(b) = (C)

Группа команд передачи управления

Название команды	Мнемокод	КОП	Т	Б	Ц	Операция
Длинный переход в полном объеме памяти в программ	LJMP ad16	00000010	12	3	2	(PC) = ad16
Абсолютный переход внутри страницы в 2 Кбайта	AJMP ad11	$A_{10}a_9a_8$00001	6	2	2	(PC) = (PC) + 2 (PC_{0-10}) = ad11
Короткий относительный переход внутри страницы в 256 байт	SJMP rel	10000000	5	2	2	(PC) = (PC) + 2 (PC) = (PC) + rel
Косвенный относительный переход	JMP @A+DPTR	01110011	1	1	2	(PC) = (A) + (DPTR)
Переход, если аккумулятор равен нулю	JZ rel	01100000	5	2	2	(PC) = (PC) + 2, если (A) = 0, то (PC) = (PC) + rel
Переход, если аккумулятор не равен нулю	JNZ rel	01110000	5	2	2	(PC) = (PC) + 2, если (A) <> 0, то (PC) = (PC) + rel
Переход, если перенос равен единице	JC rel	01000000	5	2	2	(PC) = (PC) + 2, если (C) = 1, то (PC) = (PC) + rel
Переход, если перенос равен нулю	JNC rel	01010000	5	2	2	(PC) = (PC) + 2, если (C) = 0, то (PC) = (PC) + rel
Переход, если бит равен единице	JB bit, rel	00100000	11	3	2	(PC) = (PC) + 3, если (b) = 1, то (PC) = (PC) + rel
Переход, если бит равен нулю	JNB bit, rel	00110000	11	3	2	(PC) = (PC) + 3, если (b) = 0, то (PC) = (PC) + rel
Переход, если бит установлен, с последующим сбросом бита	JBC bit, rel	00010000	11	3	2	(PC) = (PC) + 3, если (b) = 1, то (b) = 0 и (PC) = (PC) + rel
Декремент регистра и переход, если не нуль	DJNZ Rn, rel	11011rrr	5	2	2	(PC) = (PC) + 2, (Rn) = (Rn) -1, если (Rn) <> 0, то (PC) = (PC) + rel
Декремент прямоадресуемого байта и переход, если не нуль	DJNZ ad, rel	11010101	8	3	2	(PC) = (PC) + 2, (ad) = (ad) -1, если (ad) <> 0, то (PC) = (PC) + rel
Сравнение аккумулятора с прямоадресуемым байтом и переход, если не равно	CJNE A, ad, rel	10110101	8	3	2	(PC) = (PC) + 3, если (A) <> (ad), то (PC) = (PC) + rel, если (A) < (ad), то (C) = 1, иначе (C) = 0

Название команды	Мнемокод	КОП	Т	Б	Ц	Операция
Сравнение аккумулятора с константой и переход, если не равно	CJNE A, #d, rel	10110100	10	3	2	(PC) = (PC) + 3, если (A) <> #d, то (PC) = (PC) + rel, если (A) < #d, то (C) = 1, иначе (C) = 0
Сравнение регистра с константой и переход, если не равно	CJNE Rn, #d, rel	10111rrr	10	3	2	(PC) = (PC) + 3, если (Rn) <> #d, то (PC) = (PC) + rel, если (Rn) < #d, то (C) = 1, иначе (C) = 0
Сравнение байта в РПД с константой и переход, если не равно	CJNE @Ri, #d, rel	1011011i	10	3	2	(PC) = (PC) + 3, если ((Ri)) <> #d, то (PC) = (PC) + rel, если ((Ri)) < #d, то (C) = 1, иначе (C) = 0
Длинный вызов подпрограммы	LCALL ad16	00010010	12	3	2	(PC) = (PC) + 3, (SP) = (SP) + 1, ((SP)) = (PC_{0-7}), (SP) = (SP) + 1, ((SP)) = (PC_{8-15}), (PC) = ad16
Абсолютный вызов подпрограммы в пределах страницы в 2 Кбайта	ACALL ad11	$a_{10}a_9a_8$10001	6	2	2	(PC) = (PC) + 2, (SP) = (SP) + 1, ((SP)) = (PC_{0-7}), (SP) = (SP) + 1, ((SP)) = (PC_{8-15}), (PC_{0-10}) = ad11
Возврат из подпрограммы	RET	00100010	1	1	2	(PC_{8-15}) = ((SP)), (SP) = (SP) - 1, (PC_{0-7}) = ((SP)), (SP) = (SP) - 1
Возврат из подпрограммы обработки прерывания	RETI	00110010	1	1	2	(PC_{8-15}) = ((SP)), (SP) = (SP) - 1, (PC_{0-7}) = ((SP)), (SP) = (SP) - 1
Холостая команда	NOP	00000000	1	1	1	(PC) = (PC) + 1

Библиографический список

1. ANALOG DEVICES MicroConverter, Dual-Channel 16-/24-Bit ADCs with Embedded FLASH MCU. REV. A.
2. ADUC824 GETSTARTED GUIDE a tutorial guide for use with the ADuC824 QuickStart Development System v1.1.
3. MicroConverter Technical Note - uC009. Addressing 16MB of External Data Memory. Ver 0.1 November 2001.
4. MicroConverter Technical Note - uC004. Understanding the Serial Download Protocol. Ver 2.1 Sept 2001.
5. Голов А. А. Общие положения и введение в логику работы шины I2C. Практические рекомендации / А. А. Голов. – КТЦ-МК, 1997.
6. MicroConverter Technical Note uC001. MicroConverter I2C. Compatible Interface. Ver 2.1 November 2002.
7. ADUC824 EVALUATION BOARD REFERENCE GUIDE Version 1.0 Analog Devices Inc., MicroConverter
8. 8051 CROSS ASSEMBLER USER'S MANUAL. MetaLink Corporation. Chandler, Arizona
9. Фрунзе А. В. Микроконтроллеры? Это же просто! /А. В.Фрунзе. – Т. 3. – М.: ООО «ИД СКИМЕН», 2003.
10. Библиотека электронных компонентов. Выпуск 8: Жидкокристаллические индикаторы фирмы DATA International. – М.: ДОДЭКА, 1999.
11. Алфавитно-цифровые индицирующие ЖК-модули на основе контроллера HD44780. (c) 2000 КТЦ-МК.
12. Интегральные микросхемы: Микросхемы для аналого-цифрового преобразования и средств мультимедиа. Выпуск 1 – М.: ДОДЭКА, 1996.
13. Atmel 4-megabit 2,5-volt Only or 2,7-volt Only DataFlach AT45DB041B. Rev. 1938D-1/02.
14. Microchip 24AA64/24LC64 64K I2C ™ CMOS Serial EEPROM. 1999 Microchip Technology Inc. DS21189C.
15. Atmel 8-bit Microcontroller with 2K Bytes Flash AT89C2051. Rev. 0368E-02/00.

Учебное издание

РЕДЬКИН Павел Павлович
ВИНОГРАДОВ Александр Борисович

МИКРОКОНВЕРТОРЫ ФИРМЫ ANALOG DEVICES В МИКРОПРОЦЕССОРНЫХ ПРИБОРНЫХ КОМПЛЕКСАХ

Учебное пособие

Редактор *О. А. Фирсова*

Подписано в печать 30.12.2005. Формат 60×84/16.
Бумага офсетная. Печать трафаретная.
Усл. печ. л. 18,25. Уч.-изд. л. 18,00.
Тираж 100 экз. Заказ

Ульяновский государственный технический университет,
432027, Ульяновск, Сев. Венец, 32.
Типография УлГТУ, 432027, Ульяновск, Сев. Венец, 32.