

Учебный стенд SDK-1.1

Руководство пользователя

/Версия 1.0.11/

Версия	Дата	Описание сделанных изменений	Автор
1.0.3	22.11.01	Исправлена таблица «Прерывания ADuC812».	Ключев А. О.
1.0.4	20.12.01	Добавлены переводы описаний E ² PROM, LCD и RTC.	Ключев А. О.
1.0.5	29.12.01	Редакторская правка документа.	Маковецкая Н. А.
1.0.6	29.12.01	Редакторская правка документа.	Ключев А. О.
1.0.7	08.01.02	Редакторская правка документа.	Маковецкая Н. А.
1.0.8	13.05.02	Редакторская правка документа.	Лукичев А.Н.
1.0.9	25.04.05	Исправление регистров MAX	Ключев А. О.
1.0.10	26.04.05	Описание различий регистровых моделей разных конфигураций MAX.	Ковязин Р. Р.
1.0.11	14.03.06	Добавлено описание ревизии R4 печатной платы.	Ковязин Р.Р.

ОГЛАВЛЕНИЕ

Введение.....	4
Архитектура стенда SDK-1.1	5
Структура аппаратной части	5
Микроконтроллер ADuC812BS	6
Внешняя E ² PROM	6
Матричная клавиатура AK1604A-WWB.....	6
ЖКИ WH1602B-YGK-CP	6
Часы реального времени PCF8583.....	6
Инструкция по эксплуатации.....	8
Общий вид стенда SDK-1.1	8
Разъемы стенда и назначение выводов	9
Распределение памяти в SDK-1.1	13
Карта портов ввода-вывода	14
Регистры ПЛИС.....	14
Регистр клавиатуры KB	15
Регистр шины данных ЖКИ DATA_IND	15
Регистр данных параллельного порта EXT_LO	15
Регистр данных параллельного порта EXT_HI	16
Регистр управления ENA.....	16
Регистр управления ЖКИ C_IND	17
Регистр управления светодиодами SV	17
Доступ к регистрам ПЛИС	17
Проблемы, часто возникающие при доступе к регистрам ПЛИС	18
Система прерываний	18
Программирование и отладка	20
Описание периферийных микросхем.....	21
E ² PROM AT24C01A.....	21
Особенности.....	21
Описание	21
Конфигурации выводов	21

Описание выводов.....	23
Работа с устройствами	26
ЖКИ WH1602B-YGK-CP	32
Меры предосторожности при использовании ЖКИ	32
Общее описание.....	33
Максимально возможные значения параметров	33
Электрические характеристики	33
Оптические характеристики.....	34
Описание выходов.....	35
Схематическое изображение и блок-схема.....	36
Описание функций	37
Образы символов, хранящихся в ПЗУ	40
Таблица команд	41
Временные характеристики.....	42
Инициализация LCM	44
Информация о подсветке.....	46
Часы / календарь с ОЗУ 240x8 бит PCF8583	46
Особенности.....	46
Общее описание.....	46
Краткие характеристики	47
Блок-схема.....	47
Выводы микросхемы.....	48
Описание функций	48
Характеристики шины I ² C.....	57
Протокол передачи данных шины I ² C.....	59
Предельные величины	61
Уход за устройством	61
Характеристики постоянного тока	61
Характеристики переменного тока.....	62
Информация по использованию	63
Программное обеспечение стенда SDK-1.1	65
Резидентный загрузчик HEX202.....	65
Инструментальная система для MS-DOS (T167B).....	66
Инструментальная система для Win 9x/NT	69
Программатор Flash для ADuC812 (DL)	70
Набор средств тестирования стенда SDK-1.1	71
Простейшая программа на языке Си	72
Инструментальные средства фирмы Keil Software	73
Компилятор языка Си фирмы Keil Software	73
Расширение языка C до C51	75
Типы данных.....	75
Модификаторы памяти	76
Модели памяти	77
Указатели	77
Нетипизированные указатели	77
Память-зависимые указатели	78
Сравнение память-зависимых и нетипизированных указателей.....	78
Реентерабельные функции	78
Функции – обработчики прерываний.....	79
Передача параметров	79
Значения, возвращаемые функцией	80
Оптимальное выделение регистров.....	80

Добавление ассемблерного кода.....	80
Работа с языком PL/M-51	81
Оптимизация кода	81
Отладка программ	83
Файлы библиотек	83
Встроенные библиотеки	84
Командная строка.....	84
Пример файла листинга	85
Макроассемблер A51	86
Обзор функций	87
Конфигурация.....	87
Командная строка.....	87
Пример файла листинга	88
Загрузчик/компоновщик BL51	89
Управление доступом к данным	89
Распределение кода в памяти.....	89
Общая область памяти	90
Выполнение функций в других банках	90
Командная строка.....	90
Пример файла листинга	91
Утилиты.....	92
Конвертер объектных файлов OC51.....	92
Шестнадцатеричный конвертер OH51	93
Менеджер библиотек LIB51	93
Отладчик/симулятор Keil dScope.....	94
Интегрированная среда разработки Keil μ Vision.....	96
Операционная система реального времени RTX51	98

Введение

Учебный лабораторный комплекс SDK-1.1 предназначен для освоения студентами архитектуры и методов проектирования:

- Систем на базе микропроцессоров и однокристальных микроЭВМ;
- Встраиваемых контроллеров и систем сбора данных;
- Периферийных блоков вычислительных систем;
- Подсистем ввода-вывода встраиваемых систем.

С использованием стенда SDK-1.1 для студентов высших учебных заведений могут проводиться лабораторные работы по курсам:

- Организация ЭВМ и вычислительных систем;
- Прикладная теория цифровых автоматов;
- Системы ввода-вывода;
- Информационно-управляющие системы;
- Распределенные управляющие системы;
- Операционные системы реального времени.

Архитектура стенда SDK-1.1

Структура аппаратной части

В состав учебного стенда SDK-1.1 входят:

- Микроконтроллер ADuC812BS;
- Внешняя E²PROM объемом 256 байт;
- Клавиатура AK1604A-WWB фирмы ACCORD;
- Жидкокристаллический индикатор (ЖКИ) WH1602B-YGK-CP фирмы Winstar Display;
- Часы реального времени PCF8583;
- 128K внешней SRAM с возможностью расширения до 512K;
- Набор сигнальных светодиодов (8 шт.).

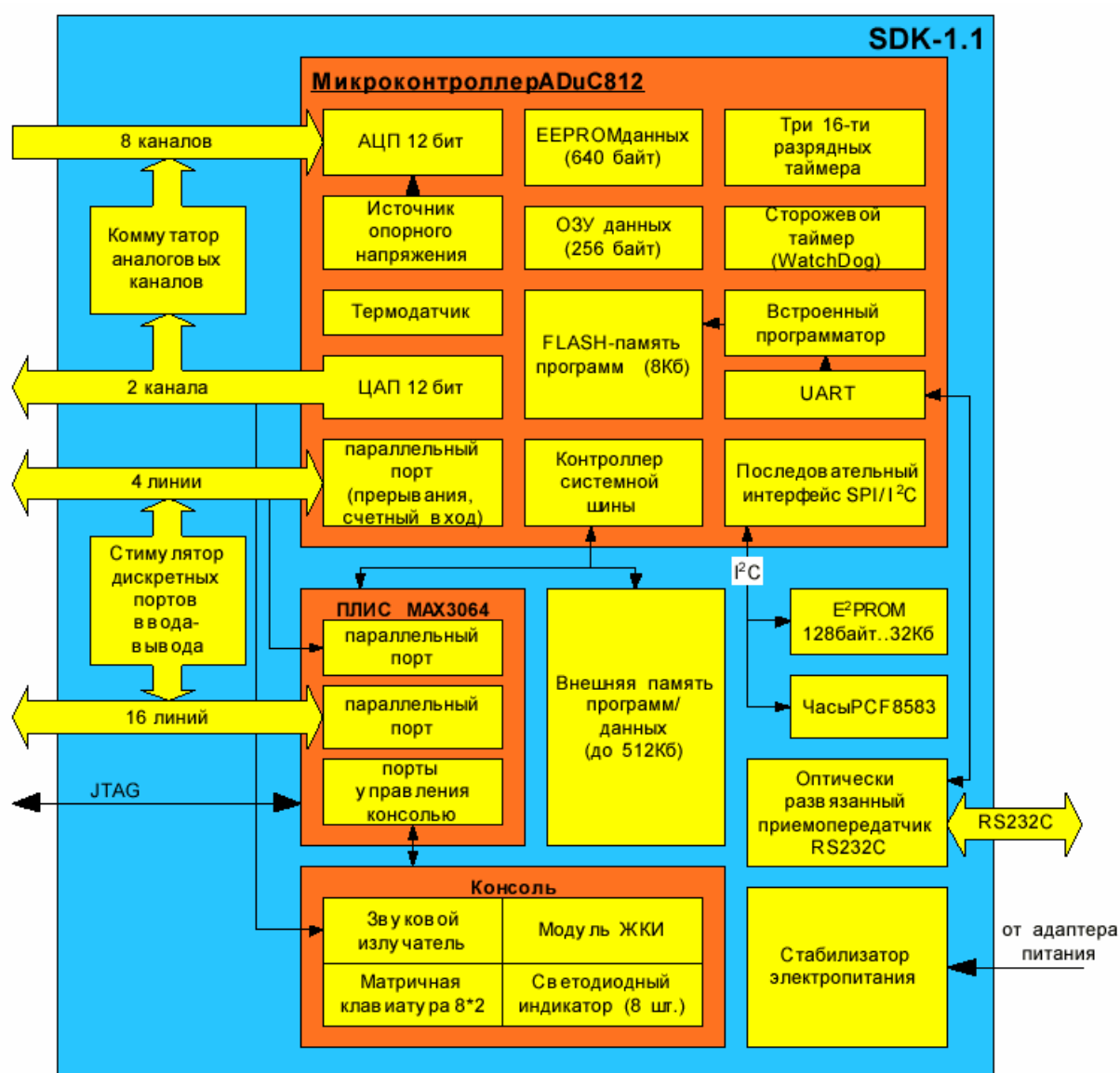


Рисунок 1. Структура аппаратной части учебного стенда SDK-1.1.

Микроконтроллер ADuC812BS

Процессор ADuC812 является клоном Intel 8051 со встроенной периферией.

Основные характеристики:

- Рабочая частота 11.0592 МГц.
- 8-канальный 12-битный АЦП со скоростью выборки 200 К/с (в режиме ПДП).
- Два 12-битных ЦАП (код-напряжение).
- Внутренний температурный датчик.
- 640 байт программируемого E²PROM со страничной организацией (256 страниц по 4 байта).
- 256 байт внутренней памяти данных.
- Адресное пространство 16 Мб.
- Режим управления питанием.
- Асинхронный последовательный ввод-вывод.
- Интерфейс I²C.
- Три 16-битных таймера/счетчика и таймер WatchDog.

Внешняя E²PROM

E²PROM – перепрограммируемое электрически стираемое постоянное запоминающее устройство. Объем памяти E²PROM, установленной в стенде SDK-1.1, составляет 128 байт (возможна установка E²PROM большего объема, до 32 Кб). Микросхема E²PROM взаимодействует с процессором посредством интерфейса I²C.

Адрес I ² C							
1	0	1	0	0	0	1	R/W

Рисунок 2

Основные характеристики:

- Возможность перезаписи до 1 млн. раз.
- Возможность побайтной и постраничной записи (в текущей конфигурации размер страницы составляет 8 байт).

Матричная клавиатура AK1604A-WWB

Клавиатура организована в виде матрицы 4x4. Доступ к колонкам и рядам организован как чтение/запись определенного байта внешней памяти (4 бита соответствуют 4 колонкам, другие 4 бита - рядам).

ЖКИ WH1602B-YGK-CP

ЖКИ работает в текстовом режиме (2 строки по 16 символов), имеет подсветку (цвет желто-зеленый). Основные характеристики:

- Габариты: 80x36x13.2 мм.
- Активная область 56.21x11.5 мм.
- Размеры точки 0.56x0.66 мм; размеры символа 2.96x5.56 мм.
- Встроенный набор 256 символов (ASCII + кириллица).
- Генератор символов с энергозависимой памятью на 8 пользовательских символов.

Часы реального времени PCF8583

PCF8583 – часы/календарь с памятью объемом 256 байт, работающие от кварцевого резонатора с частотой 32.768 кГц. Питание осуществляется ионистором (0.1 ф). Из 256 байт памяти собственно часами используются только первые 16 (8 постоянно обновляемых

регистров-защелок на установку/чтение даты/времени и 8 на будильник), остальные 240 байт доступны для хранения данных пользователя. Точность измерения времени – до сотых долей секунды. Взаимодействие с процессором осуществляется через интерфейс I²C.

Адрес I ² C							
1	0	1	0	0	0	0	R/W

Рисунок 3

Инструкция по эксплуатации

Общий вид стенда SDK-1.1

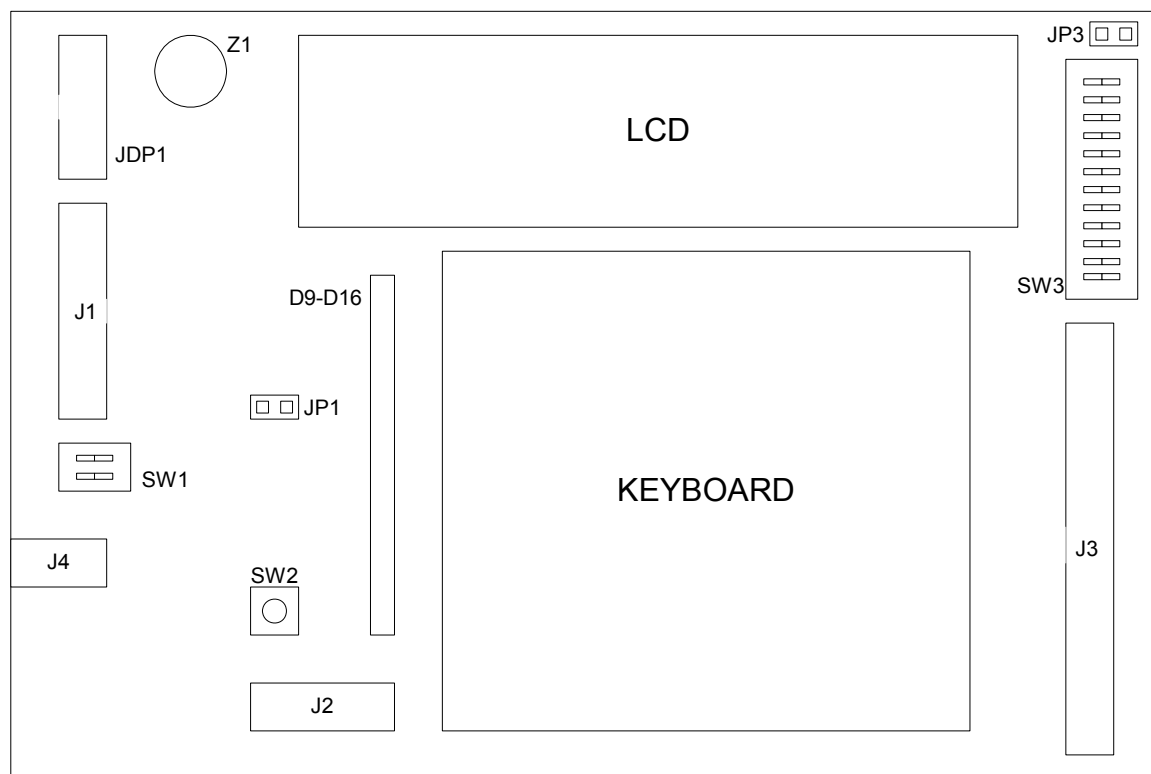


Рисунок 4. Схематическое изображение стенда SDK-1.1.

На рисунке представлено схематическое изображение лицевой панели стенда SDK-1.1. Расшифровка обозначений на схеме дана в таблице.

Таблица 1. Расшифровка обозначений на схеме лицевой панели стенда SDK-1.1.

Элемент	Описание
LCD	Жидкокристаллический индикатор WH1602B-YGK-CP (см. предыдущий раздел).
KEYBOARD	Матричная клавиатура AK1604A-WWB (см. предыдущий раздел).
Z1	Звуковой пьезокерамический излучатель.
SW2	Кнопка сброса RESET.
J4	Разъем питания стенда 10-14 В типа «JACK», полярность безразлична.
JDP1	Разъем последовательного порта стенда.
J1	Выводы каналов АЦП и ЦАП.
SW1	Переключатель, замыкающий каналы 0 и/или 1 ЦАП на входы соответствующих (0, 1) каналов АЦП.
J3	16 линий параллельного порта ПЛИС MAX и 4 линии параллельного порта P3 микроконтроллера ADuC812 (INT0/1, T0/1).
SW3	Набор переключателей, замыкающих соответствующие выводы J3 на корпус (переключение в лог. «0»).
J2	Выводы JTAG-интерфейса ПЛИС MAX.
JP1	Перемычка, замыкающая вывод PSEN микроконтроллера ADuC812 на корпус.
JP3	Разъемы подключения внешней батареи питания часов реального времени PCF8583.
D9-D16	Набор сигнальных светодиодов.

Разъемы стенда и назначение выводов

Перемычка JP1. Перемычка предназначена для замыкания вывода PSEN микроконтроллера ADuC812 через резистор 1 КОм на корпус. По сигналу RESET или при включении питания микроконтроллер ADuC812 анализирует состояние этого вывода и если он находится в лог. «0» (перемычка замкнута), то запускается встроенная в микроконтроллер процедура перезаписи внутренней Flash-памяти. (См. раздел «Программатор Flash для ADuC812».)

Разъем JP3. Разъем предназначен для подключения внешней батареи питания + 5 В часов реального времени PCF8583. Если батарея не подключена, питание часов осуществляется через ионистор емкостью 0.1 ф. Назначение выводов относительно надписей на плате представлено на рисунке.

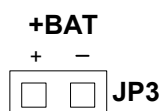


Рисунок 5. Разъем JP3: назначение выводов.

Разъем JDP1. Этот разъем предназначен для подключения кабеля асинхронного последовательного интерфейса, связывающего стенд с СОМ-портом РС. Назначение выводов представлено на рисунке.

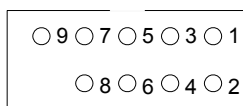


Рисунок 6. Выводы разъема JDP1.

К разъему подключается кабель, на одном конце которого находится стандартный для РС разъем DB25F, а на другом – IDC10F. Распайка кабеля представлена на рисунке.

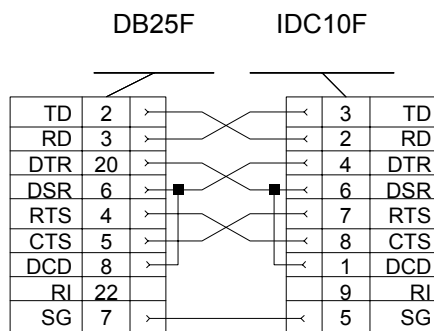


Рисунок 7. Распайка кабеля.

Разъем J2. Разъем предназначен для программирования ПЛИС MAX3064 (MAX3128) через интерфейс JTAG (IEEE1149). Нумерация выводов относительно надписи «J2» на плате показана на рисунке.

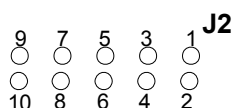


Рисунок 8. Нумерация выводов разъема J2.

Таблица 2. Выводы разъема J2.

№ вывода	Описание (JTAG)
1	TCK
2	Корпус
3	TDO
4	+ 5 V
5	TMS
6	-
7	-
8	#RESET
9	TDI
10	Корпус

При перепрограммировании ПЛИС ООО «ЛМТ» не несет ответственности за потерю работоспособности устройства.

Разъем J1. Разъем представляет собой набор входов восьмиканального АЦП и выводов двухканального ЦАП микроконтроллера ADuC812.

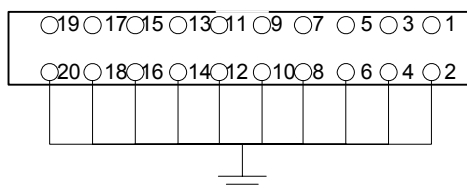


Рисунок 9. Разъем J1.

Таблица 3. Назначение выводов разъема J1.

№ вывода	Описание
1	Канал 0 ЦАП
3	Канал 1 ЦАП
5	Вход канала 0 АЦП
7	Вход канала 1 АЦП
9	Вход канала 2 АЦП
11	Вход канала 3 АЦП
13	Вход канала 4 АЦП
15	Вход канала 5 АЦП
17	Вход канала 6 АЦП
19	Вход канала 7 АЦП
Четные 2-20	Корпус

Напряжение, подаваемое на входы АЦП, делится на два при поступлении на соответствующие выводы микроконтроллера ADuC812 (ADC0-ADC7, см. документ ADuC812_a.pdf, прилагаемый к настоящему руководству).

На панели стенда смонтирован переключатель SW1, замыкающий соответственно канал 0 ЦАП на вход канала 0 АЦП (перемычка «1» в положении ON) и канал 1 ЦАП на вход канала 1 АЦП (перемычка «2» в положении ON).

Внимание! Замыкание каналов ЦАП на корпус при ненулевом напряжении на них может привести к выходу микроконтроллера ADuC812 из строя.

Разъем J3. Разъем представляет собой выводы параллельного порта ПЛИС MAX 3064 (MAX3128) и 4 линии порта P3 микроконтроллера ADuC812. Нумерация выводов представлена на рисунке.

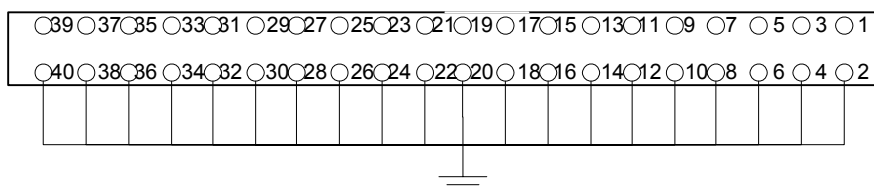


Рисунок 10. Нумерация выводов разъема J3.

Таблица 4. Назначение выводов разъема J3.

№ вывода	Описание
1	Вход INT0 ADuC812.
3	Вход INT1 ADuC812.
5	Вход T0 ADuC812.
7	Вход T1 ADuC812.
9	Вывод линии 0 параллельного порта ПЛИС МАХ.
11	Вывод линии 1 параллельного порта ПЛИС МАХ.
13	Вывод линии 2 параллельного порта ПЛИС МАХ.
15	Вывод линии 3 параллельного порта ПЛИС МАХ.
17	Вывод линии 4 параллельного порта ПЛИС МАХ.
19	Вывод линии 5 параллельного порта ПЛИС МАХ.
21	Вывод линии 6 параллельного порта ПЛИС МАХ.
23	Вывод линии 7 параллельного порта ПЛИС МАХ.
25	Вывод линии 8 параллельного порта ПЛИС МАХ.
27	Вывод линии 9 параллельного порта ПЛИС МАХ.
29	Вывод линии 10 параллельного порта ПЛИС МАХ.
31	Вывод линии 11 параллельного порта ПЛИС МАХ.
33	Вывод линии 12 параллельного порта ПЛИС МАХ.
35	Вывод линии 13 параллельного порта ПЛИС МАХ.
37	Вывод линии 14 параллельного порта ПЛИС МАХ.
39	Вывод линии 15 параллельного порта ПЛИС МАХ.
Четные 2-40	Корпус.

Поскольку напряжение питания ПЛИС МАХ3064 (МАХ3128) составляет 3.3 В, уровнем лог. «1» на линиях параллельного порта ПЛИС считается напряжение около 3 В. Уровнем лог. «1» на линиях INT0/1, T0/1 микроконтроллера ADuC812 (выводы 1, 3, 5, 7) считается напряжение около 5 В.

Внимание! Прямое замыкание линий параллельного порта ПЛИС или порта P3 ADuC812 на корпус при ненулевом напряжении на них может привести к выходу соответствующих микросхем из строя.

Для принудительного «обнуления» линий INT0/1, T0/1, а также линий 0-7 параллельного порта ПЛИС в схему введен набор переключателей SW3, замыкающих соответствующие линии через резисторы 100 Ом на корпус. Для того, чтобы принудительно «обнулить» соответствующую линию, необходимо установить соответствующий переключатель в положение «ON». Соответствие номеров переключателей и линий приведено в таблице.

Таблица 5

№ переключателя	Линия
1	Вход INT0 ADuC812 (P3.2).
2	Вход INT1 ADuC812 (P3.3).
3	Вход T0 ADuC812 (P3.4).
4	Вход T1 ADuC812 (P3.5).
5	Линия 0 параллельного порта ПЛИС МАХ.
6	Линия 1 параллельного порта ПЛИС МАХ.
7	Линия 2 параллельного порта ПЛИС МАХ.
8	Линия 3 параллельного порта ПЛИС МАХ.
9	Линия 4 параллельного порта ПЛИС МАХ.
10	Линия 5 параллельного порта ПЛИС МАХ.
11	Линия 6 параллельного порта ПЛИС МАХ.
12	Линия 7 параллельного порта ПЛИС МАХ.

Распределение памяти в SDK-1.1

Память в SDK-1.1 распределяется следующим образом:

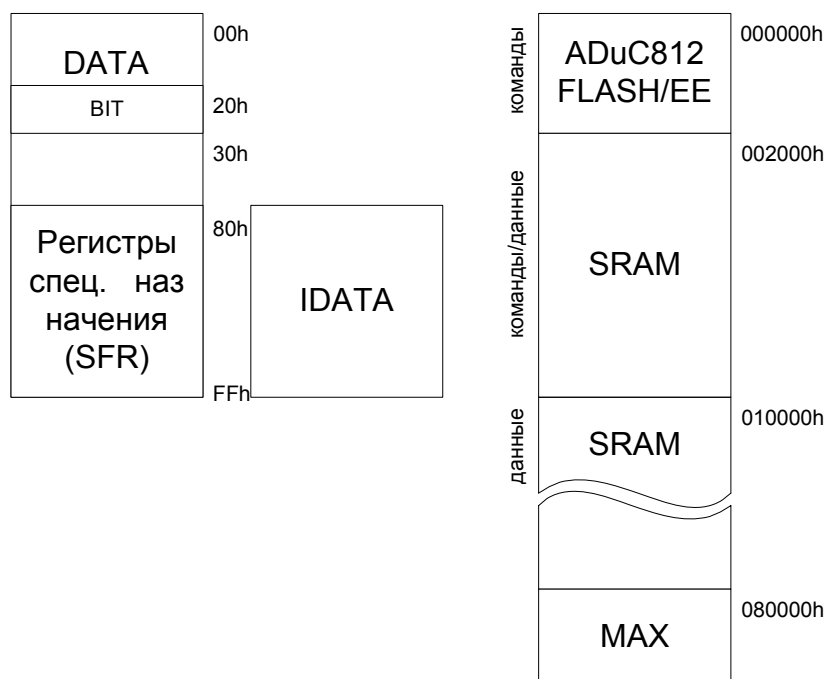


Рисунок 11. Карта памяти SDK-1.1.

Стандартная для архитектуры 8051 структура внутренней памяти представлена четырьмя банками по 8 регистров общего назначения (диапазоны адресов 00h-07h, 08h-0Fh, 10h-17h, 18h-1Fh), битовым сегментом (20h-2Fh), свободным участком 30h-7Fh, областью размещения SFR (регистров специального назначения) 80h-FFh, доступной при прямой адресации, и свободной областью 80h-FFh, доступной при косвенной адресации.

Внешняя память SDK-1.1 разбита на следующие области: AduC812 Flash/EE, SRAM, MAX.

ADuC812 Flash/EE. Это область, в которой располагается таблица векторов прерываний (см. раздел «Система прерываний») и резидентный загрузчик файлов в формате HEX в память SRAM (см. раздел «Резидентный загрузчик HEX202»).

SRAM. Статическая память SRAM в SDK-1.1 имеет страничную организацию (максимум 8 страниц по 64 К) и условно разделяется на две области. Первая занимает младшие 64 Кбайт (страница 0) и доступна для выборки команд микроконтроллером ADuC812. Таким образом, программы могут располагаться только в этих младших 64 К адресного пространства¹. Остальные страницы доступны только для размещения данных. Для адресации ячейки памяти определенной страницы необходимо записать номер страницы в регистр специального назначения DPP ADuC812 (адрес 84h, см. руководство по ADuC812).

MAX. В младших адресах восьмой страницы адресного пространства (080000h-080007h) располагается 8 ячеек-регистров ПЛИС MAX8064 (MAX8128). Эта область предназначена для взаимодействия с периферийными устройствами стенда (см. раздел «Карта портов ввода-вывода»).

¹ За вычетом 8 К Flash-памяти ADuC812, которая отображается в самые младшие адреса (0000h-1FFFh), а также зарезервированного резидентным загрузчиком HEX202 участка (F000h-FFFFh, см. соответствующий раздел с описанием HEX202). Фактически для размещения программ доступно 52 К статической памяти.

Карта портов ввода-вывода

В стенде SDK-1.1 ввод-вывод данных осуществляется с помощью портов микроконтроллера и микросхемы ПЛИС, которая имеет 8 регистров, отображаемых во внешнее адресное пространство процессора.

Таблица 6. Порты ввода-вывода микроконтроллера.

Порт	Назначение
P0.7-P0.0	Шина адреса/данных AD(7-0) системного интерфейса.
P1.7-P1.0	Аналоговый вход, линии которого мультиплексируются с линиями 7-0 АЦП.
P2.7-P2.0	Адресная шина системного интерфейса A(15-8).
P3.0	RxD – входные данные приемопередатчика UART.
P3.1	TxD – выходные данные приемопередатчика UART.
P3.2	#INT0 – сигнал внешнего прерывания 0, активный уровень – лог. «0».
P3.3	#INT1 – сигнал внешнего прерывания 1, активный уровень – лог. «0».
P3.4	Счетный вход таймера-счетчика T0, активный уровень – лог. «0».
P3.5	Счетный вход таймера-счетчика T1, активный уровень – лог. «0».
P3.6	#WR – сигнал записи во внешнюю память XRAM.
P3.7	#RD – сигнал чтения из внешней памяти XRAM.

Регистры ПЛИС

Существует несколько ревизий печатной платы SDK-1.1. Данный документ соответствует всем ревизиям. Объем соответствия указан в таблице.

Таблица 7. Ревизии печатной платы SDK-1.1.

Ревизия	Конфигурация ПЛИС	Особенности
R1	Упрощенная	
R2	Полная	
R3	Полная	Не подключены сигналы SND0 и SND1.
R4	Полная	Не подключены сигналы SND0 и SND1.

В ревизии R1 печатной платы стенда SDK-1.1 используется упрощенная конфигурация регистров микросхемы ПЛИС. В руководстве представлена полная версия регистров ПЛИС. Различия между полной и упрощенной конфигурациями указывается индивидуально для каждого регистра ПЛИС.

Таблица 8. Перечень регистров ПЛИС.

Адрес	Регистр	Доступ	Назначение
080000H	KB	R/W	Регистр клавиатуры.
080001H	DATA_IND	R/W	Регистр шины данных ЖКИ.
080002H	EXT_LO	R/W	Регистр данных параллельного порта (разряды 0..7).
080003H	EXT_HI	R/W	Регистр данных параллельного порта (разряды 8..15).
080004H	ENA	W	Регистр управления портами ввода-вывода, звуком, сигналом INT0 и прерыванием от клавиатуры.
080006H	C_IND	W	Регистр управления ЖКИ.
080007H	SV	W	Регистр управления светодиодами.

Регистр клавиатуры KB

Адрес 080000H. Значение после сброса xxxx1111B (полная конфигурация) и xxxx0000B (упрощенная конфигурация).

7	6	5	4	3	2	1	0
R	R	R	R	R/W	R/W	R/W	R/W
ROW				COL			

Рисунок 12. Регистр клавиатуры KB.

Таблица 9. Назначение битов регистра KB.

Биты	Поле	Описание
0..3	COL	Поле предназначено для сканирования клавиатуры (колонки матрицы). Сканирование производится посредством записи логического «0» в один из разрядов поля. В полной конфигурации ПЛИС из этой тетрады считывается записанное ранее в нее значение. В упрощенной конфигурации всегда возвращается «0».
4..7	ROW	Поле предназначено для считывания данных с клавиатурной матрицы (строки). Если ни одна из кнопок в строке не нажата, все биты поля ROW содержат логические «1». Если кнопка нажата и на ее колонку подан логический «0», то в поле ROW также появится логический «0».

Данный регистр можно использовать для определения типа конфигурации. Если при старте программы записать в тетраду COL регистра KB ненулевое значение, а затем его считать, то в полной конфигурации будет считано записанное значение, а в упрощенной — нули.

Регистр шины данных ЖКИ DATA_IND

Адрес 080001H. Значение после сброса 00000000B.

7	6	5	4	3	2	1	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
D7	D6	D5	D4	D3	D2	D1	D0

Рисунок 13. Регистр шины данных ЖКИ DATA_IND.

Таблица 10. Назначение битов регистра DATA_IND.

Биты	Поле	Описание
0..7	D0..D7	Регистр DATA_IND позволяет устанавливать данные на шине данных ЖКИ и считывать их оттуда. Для организации взаимодействия с ЖКИ (формирования временных диаграмм чтения и записи) необходимо использование регистра C_IND.

Регистр данных параллельного порта EXT_LO

Адрес 080002H. Значение после сброса 00000000B.

7	6	5	4	3	2	1	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
D7	D6	D5	D4	D3	D2	D1	D0

Рисунок 14. Регистр данных параллельного порта EXT_LO.

Таблица 11. Назначение битов регистра EXT_LO.

Биты	Поле	Описание
0..7	D0..D7	Регистр EXT_LO позволяет считывать и записывать биты 0..7 параллельного порта. Для того, чтобы данные из регистра попали на выход, необходимо установить бит EN_LO в логическую «1» (см. регистр ENA). Для чтения данных необходимо установить этот бит в логический «0».

Регистр данных параллельного порта EXT_HI

Адрес 080003H. Значение после сброса 0000000B.

7	6	5	4	3	2	1	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
D7	D6	D5	D4	D3	D2	D1	D0

Рисунок 15. Регистр данных параллельного порта EXT_HI.

Таблица 12. Назначение битов регистра EXT_HI.

Биты	Поле	Описание
0..7	D0..D7	Регистр EXT_HI позволяет считывать и записывать биты 8..15 параллельного порта. Для того, чтобы данные из регистра попали на выход, необходимо установить бит EN_LO или EN_HI в логическую «1» (см. регистр ENA). Для чтения данных необходимо установить этот бит в логический «0».

Регистр управления ENA

Адрес 080004H. Значение после сброса x0100000B.

7	6	5	4	3	2	1	0
-	W	W	W	W	W	W	W
-	KB	INT0	SND2	SND1	SND0	EN_HI	EN_LO

Рисунок 16. Регистр управления ENA.

Таблица 13. Назначение битов регистра ENA.

Биты	Поле	Описание
0	EN_LO	В полной конфигурации бит EN_LO нужен для управления младшими 8 разрядами (биты 0..7) 16-разрядного порта ввода-вывода. Если записать в EN_LO логический «0», то порт ввода-вывода переводится в Z-состояние и появляется возможность чтения данных из EXT_LO. При записи в данный бит логической «1» порт переключается на вывод и данные, записанные в регистр EXT_LO, попадают на выход порта ввода-вывода. В упрощенной конфигурации этот бит управляет всеми 16 разрядами порта ввода-вывода. Если записать в EN_LO логический «0», то весь порт ввода-вывода переводится в Z-состояние и появляется возможность чтения данных из регистров EXT_LO и EXT_HI. При записи в данный бит логической «1» порт переключается на вывод и данные, записанные в регистры EXT_LO и EXT_HI, попадают на выход порта ввода-вывода.
1	EN_HI	В полной конфигурации бит EN_HI нужен для управления старшими 8 разрядами (биты 8..15) 16-разрядного порта ввода-вывода. Если записать в EN_HI логический «0», то порт ввода-вывода переводится в Z-состояние и появляется возможность чтения данных из EXT_HI. При записи в данный бит логической «1» порт переключается на вывод и данные, записанные в регистр EXT_HI, попадают на выход порта ввода-вывода. В упрощенной конфигурации бит EN_HI не влияет на функционирование порта ввода-вывода. Все управление портом производится битом EN_LO.
2..4	SND0-SND2	Выход звукового ЦАП. Задаёт уровень напряжения на динамике. Позволяет формировать звуковые сигналы различной тональности и громкости.
5	INT0	При записи логического «0» в этот бит на вход INT0 ADuC812 также попадает логический «0». Бит можно использовать для формирования внешнего прерывания для микроконтроллера.
6	KB	В полной конфигурации при записи логического «0» прерывание от клавиатуры запрещается. Если бит установлен в «1», то прерывание от клавиатуры разрешено. В упрощенной конфигурации бит KB всегда равен нулю, т.е. прерывание клавиатуры запрещено.

Регистр управления ЖКИ C_IND

Адрес 080006H. Значение после сброса xxxxx010B.

7	6	5	4	3	2	1	0
-	-	-	-	-	W	W	W
-	-	-	-	-	RS	RW	E

Рисунок 17. Регистр управления ЖКИ C_IND.

Таблица 14. Назначение битов регистра C_IND.

Биты	Поле	Описание
0	E	Бит управления входом «E» ЖКИ. Наличие положительного импульса на входе «E» позволяет зафиксировать данные на шине ЖКИ (данные, сигналы RW и RS к этому моменту должны быть уже установлены).
1	RW	Бит переключения шины данных ЖКИ на чтение или запись. 0 – запись, 1 – чтение.
2	RS	Бит переключения режимов команды/данные ЖКИ. 1 – данные, 0 – команды.

Регистр управления светодиодами SV

Адрес 080007H. Значение после сброса 00000000B.

7	6	5	4	3	2	1	0
W	W	W	W	W	W	W	W
D7	D6	D5	D4	D3	D2	D1	D0

Рисунок 18. Регистр управления светодиодами SV.

Таблица 15. Назначение битов регистра SV.

Биты	Поле	Описание
0..7	D0..D7	Биты управления светодиодами. Подача логической «1» зажигает светодиоды.

Доступ к регистрам ПЛИС

Для доступа к регистрам ПЛИС нужно переключить страничный регистр DPP на 8 страницу памяти. Адреса регистров внутри страницы находятся в диапазоне от 0 до 7.

Доступ к регистрам возможен через указатель: `unsigned char xdata *regnum`

Ниже приведен пример функций для доступа к регистрам ПЛИС.

```
#define MAXBASE 8 // Страница памяти, в которую отображаются регистры ПЛИС

/*
WriteMAX - Запись байта в регистр ПЛИС
regnum   - адрес регистра ПЛИС
val      - записываемое значение
результат - нет
*/

void WriteMax (unsigned char xdata *regnum, unsigned char val)
{
    unsigned char oldDPP = DPP;

    DPP = MAXBASE;
    *regnum = val;
    DPP = oldDPP;
}
```

```

/*
ReadMAX - Чтение байта из регистра ПЛИС
regnum  - адрес регистра ПЛИС
результат - прочитанное значение
*/

unsigned char ReadMax(unsigned char xdata *regnum)
{
    unsigned char oldDPP = DPP;
    unsigned char val = 0;

    DPP = MAXBASE;
    val = *regnum;
    DPP = oldDPP;
    return val;
}

```

Проблемы, часто возникающие при доступе к регистрам ПЛИС

Необходимо помнить, что при переключении страниц становятся недоступными все данные, размещенные в странице 0.

Для того, чтобы избежать проблем со страничным регистром DPP, нужно использовать специальные функции для доступа к ПЛИС, которые перед началом работы с регистрами ПЛИС будут запоминать старое значение страничного регистра, а по окончании работы возвращать его обратно.

Нужно следить, чтобы передаваемые в регистры ПЛИС значения хранились во внутренней памяти микроконтроллера (DATA, IDATA). Убедиться, что передаваемая информация не содержится во внешней памяти контроллера (XDATA), достаточно легко: так как для доступа к внешней памяти в микроконтроллерах семейства C51 используется регистр DPTR, нужно просто просмотреть листинг программы и убедиться в том, что для доступа к переменным компилятор не использует DPTR.

Система прерываний

Микроконтроллер ADuC812 обеспечивает восемь источников и два уровня приоритета прерываний. Соответствующий определенному прерыванию приоритет можно установить в регистре специального назначения IP (адрес B8h, см. руководство по ADuC812).

Таблица 16. Прерывания ADuC812.

Прерывание	Наименование	Адрес вектора	Приоритет
PSMI	Источник питания ADuC812.	43H	1
IE0	Внешнее прерывание INT0.	03H	2
ADCI	Конец преобразования АЦП.	33H	3
TF0	Переполнение таймера 0.	0BH	4
IE1	Внешнее прерывание INT1.	13H	5
TF1	Переполнение таймера 1.	1BH	6
I2CI/ISPI	Прерывание последовательного интерфейса (I ² C, SPI).	3BH	7
RI/TI	Прерывание UART.	23H	8
TF2/EXF2	Переполнение таймера 2.	2BH	9

Прерывания ADuC812 имеют вектора в диапазоне 0003h-0043h, которые попадают в область младших адресов памяти программ. Это пространство соответствует 8Кб (0000h-2000h) Flash-памяти. Следовательно, пользователь, не имеющий возможности записи во Flash-память, не может подставить свои процедуры обработки прерываний (точнее, команды перехода к процедурам) по адресам, соответствующим векторам прерываний.

Проблема использования прерываний в пользовательских программах решается следующим образом:

1. По адресам (0003h-0043h) векторов прерываний во FLASH-памяти SDK-1.1 располагаются команды переходов на вектора пользовательской таблицы, размещенной в адресах 2003h-2043h.
2. По адресам векторов пользовательской таблицы пользователем указываются команды переходов на процедуры обработки прерываний.

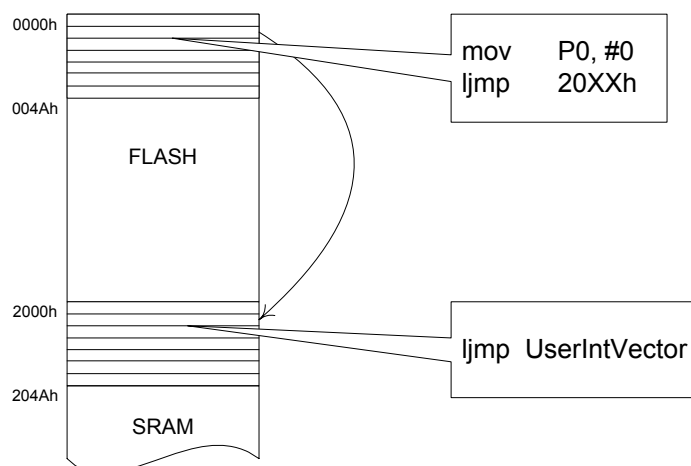


Рисунок 19. Использование прерываний в SDK-1.1².

Приведем пример помещения собственного вектора в пользовательскую таблицу. Пусть требуется осуществить обработку прерываний от таймера 0 (прерывание 0Bh). В программу на языке Си (компилятор фирмы Keil Software) можно вставить следующий код:

```
void T0_ISR(void) interrupt 1 // Обработчик прерывания от таймера 0
{
    // Действия, выполняемые обработчиком
}

void SetVector(unsigned char xdata * Address, void * Vector)
// Функция, устанавливающая вектор прерывания Vector по адресу Address
{
    unsigned short xdata * TmpVector; // Временная переменная

    *Address = 0x02; // Первым байтом по указанному адресу записывается код команды
                    // передачи управления ljmp, равный 02h

    TmpVector = (unsigned short xdata *) (Address+1);
    *TmpVector = (unsigned short) Vector; // Далее записывается адрес перехода Vector

    // Таким образом, по адресу Address теперь располагается инструкция ljmp Vector
}

void main(void)
{
    /*...*/
    SetVector(0x200B, (void *) T0_ISR); // Установка вектора в пользовательской таблице
    ET0=1; EA=1; // Разрешение прерываний от таймера 0
    /*...*/
}
```

² В таблице векторов, находящейся во FLASH, перед переходом в пользовательскую таблицу в порт P0 записывается значение 0. Это связано с аппаратной ошибкой в процессоре ADuC812, которая заключается в некорректной выборке команд при передаче управления из младших 8К памяти команд в старшие адреса. Ошибка устраняется путем обнуления p0 непосредственно перед передачей управления. Рекомендуем обратиться к документу ADuC812 Errata Sheet на редакцию чипа со штампом даты больше 9933.

Программирование и отладка

Для программирования стенда может использоваться любой транслятор ассемблера или C для ядра 8051, например, пакет μ Vision (Keil Software). До начала программирования на языке C рекомендуется внимательно ознакомиться с документацией по используемому компилятору, так как компиляторы для микроконтроллеров имеют нестандартные расширения.

Основные этапы программирования стенда:

- Подготовка программы в текстовом редакторе или среде программирования.
- Транслирование исходного текста и получение загрузочного HEX-модуля программы.
- Подготовка и загрузка HEX-модуля в стенд через интерфейс RS232C с помощью поставляемых инструментальных систем. Под подготовкой понимается добавление в конец модуля строчки со стартовым адресом программы, то есть адреса, по которому передается управление после загрузки в стенд.
- Прием и обработка HEX-модуля резидентным загрузчиком HEX202, передача управления загруженной программе.

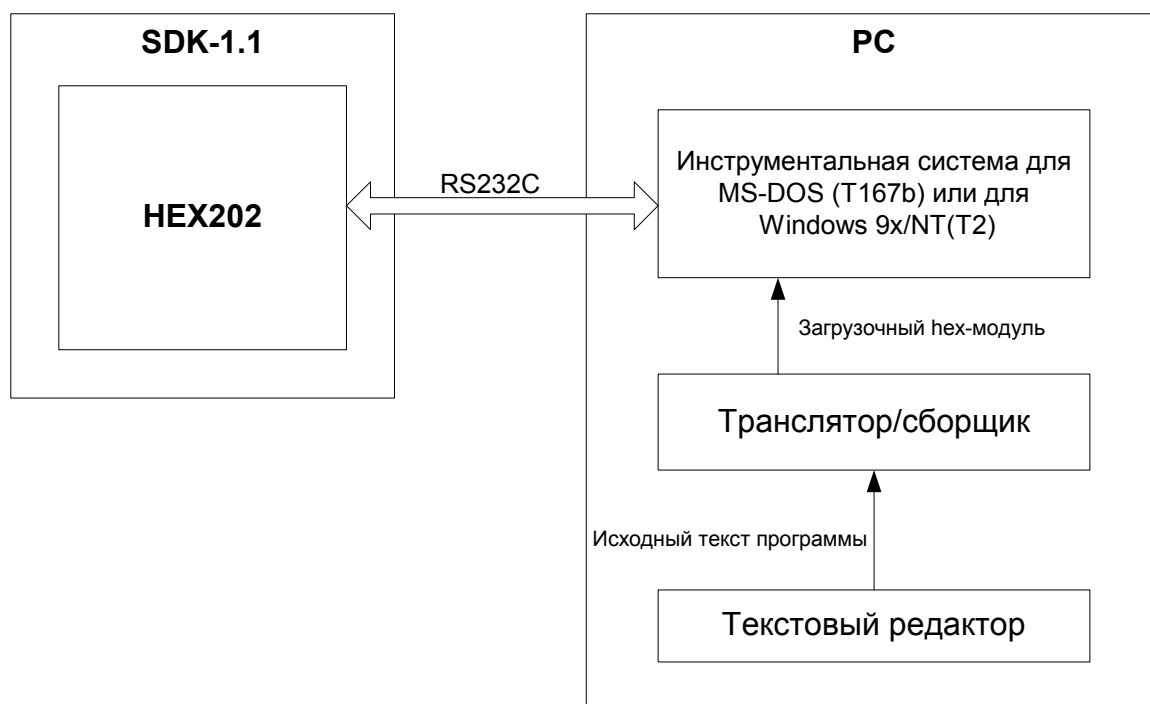


Рисунок 20. Этапы программирования стенда SDK-1.1.

В соответствующих разделах главы «Программное обеспечение стенда SDK-1.1» и в главе «Инструментальные средства фирмы Keil Software» дано описание основных инструментальных средств, участвующих в каждом этапе программирования стенда.

Описание периферийных микросхем³

E²PROM AT24C01A

Особенности

- Внутреннее ПЗУ 128x8 (1 К), 256x8 (2 К), 512x8 (4 К), 1024x8 (8 К) или 2048x8 (16 К).
- Двухпроводной последовательный интерфейс.
- Двухнаправленный протокол передачи данных.
- Совместимость по частоте 100 кГц (1.8 V, 2.5 V, 2.7 V) и 400 кГц (5 V).
- Вывод защиты записи, обеспечивающий аппаратную защиту данных.
- Поддержка страничной записи в 8-байтном (1 К, 2 К), и 16-байтном (4 К, 8 К, 16 К) режимах.
- Поддержка неполной страничной записи.
- Самосинхронизирующийся цикл записи (максимум – 10 мс).
- Высокая надежность:
 - продолжительность работы 1 миллион циклов;
 - сохранение данных в памяти в течение 100 лет.
- Автоматическая градуировка и возможность работы в широком диапазоне температур.
- 8-штыревой и 14-штыревой модуль JEDEC SOIC, 8-штыревой модуль PDIP.

Описание

AT24C01A / 02 / 04 / 08 / 16 содержит 1024 / 2048 / 4096 / 8192 / 16384 бит последовательной памяти E²PROM, состоящей из 128 / 256 / 512 / 1024 / 2048-битных слов, которая может быть перезаписана с помощью электрических сигналов и считана программным путем. Данное устройство предназначено для применения в промышленных и коммерческих областях, где важным условием является низкое энергопотребление и малое напряжение питания.

AT24C01A / 02 / 04 / 08 / 16 включает 8-штыревой модуль PDIP, 8- и 14-штыревой модуль SOIC. Доступ осуществляется через 2-проводной последовательный интерфейс. Кроме того, разработано несколько вариантов микросхем данного семейства: 5.0V (4.5V - 5.5V), 2.7V (2.7V - 5.5V), 2.5V (2.5V - 5.5V) и 1.8V (1.8V - 5.5V).

Конфигурации выводов

Таблица 17. Назначение выводов микросхемы.

Вывод	Функция
A0 – A2	Адресные входы.
SDA	Последовательная передача данных.
SCL	Линия синхронизации.
WP	Защита от записи.
NC	Не используется.

³ Перевод документации к E²PROM AT24C01A, ЖКИ WH1602B-YGK-CP и часам / календарю PCF8583 выполнен Е. Клейменовою и А. Ильенковой.

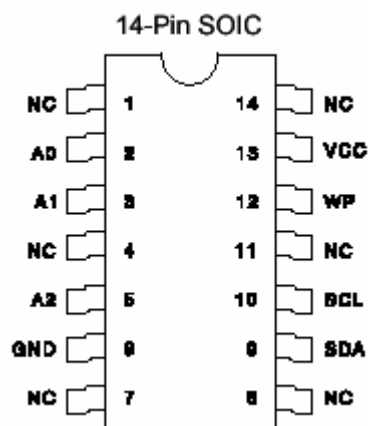


Рисунок 21

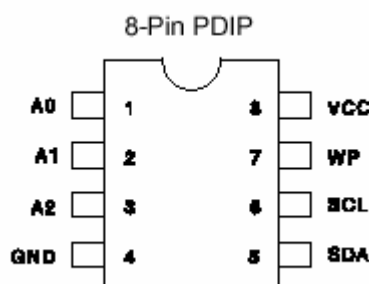


Рисунок 22

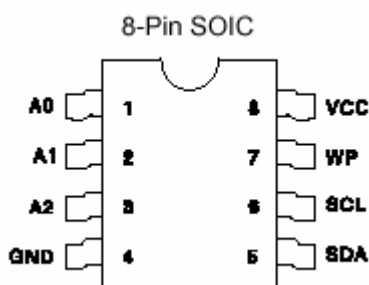


Рисунок 23

Таблица 18. Номинальные значения⁴.

Диапазон рабочих температур	-55°C - +125°C
Условия хранения	-65°C - +150°C
Напряжение на ножках с учетом заземления	-0.1V - +7.0V
Максимальное рабочее напряжение	6.25V
Постоянный ток выхода	5.0 mA

⁴ Примечание. Использование нагрузок, превышающих номинальные, может привести к ухудшению качества работы устройства. В таблице даны стандартные условия использования и рассчитанные на них нагрузки. За использование в иных условиях разработчик ответственности не несет. Если устройство в течение длительного периода времени испытывает максимальные нагрузки, это может отразиться на надежности его работы.

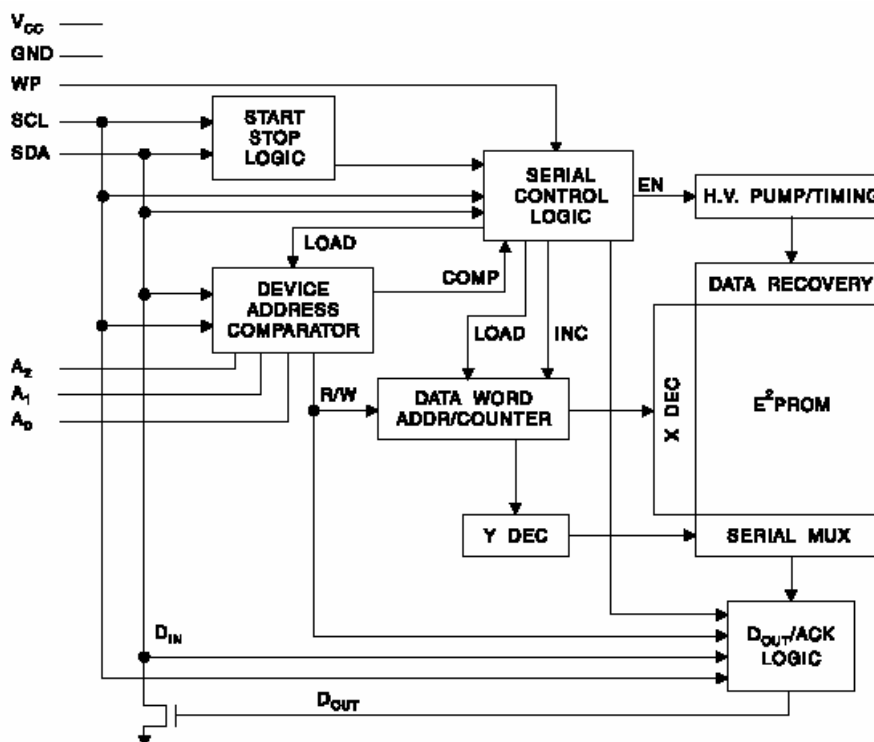


Рисунок 24

Обозначения на схеме:

- Start-stop logic – логика «пуск-останов».
- Device Address Comparator – блок сравнения адреса.
- Serial Control Logic – последовательная логика управления.
- Data Word Addr/Counter – адрес/счетчик слова данных.
- H.V. Pump/Timing – генератор подкачки заряда / тактирование.
- Data Recovery – восстановление данных.
- Serial Mux – мультиплексор последовательной передачи.
- Dout/ACK Logic – логика подтверждения передачи сигнала.

Описание выводов

SERIAL CLOCK (SCL) – линия синхронизации.

Вход SCL используется при передаче в E²PROM (положительный фронт) и отправке данных на любое внешнее устройство (отрицательный фронт).

SERIAL DATA (SDA) – линия последовательной передачи данных.

SDA – вывод для двунаправленной последовательной передачи данных. Это вывод со свободным стоком, к нему можно подключать любое количество открытых коллекторов или коллекторов со свободным стоком.

Адреса страниц/устройства (A2, A1, A0).

Выводы A2, A1 и A0 – это адресные входы устройств, разработанные для микросхем AT24C01A и AT24C02. К одной шине может быть подключено до 8 1K/2K - устройств (адресация устройств более подробно обсуждается в разделе "Адресация устройств").

Микросхема AT24C04 использует для фиксированной адресации два вывода A2 и A1, что позволяет подключить к одной шине до четырех таких микросхем. Вывод A0 не используется.

Микросхема AT24C08 использует для фиксированной адресации только вывод A2, что позволяет подключить к одной шине до двух таких микросхем. Выводы A0 и A1 не используются.

Микросхема AT24C16 не использует адресные выводы. К одной шине можно подключить только одно устройство. Выводы A0, A1 и A2 не используются.

Защита от записи (WP).

Схемы семейства AT24C01A / 02 / 04 / 16 имеют вывод защиты от записи, с помощью которого можно защитить аппаратные данные. Этот вывод используется для обычных операций чтения/записи в случае, если он подключен к заземлению (GND). Когда на этот вывод подается напряжение, свойство защиты от записи проявляется так, как показано в таблице ниже.

Таблица 19. Защита от записи в схемах семейства AT24C01A/02/04/16.

Состояние вывода	Защита массива данных				
	24C01A	24C02	24C04	24C08	24C16
Напряжение	Полностью (1К)	Полностью (2К)	Полностью (4К)	Обычные операции чтения/записи	Верхняя половина массива (8К)
Земля	Обычные операции чтения/записи				

Организация памяти

AT24C01A, 1К последовательная E²PROM. Внутренняя память, состоящая из 128 однобайтовых страниц общим объемом в 1 К, для произвольного доступа к которой требуются 7-битные адреса.

AT24C02, 2К последовательная E²PROM. Внутренняя память, состоящая из 256 однобайтовых страниц общим объемом в 2К, для произвольного доступа к которой требуются 8-битные адреса.

AT24C04, 4К последовательная E²PROM. Внутренняя память объемом в 4К, состоящая из 256 страниц по 2 байта каждая. Для произвольного доступа к данным требуются 9-битные адреса.

AT24C08, 8К последовательная E²PROM. Внутренняя память объемом в 8К, состоящая из 4 блоков. Каждый блок содержит 256 4-байтных страниц. Для произвольного доступа к данным требуется 10-битная адресация.

AT24C16, 16К последовательная E²PROM. Внутренняя память объемом в 16К, состоящая из 8 блоков. Каждый блок содержит 256 8-байтных страниц. Для произвольного доступа к данным необходима 11-битная адресация.

Емкостные характеристики выводов⁵

Характеристики описаны для следующих условий: $t = 25^{\circ}\text{C}$, $f = 1.0 \text{ MHz}$, $V_{\text{CC}} = +1.8\text{V}$.

Таблица 20. Емкостные характеристики выводов.

Обозначение	Описание	Максимум	Единицы	Условие
CI/O	Емкость ввода/вывода (SDA)	8	pF	VI/O = 0V
CIN	Входная емкость (A0, A1, A2, SCL)	6	pF	VIN = 0V

Электрические характеристики

Характеристики описаны для рекомендуемых условий эксплуатации:

$t_{\text{AI}} = -40^{\circ}\text{C} \dots +85^{\circ}\text{C}$, $V_{\text{CC}} = +1.8 \text{ V} \dots +5.5 \text{ V}$, $t_{\text{AC}} = 0^{\circ}\text{C} \dots +70^{\circ}\text{C}$, $V_{\text{CC}} = +1.8 \text{ V} \dots +5.5 \text{ V}$ (если не указаны другие значения).

⁵ Данные характеристики носят описательный характер и не были полностью протестированы.

Таблица 21. Электрические характеристики.

Обозн-е	Параметр	Условия тестирования	Мин.	Обычн.	Макс.	Ед.
VCC1	Напряжение		1.8		5.5	V
VCC2	Напряжение		2.5		5.5	V
VCC3	Напряжение		2.7		5.5	V
VCC4	Напряжение		4.5		5.5	V
ICC	Ток питания, VCC = 5.0V	Чтение на частоте 100 кГц		0.4	1.0	мА
ICC	Ток питания, VCC = 5.0V	Запись на частоте 100 кГц		2.0	3.0	мА
ISB1	Ток холостого хода, VCC = 1.8V	VIN = VCC или VSS		0.6	3.0	мкА
ISB2	Ток холостого хода, VCC = 2.5V	VIN = VCC или VSS		1.4	4.0	мкА
ISB3	Ток холостого хода, VCC = 2.7V	VIN = VCC или VSS		1.6	4.0	мкА
ISB4	Ток холостого хода, VCC = 5.0V	VIN = VCC или VSS		8.0	18.0	мкА
ILI	Входной ток утечки	VIN = VCC или VSS		0.1	3.0	мкА
ILO	Выходной ток утечки	VOUT = VCC или VSS		0.05	3.0	мкА
VIL	Входное напряжение низкого уровня ⁶		-1.0		VCC x 0.3	V
VIH	Входное напряжение высокого уровня ⁷		VCC x 0.7		VCC + 0.5	V
VOL2	Выходное напряжение низкого уровня, VCC = 3.0V	IOL = 2.1 мА			0.4	V
VOL1	Выходное напряжение низкого уровня, VCC = 1.8V	IOL = 0.15 мА			0.2	V

⁶ VIL min и VIH max представляют собой справочные значения и не проверялись на практике.

⁷ VIL min и VIH max представляют собой справочные значения и не проверялись на практике.

Таблица 22. Значения переменного тока.

Обозн-е	Параметр	2.7-, 2.5-, 1.8-В		5.0-В		Ед.
		Мин.	Макс.	Мин.	Макс.	
fSCL	Частота синхронизации SCL		100		400	кГц
tLOW	Синхронизация импульса низкого уровня	4.7		1.2		мкс
tHIGH	Синхронизация импульса высокого уровня	4.0		0.6		мкс
tI	Время подавления шумов ⁸		100		50	нс
tAA	Низкий уровень синхронизации, в течение которого возможна передача	0.1	4.5	0.1	0.9	мкс
tBUF	Время перед очередной передачей, в течение которого шина должна быть свободна	4.7		1.2		мкс
tHD.STA	Старт-сигнал удержания	4.0		0.6		мкс
tSU.STA	Старт-сигнал установки	4.7		0.6		мкс
tHD.DAT	Передача сигнала	0		0		мкс
tSU.DAT	Начало передачи сигнала	200		100		нс
tR	Время нарастания сигнала ⁹		1.0		0.3	мкс
tF	Время спада сигнала ¹⁰		300		300	нс
tSU.STO	Время установки стоп-сигнала	4.7		0.6		мкс
tDH	Удержание сигнала	100		50		нс
tWR	Цикл записи		10		10	мс

Работа с устройствами

Синхронизация и передача данных:

Вывод SDA обычно соединяется с внешним устройством. Данные могут быть переданы по SDA только тогда, когда на SCL подан сигнал низкого уровня (см. таблицу). Если на SCL подан сигнал высокого уровня, то изменение уровня на SDA будет означать выдачу сигналов старт-стопных состояний, как описано ниже.

Старт-состояние:

Изменение уровня сигнала с высокого на низкий на SDA при условии сигнала высокого уровня на линии SCL означает, что линия SDA находится в старт-состоянии, что должно предшествовать выполнению любой другой команды (см. обозначения Start, Stop на временной диаграмме).

Стоп-состояние:

Изменение уровня сигнала с низкого на высокий на линии SDA при условии наличия на линии SCL сигнала высокого уровня свидетельствует о том, что линия SDA находится в стоп-состоянии. После чтения данных и получения команды "стоп" E²PROM перейдет в режим резервного питания (см. обозначения Start, Stop на временной диаграмме).

Подтверждение приема:

Обмен адресами и данными с E²PROM производится 8-битовыми словами. После получения каждого слова E²PROM выдает "0". Это происходит в процессе передачи 9 импульсов синхронизации.

Режим ожидания:

Режим ожидания для схем семейства AT24C01A / 02 / 04 / 08 / 16 доступен: после включения питания и получения стопового бита, а также после завершения любых внутренних операций.

Временная диаграмма работы шины

(SCL – линия синхронизации, SDA – линия последовательной передачи данных)

⁸ Данные характеристики не были полностью оттестированы.

⁹ Данные характеристики не были полностью оттестированы.

¹⁰ Данные характеристики не были полностью оттестированы.

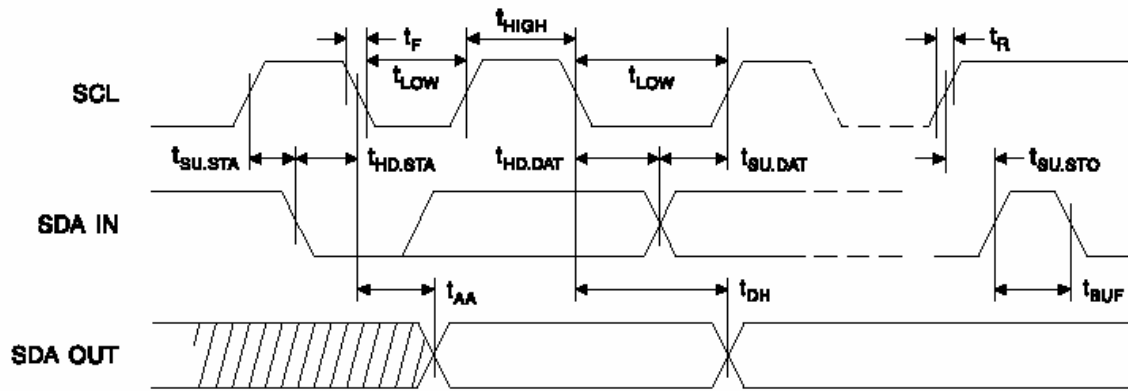


Рисунок 25. Временная диаграмма работы шины.

Временная диаграмма цикла записи

(SCL – линия синхронизации, SDA – линия последовательной передачи данных)

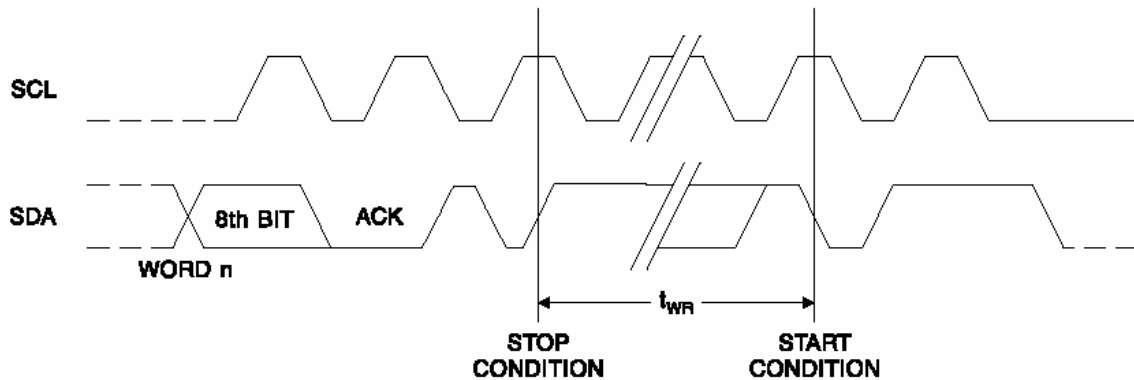


Рисунок 26. Временная диаграмма цикла записи.

Время записи t_{WR} -- это промежуток времени с того момента, когда линия находится в стоп-состоянии при записи последовательных данных, до момента окончания внутреннего цикла очистки/записи.

Достоверность данных

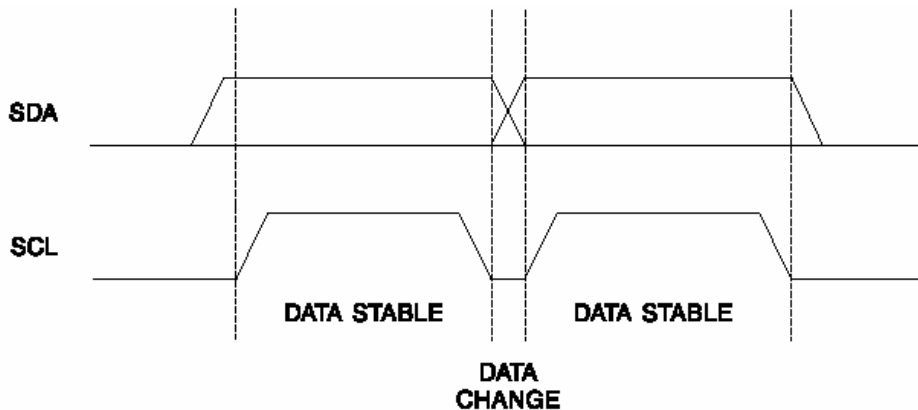


Рисунок 27. Достоверность данных.

Старт- и стоп-состояние

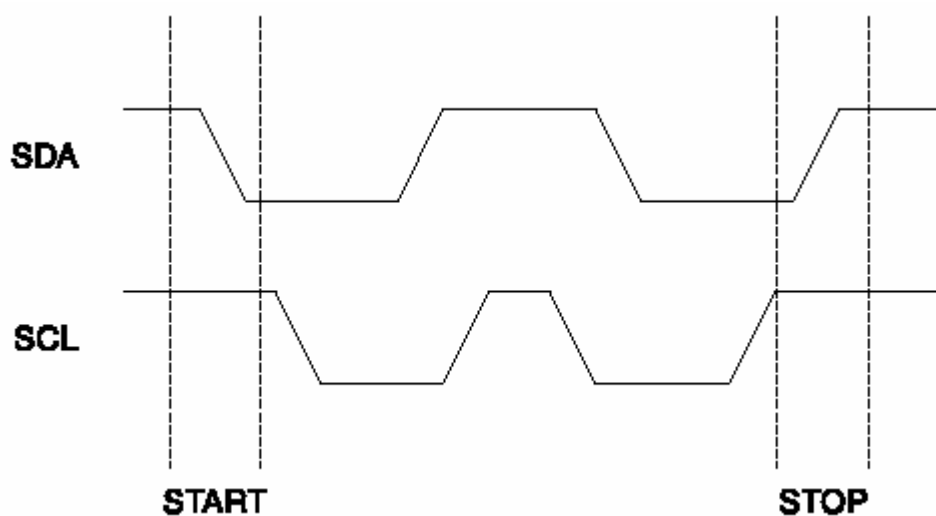


Рисунок 28. Старт- и стоп-состояние.

Подтверждение передачи

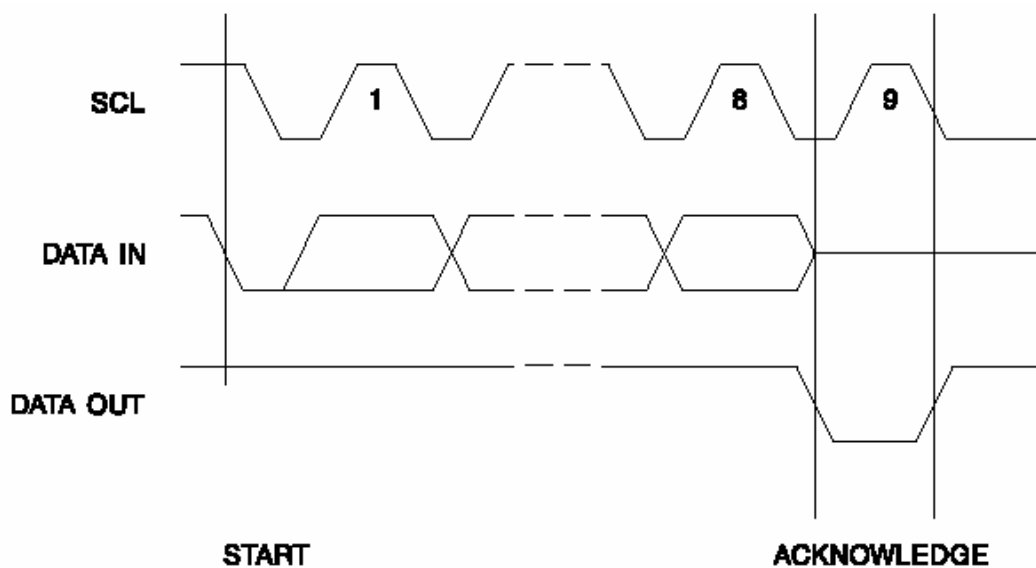


Рисунок 29. Подтверждение передачи.

Адресация устройств

Устройства E²PROM с объемом памяти 1 К, 2 К, 4 К, 8 К и 16 К после перехода в старт-состояние должны получать слово (8 бит) с адресом устройства. Только тогда микросхема сможет произвести операцию чтения или записи (см. рисунок).

1K/2K	1	0	1	0	A ₂	A ₁	A ₀	R/W
	MSD				LSB			
4K	1	0	1	0	A ₂	A ₁	P0	R/W
8K	1	0	1	0	A ₂	P1	P0	R/W
16K	1	0	1	0	P2	P1	P0	R/W

Рисунок 30. Адреса устройств.

Первые четыре бита слова адреса представляют собой обязательную последовательность "1010". Данная последовательность одинакова для всех устройств E²PROM. Следующие 3 бита представляют собой адреса устройств A₂, A₁ и A₀ для 1K/2K E²PROM. Эти биты соответствуют входам с аналогичными названиями.

E²PROM с объемом памяти 4 К использует только биты адресов A₂ и A₁, а P0 представляет собой адрес страницы памяти. Оба бита адресов устройств соответствуют выходам на микросхеме с аналогичными названиями. Вывод A₀ не подключен.

Адресный байт для E²PROM с объемом памяти 8 К содержит только один бит адреса устройства A₂, а биты P1 и P0 используются для адресации страницы памяти. Бит A₂ соответствует выводу A₂ на микросхеме. Выводы A₁ и A₀ не подключены.

В E²PROM с 16 К памяти биты P0, P1 и P2 представляют собой адрес страницы памяти в устройствах 4 К, 8 К и 16 К.

Выводы A₀, A₁ и A₂ не подключены.

Восьмой бит адреса устройств используется для выбора режима чтения/записи. Если бит равен 1, происходит чтение, иначе – запись. После сравнения адресов устройств E²PROM выдает 0. Если сравнение не было произведено, микросхема возвращается в режим ожидания.

Операция записи

Запись байта: После того, как E²PROM получит адресный байт и подтвердит возможность приема, должна происходить операция записи. Получив адрес и ответив выдачей "0", устройство примет первые 8 бит данных. Затем E²PROM выдает "0". Адресующее устройство (например, микроконтроллер) должно остановить процесс записи путем выдачи стоп-сигнала. В этот момент E²PROM начинает цикл записи в постоянную память. До тех пор, пока запись не будет завершена, отключаются все входы и E²PROM не реагирует ни на какие сигналы (см. рисунок).

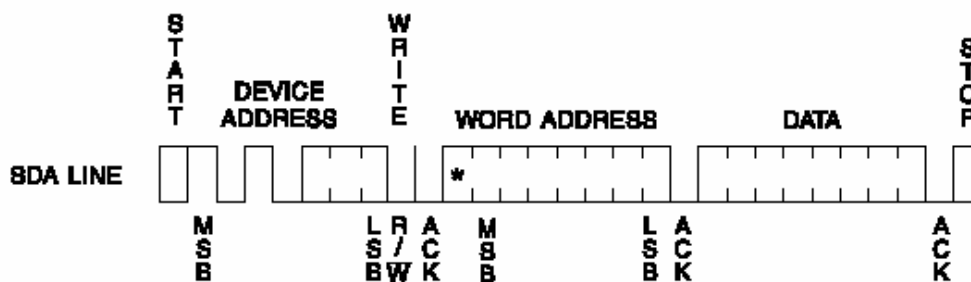


Рисунок 31. Запись байта.

Список обозначений:

- WRITE – запись;
- ACK – сигнал подтверждения;
- SDA LINE – линия передачи данных SDA;
- DATA – данные;
- DEVICE ADDRESS – адрес устройства;
- WORD ADDRESS – адрес слова данных.

Страничная запись: 1K/2K E²PROM может производить страничную запись (по 8 байт), а устройства с объемом памяти в 4 К, 8 К и 16 К производят 16-байтную запись.

Процесс страничной записи инициируется так же, как запись одного байта, отличие в том, что микроконтроллер после передачи первого слова не выдает стоп-сигнал.

Вместо этого, как только E²PROM подтвердит получение первого слова данных, микроконтроллер может передать ему еще до 7 (1 К / 2 К) или 15 (4 К, 8 К, 16 К) слов данных. После получения каждого слова E²PROM будет выдавать на линии "0". Микроконтроллер прекращает страничную запись, выдавая стоп-сигнал (см. рисунок).

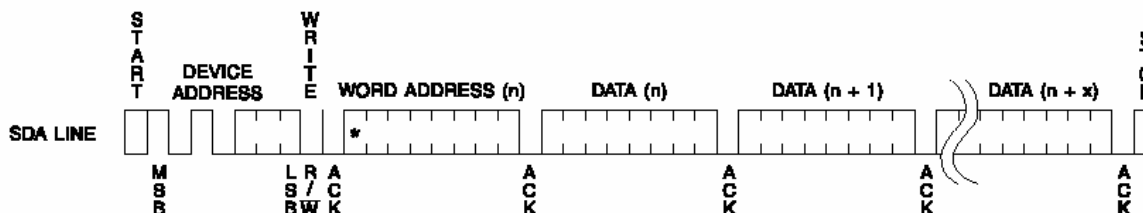


Рисунок 32. Запись страницы.

* Этот бит для устройства 1К может быть любым.

Список обозначений:

- WRITE – запись;
- ACK – сигнал подтверждения;
- SDA LINE – линия передачи данных SDA;
- DATA – данные;
- DEVICE ADDRESS – адрес устройства;
- WORD ADDRESS – адрес слова данных.

Каждый раз при получении слова данных E²PROM инкрементирует младшие 3 (1 К / 2 К) или 4 (4 К, 8 К, 16 К) адресных бита. Старшие адресные биты не инкрементируются.

Опрос устройства: Как только E²PROM начнет внутренне тактируемый цикл записи и отключит свои входы, можно инициировать запрос на подтверждение получения данных. Этот процесс включает отправку слова с адресом устройства, а затем выдачу стоп-сигнала. Бит чтения/записи устанавливается в зависимости от требуемой операции. E²PROM выставит "0", позволяющий продолжить запись или чтение, только после завершения своего внутреннего цикла.

Операции чтения

Операция чтения инициируется точно так же, как и операция записи, за тем исключением, что бит чтения/записи в слове адреса устройства устанавливается равным 1. Существует три операции чтения: чтение текущего адреса, произвольная выборка адреса и последовательное чтение.

Чтение текущего адреса: Внутренний счетчик адреса содержит последний адрес, к которому производилось обращение во время операции чтения или записи, увеличенный на единицу. Этот адрес остается корректным в промежутке между операциями до тех пор, пока к микросхеме подключено питание. Во время чтения адреса "перепрыгивают" с последнего байта последней страницы памяти на первый байт первой страницы. Во время записи адреса "перепрыгивают" с последнего байта текущей страницы на первый байт той же самой страницы.

Как только байт адреса устройства с битом чтения/записи, установленным в единицу, будет синхронизирован и принят E²PROM, слово с текущим адресом данных обновляется. Микроконтроллер выдает на вход не "0", а стоп-сигнал (см. рисунок).

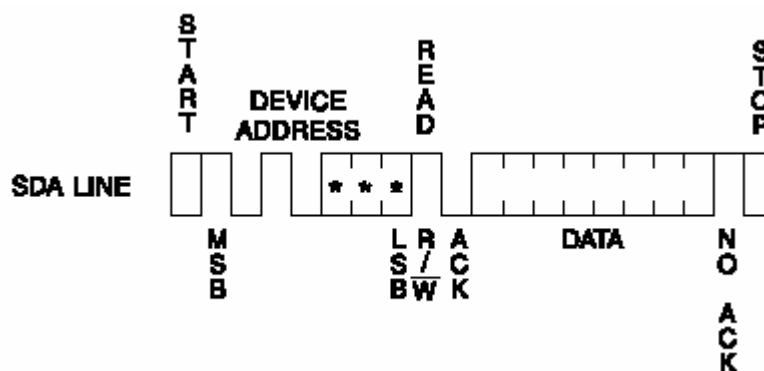


Рисунок 33. Чтение текущего адреса.

Список обозначений:

- READ – чтение;
- ACK – сигнал подтверждения;
- SDA LINE – линия передачи данных SDA;
- DATA – данные;
- DEVICE ADDRESS – адрес устройства.

Чтение в режиме произвольного доступа: Чтение в режиме произвольного доступа означает холостую загрузку байта в адрес слова данных. Как только слово адреса устройства и адрес данных будут приняты E²PROM, микроконтроллер должен сгенерировать еще один старт-сигнал. Он инициирует чтение текущего адреса путем отправки адреса устройства с битом чтения/записи, установленным в единицу. E²PROM подтверждает получение адреса устройства и последовательно считывает слово данных. Микроконтроллер отвечает не выдачей "0", а генерацией стоп-сигнала (см. рисунок).

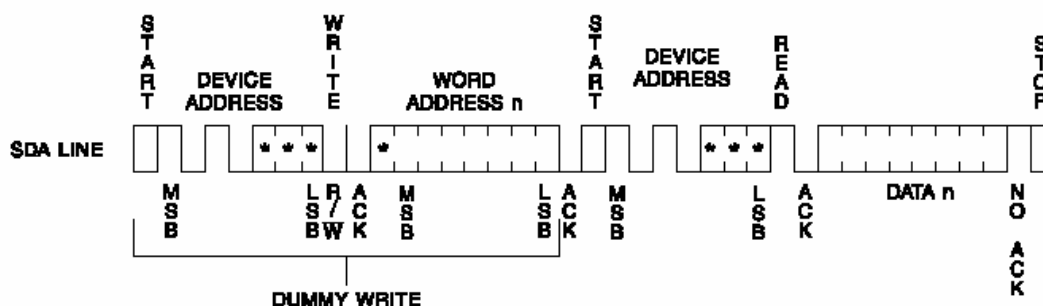


Рисунок 34. Чтение произвольного адреса.

*Эти биты для устройства 1 К могут быть любыми.

Список обозначений:

- READ – чтение;
- WRITE – запись;
- ACK – сигнал подтверждения;
- SDA LINE – линия передачи данных SDA;
- DATA – данные;
- DEVICE ADDRESS – адрес устройства;
- WORD ADDRESS – адрес слова данных;
- DUMMY WRITE – холостая запись.

Чтение в режиме последовательного доступа: Последовательное чтение данных инициируется в процессе либо чтения текущего адреса, либо чтения произвольного адреса.

После того, как микроконтроллер получит слово данных, он подтверждает их получение. Пока E²PROM получает сигнал о подтверждении, она будет продолжать наращивать адрес слова данных и последовательно считывать слова данных. Когда счетчик достигнет верхнего адреса памяти, он "перепрыгнет" на начало и последовательное чтение будет продолжено. Операция последовательного чтения будет остановлена в том случае, если контроллер не выдает на линию "0", а генерирует стоп-сигнал (см. рисунок).

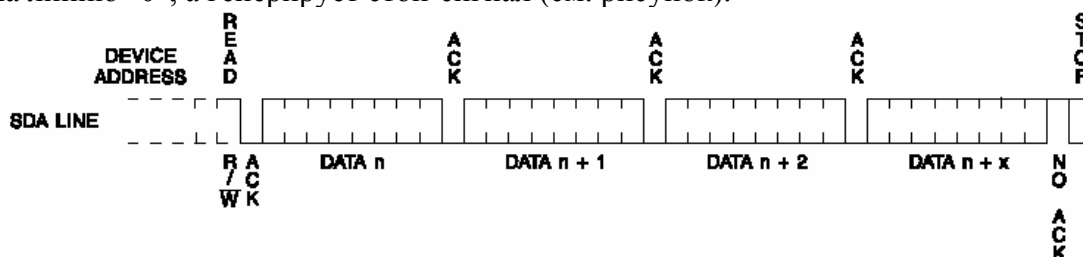


Рисунок 35. Последовательное чтение.

Список обозначений:

- READ – чтение;
- ACK – сигнал подтверждения;
- SDA LINE – линия передачи данных SDA;
- DATA – данные;
- DEVICE ADDRESS – адрес устройства.

ЖКИ WH1602B-YGK-CP

Меры предосторожности при использовании ЖКИ

При использовании ЖКИ нужно:

1. Избегать падений и ударов устройства, а также любых его видоизменений.
2. Не делать дополнительных отверстий на печатной плате, не изменять общий вид модели, не производить замену компонентов ЖКИ.

3. Не разбирать на части.
4. Не использовать при недопустимых значениях параметров.
5. Не ронять, не сгибать.
6. Запаявать только с устройствами ввода/вывода.
7. Хранить в чистом месте и оберегать от воздействия статического электричества.

Общее описание

Таблица 23. Описание ЖКИ.

Элемент	Размерность	Единицы
Число символов	16 символов x 2 строки	
Размер модуля	80.0 x 36.0 x 13.2(MAX)	мм
Область изображения	66.0 x 16.0	мм
Активная область	56.21 x 11.5	мм
Размер точки	0.56 x 0.66	мм
Расстояние между точками	0.60 x 0.70	мм
Размер символа	2.96 x 5.56	мм
Размер символа, включая расстояние между символами	3.55 x 5.94	мм
Тип ЖКИ	STN, положительный, полупрозрачный, серый	
Производительность	1/16	
Направление луча	Стрелка показывает на 12 часов	
Тип подсветки	Желто-зеленая	

Максимально возможные значения параметров

Таблица 24. Максимально возможные значения параметров.

Параметр	Обозначение	Мин.	Обычн.	Макс.	Единицы
Рабочая температура	T_{OP}	0	*	+50	°C
Температура при хранении	T_{ST}	-20	*	+60	°C
Входное напряжение	V_I	V_{SS}	*	V_{DD}	В
Напряжение, подводимое к логической схеме	$V_{DD}-V_{SS}$	-0.3	*	7	В
Напряжение, подводимое к ЖКИ	$V_{DD}-V_0$	-0.3	*	13	В

Электрические характеристики

Таблица 25. Электрические характеристики ЖКИ.

Параметр	Обозначение	Условия	Мин.	Средн.	Макс.	Единицы
Напряжение питания для логической схемы	$V_{DD}-V_{SS}$		4.5		5.5	В
Напряжение питания для ЖКИ	$V_{DD}-V_0$	$T_a=-20^{\circ}C$	*		5.2	В
		$T_a=25^{\circ}C$	*	3.8	*	В
		$T_a=70^{\circ}C$	3.4		*	В
Максимальное входное напряжение	V_{IH}	*	2.2	*	V_{DD}	В
Минимальное входное напряжение	V_{IL}	*	*	*	0.6	В
Максимальное выходное напряжение	V_{OH}	*	2.4	*	*	В
Минимальное выходное напряжение	V_{OL}	*	*	*	0.4	В
Ток питания	I_{DD}	$V_{DD}=5V$	*	1.2	*	мА

Оптические характеристики

Таблица 26. Оптические характеристики ЖКИ.

Параметр	Обозначение	Условия	Мин.	Средн.	Макс.	Единицы
Угол видимости	(V)	CR2	10	*	105	градусы
	(H)	CR2	-40	*	40	градусы
Контрастность	CR	*	*	3	*	*
Время ответа	T подъема	*	*	150	200	мс
	T спада	*	*	150	200	мс

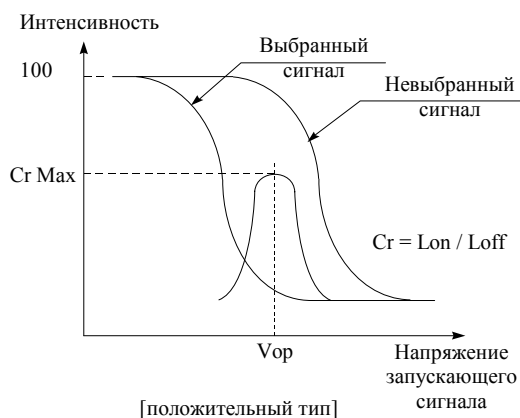


Рисунок 36. Определение действующего напряжения (V_{op}).

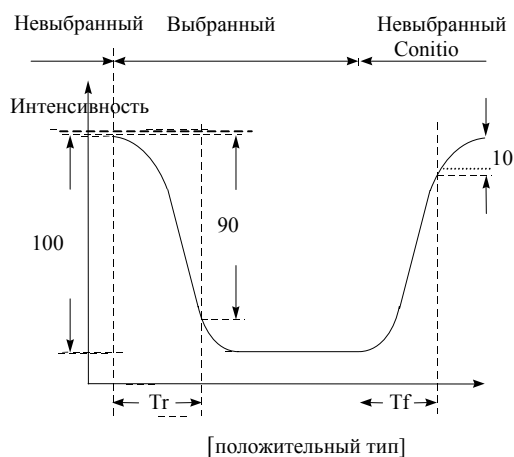


Рисунок 37. Определение времени ответа (T_r , T_f).

Условия:

- Действующее напряжение V_{op} .
- Частота 64 HZ.
- Задающая форма сигнала: 1/N производительность, 1/a смещение.

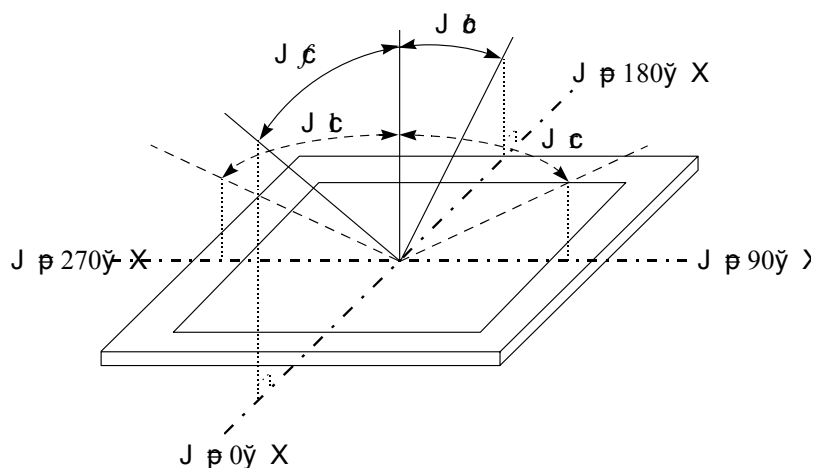


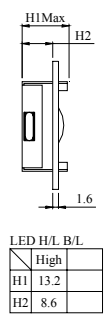
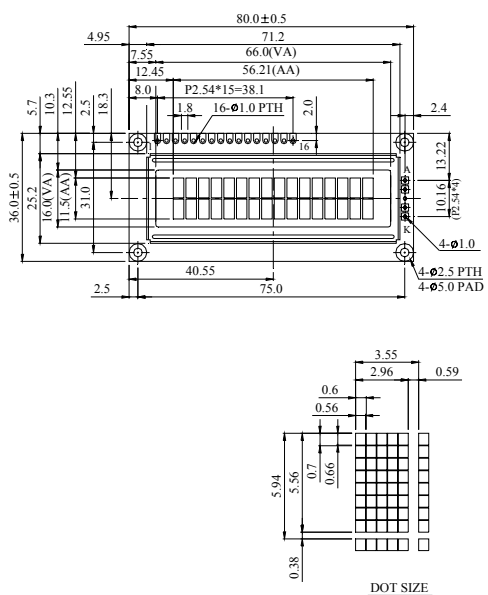
Рисунок 38. Определение угла видимости.

Описание выходов

Таблица 27. Выходы ЖКИ.

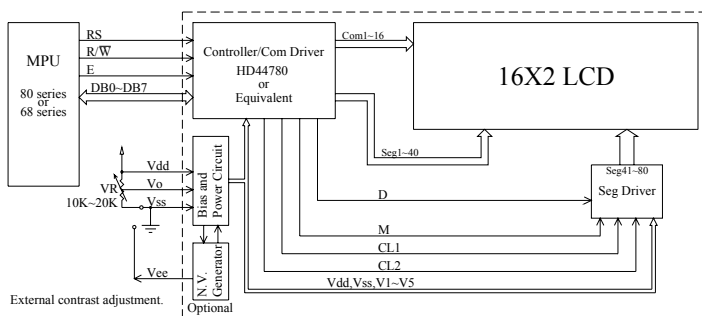
№ выхода	Обозначение	Уровень	Описание
1	V_{SS}	0В	Заземление
2	V_{DD}	5.0В	Напряжение питания для логической схемы
3	VO	(переменный)	Напряжение питания для ЖКИ
4	RS	H/L	H – данные, L – команды
5	R/W	H/L	H – чтение (с ЖКИ), L – запись (в ЖКИ)
6	E	H,H/L	Разрешающий сигнал
7	DB0	H/L	Бит данных 0
8	DB1	H/L	Бит данных 1
9	DB2	H/L	Бит данных 2
10	DB3	H/L	Бит данных 3
11	DB4	H/L	Бит данных 4
12	DB5	H/L	Бит данных 5
13	DB6	H/L	Бит данных 6
14	DB7	H/L	Бит данных 7
15	A	*	Светодиод +
16	K	*	Светодиод

Схематическое изображение и блок-схема

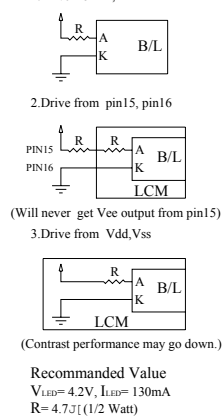


PIN NO.	SYMBOL
1	Vss
2	Vdd
3	Vo
4	RS
5	R/W
6	E
7	DB0
8	DB1
9	DB2
10	DB3
11	DB4
12	DB5
13	DB6
14	DB7
15	A/Vee
16	K

The non-specified tolerance of dimension is ±0.3mm.



LED B/L Drive Method



Recommended Value
 V_{LED}= 4.2V, I_{LED}= 130mA
 R= 4.7Ω (1/2 Watt)

Character located	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
DDRAM address	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
DDRAM address	40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F

Рисунок 39

Обозначения:

- The non-specified tolerance of dimension is ± 3mm – допустимое отклонение размеров составляет ± 3 мм.
- External contrast adjustment – внешняя регулировка контрастности.
- MPU 80 or 68 series – микропроцессор серии 80 или 68.
- Controller/Com Driver HD44780 or Equivalent – контроллер / COM-драйвер HD44780 или аналогичный.
- Bias and power circuit – схема питания.
- Seg driver – драйвер сегментов.
- LED B/L Drive Method – различные варианты подсветки.
- Drive from A, K – питание через выводы A, K.
- Drive from pin 15, pin 16 – питание через выводы 15, 16.
- Drive from Vdd,Vss - питание через Vdd и Vss.
- Contrast performance may go down – характеристика контрастности может понизиться.
- Recommended Value – рекомендуемая величина.
- Character located – размещение символов.

Описание функций

Модуль ЖКИ встроен в контроллер (БИС) и имеет два 8-битовых регистра: регистр команд (IR) и регистр данных (DR).

Регистр команд хранит коды таких операций, как очистка дисплея, перемещение курсора, а также информацию об адресах памяти отображаемых данных (DDRAM) и генератора символов (CGRAM). В регистр команд можно только записывать информацию из микропроцессора. Регистр данных временно хранит данные, предназначенные для записи или чтения из DDRAM или CGRAM. Когда адресная информация записывается в регистр команд, данные из DDRAM или CGRAM сохраняются в регистре данных. Эти два регистра можно выбрать с помощью регистрового переключателя (RS).

Таблица 28

RS	R/W	Команда
0	0	IR используется для внутренних команд (очистка дисплея и т.д.).
0	1	Считывание флага занятости (DB7) и счетчика адреса (от DB0 до DB7).
1	0	Запись данных в DDRAM или CGRAM (из регистра данных в DDRAM или CGRAM).
1	1	Чтение данных из DDRAM или CGRAM (из DDRAM или CGRAM в регистр данных).

Флаг занятости (BF)

Если флаг занятости равен 1, это значит, что БИС занята выполнением внутренних операций и следующая команда не может быть принята. Если RS=0 и R/W=1, содержимое флага занятости передается в бит DB7. Следующая команда должна быть записана только при значении флага занятости, равном 0.

Счетчик адреса (AC)

Счетчик адреса (AC) назначает адреса и DDRAM, и CGRAM.

Память данных ЖКИ (DDRAM)

Эта память используется для хранения данных, выводимых на дисплей. Один символ представлен в виде 8-битного кода. Объем памяти составляет 80×8 битов или 80 символов.

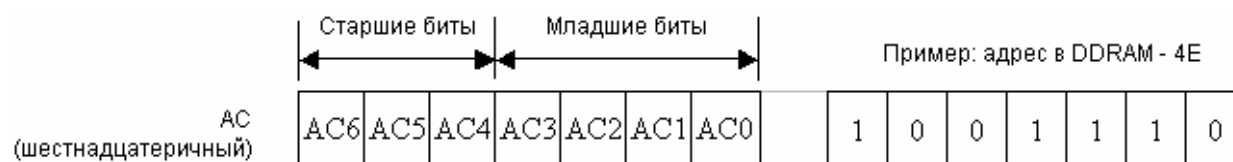


Рисунок 40

Ниже приведена схема соответствия между адресами DDRAM и позициями ЖКИ.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F

Дисплей: 2 строки по 16 символов

Рисунок 41. Соответствие между адресами DDRAM и позициями ЖКИ.

Генератор символов, встроенный в ПЗУ (CGROM)

CGROM генерирует символы размером 5×8 или 5×10 точек на основе 8-битных кодов символов.

Генератор символов ОЗУ (CGRAM)

В CGRAM пользователь может программно генерировать символы. Можно определить 8 символов размером 5×8 точек и 4 символа размером 5×10 точек.

Коды символов нужно записывать в DDRAM по адресам, отображенным в таблице. В таблице также показано, как отобразить символ, хранящийся в CGROM.

Соотношение между адресами CGRAM, кодами символов (DDRAM) и шаблонами СИМВОЛОВ.

Таблица 29. Соотношение между адресами CGRAM, кодами символов (DDRAM) и шаблонами символов. Для символов размером 5x8 точек.

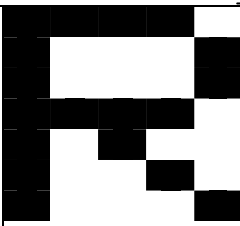
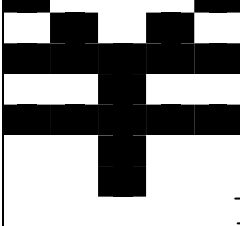
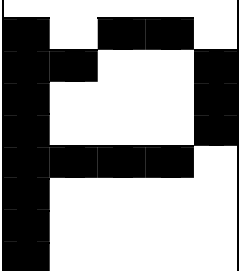
Код символа (данные DDRAM)		Адреса CGRAM		Символ (данные в CGRAM)	
Старшие	Младшие	Старш.	Младш.	Старшие	Младшие
					Образ символа №1 ↑ Курсор
					Курсор ↑ Образ символа №2 ↑ Курсор

Таблица 30. Соотношение между адресами CGRAM, кодами символов (DDRAM) и шаблонами СИМВОЛОВ. Для символов размером 5x10 точек.

Код символа (значение из DDRAM)		Адрес CGRAM		Символ (данные в CGRAM)	
Старшие	Младшие	Старш.	Младш.	Старшие	Младшие
					Образ символа пользователя ↑ Символ курсора

Образы символов, хранящихся в ПЗУ

Таблица 31. Образы символов, хранящихся в ПЗУ.

Upper 4 bit Lower 4 bit	LLLL	LLLH	LLHL	LLHH	LHLL	LHLH	LHHL	LHHH	HLLL	HLLH	HLHL	HLHH	HHLL	HHLH	HHHL	HHHH
LLLL	CG RAM (1)			0	1	2	3	4			5	6	7	8	9	.
LLLH	CG RAM (2)		!	0	1	2	3	4			5	6	7	8	9	.
LLHL	CG RAM (3)		"	0	1	2	3	4			5	6	7	8	9	.
LLHH	CG RAM (4)		#	0	1	2	3	4			5	6	7	8	9	.
LHLL	CG RAM (5)		\$	0	1	2	3	4			5	6	7	8	9	.
LHLH	CG RAM (6)		%	0	1	2	3	4			5	6	7	8	9	.
LHHL	CG RAM (7)		&	0	1	2	3	4			5	6	7	8	9	.
LHHH	CG RAM (8)		'	0	1	2	3	4			5	6	7	8	9	.
HLLL	CG RAM (1)		(0	1	2	3	4			5	6	7	8	9	.
HLLH	CG RAM (2))	0	1	2	3	4			5	6	7	8	9	.
HLHL	CG RAM (3)		*	0	1	2	3	4			5	6	7	8	9	.
HLHH	CG RAM (4)		+	0	1	2	3	4			5	6	7	8	9	.
HHLL	CG RAM (5)		,	0	1	2	3	4			5	6	7	8	9	.
HHLH	CG RAM (6)		-	0	1	2	3	4			5	6	7	8	9	.
HHHL	CG RAM (7)		.	0	1	2	3	4			5	6	7	8	9	.
HHHH	CG RAM (8)		/	0	1	2	3	4			5	6	7	8	9	.

Таблица команд

Таблица 32

Команда	Код операции										Описание	Время выполн. (fosc = 270 КГц)	
	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0			
Очистка экрана	0	0	0	0	0	0	0	0	0	1	Запись "00H" в DDRAM и установка адреса DDRAM на "00H" из AC.	1.53 мс	
Возврат в начало строки	0	0	0	0	0	0	0	0	0	1	*	Установка адреса DDRAM на "00H" из AC и возврат курсора в начало строки, если он был смещен. Содержимое DDRAM не меняется.	1.53 мс
Начальные установки	0	0	0	0	0	0	0	0	1	I/D	SH	Задаёт направление перемещения курсора и разрешает сдвиг сразу всех символов.	39 мс
Дисплей ON/OFF	0	0	0	0	0	0	0	1	D	C	B	Устанавливает / отключает биты, отвечающие за режим дисплея (D), отображение курсора (C), мерцание курсора (B).	39 мс
Передвиж. курсора по экрану	0	0	0	0	0	0	1	S/C	R/L	*	*	Установка бита движения курсора и смещения всех символов, указание направления смещения без изменения данных в DDRAM.	39 мс
Функц. установки	0	0	0	0	0	1	DL	N	F	*	*	Установка длины данных (DL:8-бит/4-бита), количества строк на дисплее (N:2-строки или 1) и размера символов (F:5×11 точек/5×8 точек).	39 мс
Установка адреса CGRAM	0	0	0	1	AC5	AC4	AC3	AC2	AC1	AC0	Установка адреса CGRAM в счетчик адреса.	39 мс	
Установка адреса DDRAM	0	0	1	AC6	AC5	AC4	AC3	AC2	AC1	AC0	Установка адреса DDRAM в счетчик адреса.	39 мс	
Чтение флага занятости и адреса	0	1	BF	AC6	AC5	AC4	AC3	AC2	AC1	AC0	Прочитав флаг занятости, можно определить, занят ли контроллер выполнением внутренних операций. Также можно прочесть содержимое счетчика адреса.	0 мс	
Записать данные в память	1	0	D7	D6	D5	D4	D3	D2	D1	D0	Запись данных во внутреннюю память (DDRAM/CGRAM).	43 мс	
Чтение данных из памяти	1	1	D7	D6	D5	D4	D3	D2	D1	D0	Чтение данных из внутренней памяти (DDRAM/CGRAM).	43 мс	

**"-Не имеет значения

Временные характеристики

Запись команды

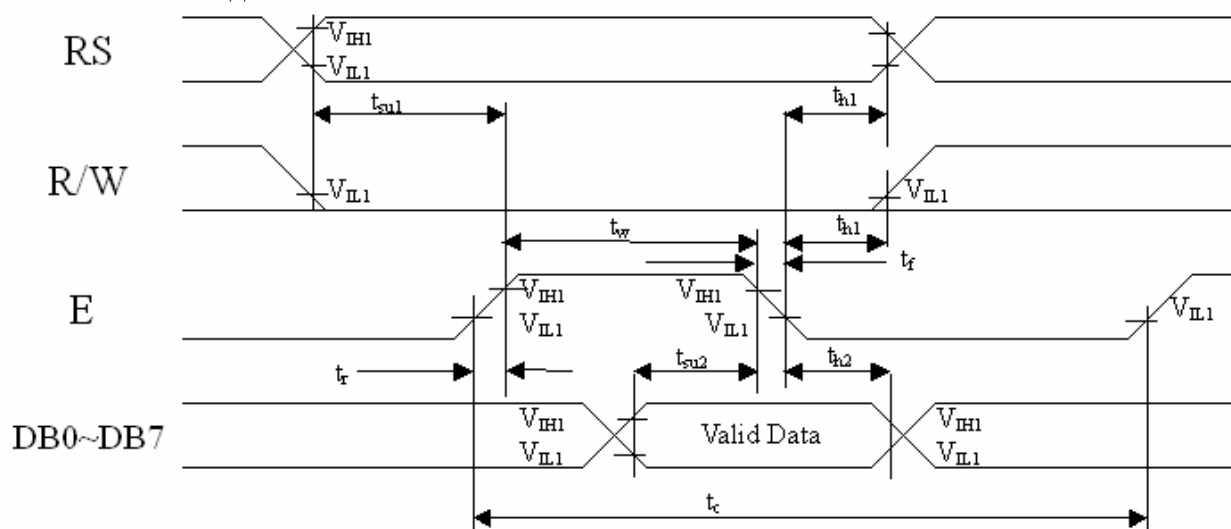


Рисунок 42

($V_{DD}=4.5V \dots 5.5V$, $T_a=-30 \dots +85^{\circ}C$)

Таблица 33

Режим	Характеристика	Обозн.	Мин.	Средн.	Макс.	Единицы
Режим записи	Сигнал E, время полного такта	tc	500	*	*	нс
	Сигнал E, время нарастания/спада	tr, tf	*	*	20	
	Сигнал E, длительность импульса (высокий, низкий)	tw	230	*	*	
	Время установки сигналов R/W и RS	tsu1	40	*	*	
	Время передачи сигналов R/W and RS	th1	10	*	*	
	Время установки сигнала передачи данных	tsu2	80	*	*	
	Время передачи данных	th2	10	*	*	

Операция чтения

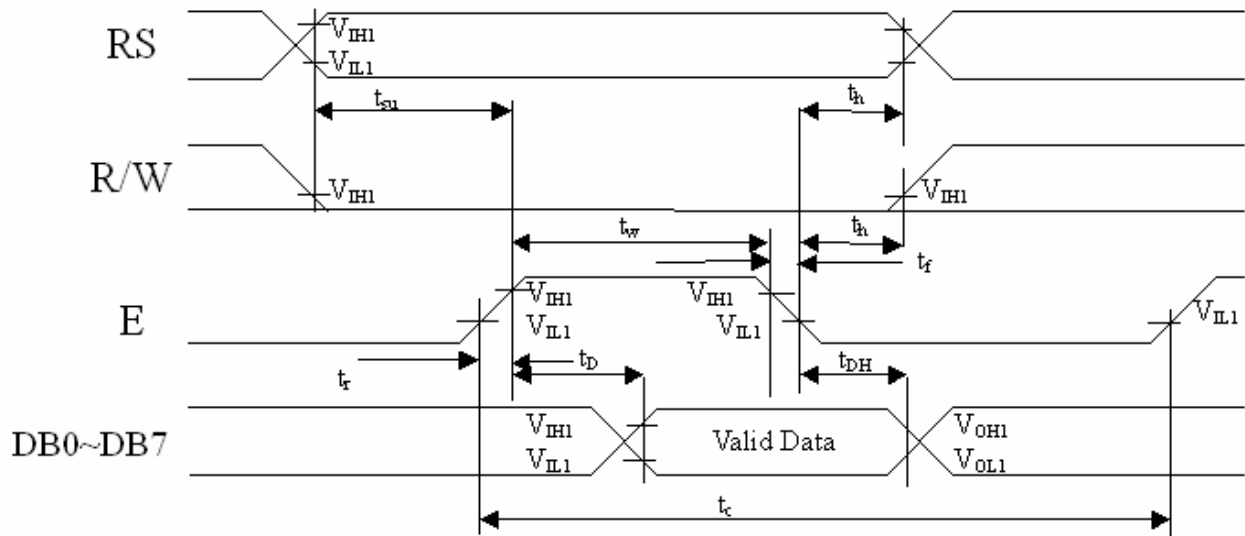


Рисунок 43

 $(V_{DD}=4.5V..5.5V, T_a=-30...+85^{\circ}C)$

Таблица 34

Режим	Характеристика	Обозн.	Мин.	Средн.	Макс.	Единицы
Режим чтения	Сигнал E, время полного такта	t_c	500	*	*	нс
	Сигнал E, время нарастания/спада	t_r, t_f	*	*	20	
	Сигнал E, длительность импульса (высокий, низкий)	t_w	230	*	*	
	Время установки сигналов R/W и RS	t_{su}	40	*	*	
	Время передачи сигналов R/W и RS	t_{th}	10	*	*	
	Время задержки вывода данных	t_d	*	*	120	
	Время передачи данных	t_{dH}	5	*	*	

Инициализация LCM

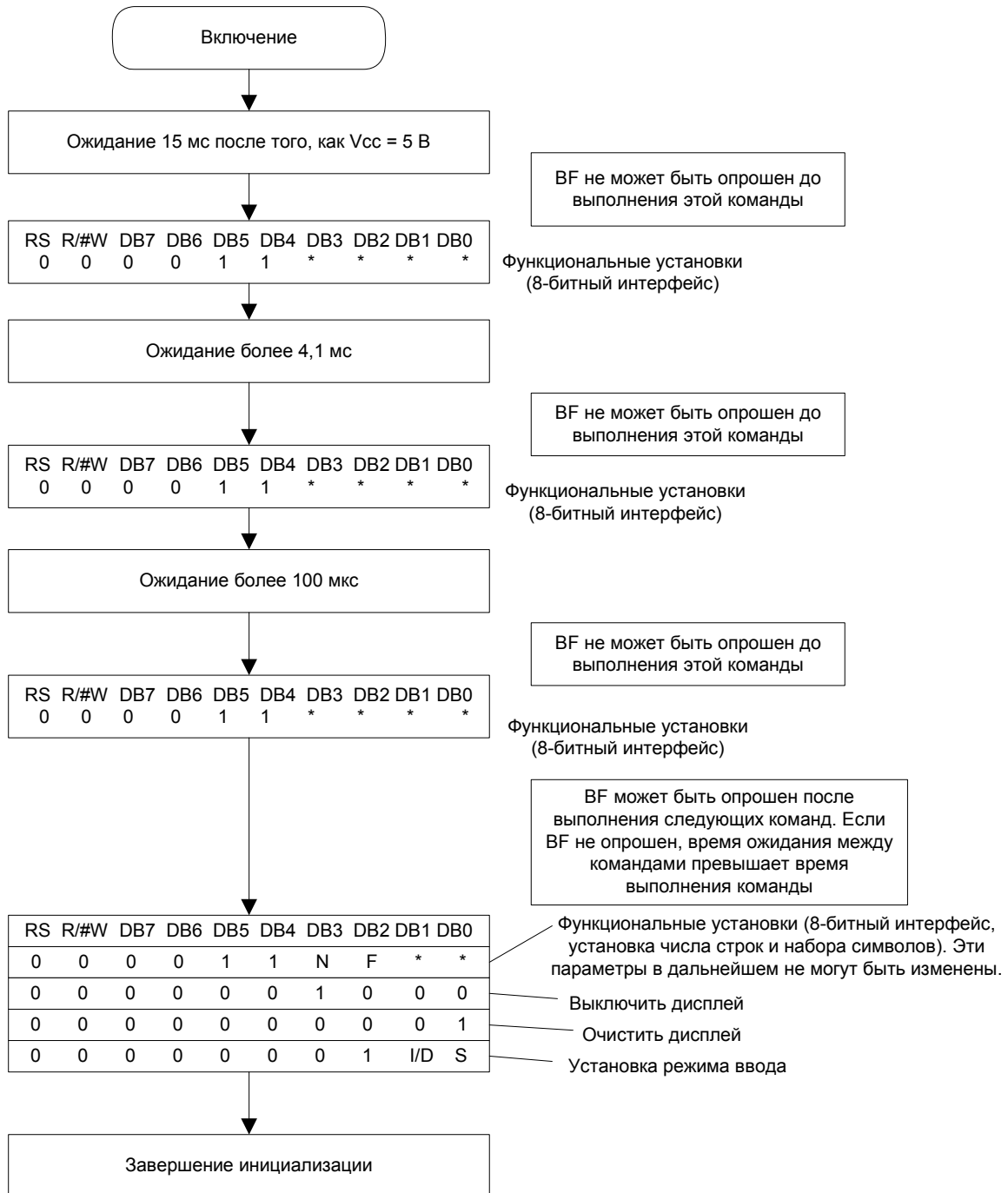


Рисунок 44. 8-битный интерфейс.

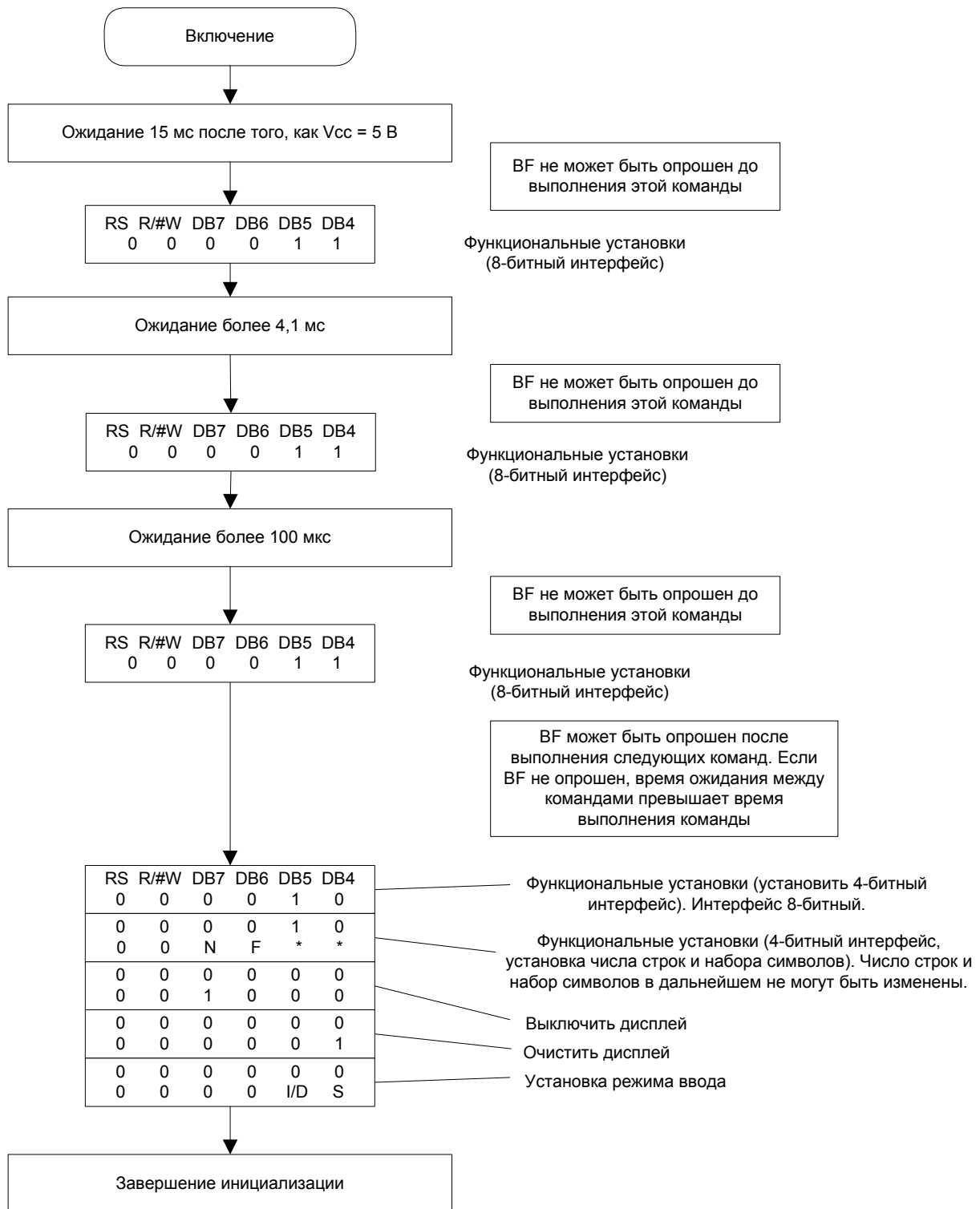


Рисунок 45. 4-битный интерфейс.

Информация о подсветке

Таблица 35. Информация о подсветке.

Параметр	Обозначение	Мин.	Средн.	Макс.	Единицы	Условия теста
Ток питания	I _{LED}	—	130	—	мА	V=4.2 В
Напряжение питания	V	—	4.2	4.6	В	—
Обратное напряжение	V _R	—	—	8	В	—
Сила света	I _V	10	□	□	Кд/м ²	I _{LED} =130 мА
Длины волны	λ _p	—	573	□	Нм	I _{LED} =130 мА
Время функционирования	—	—	100000	—	Час	4.6 В
Цвет	Желто-зеленый					

Часы / календарь с ОЗУ 240x8 бит PCF8583

Особенности

- I²C-интерфейс.
- Диапазон рабочих напряжений питания в пределах от 1.0 В до 6.0 В (при температуре от 0 до +70 °С).
- Низковольтная память объемом 240x8 бит.
- Напряжение сохранения данных от 1.0 В до 6.0 В.
- Рабочий ток (при частоте f_{SCL} = 0 Гц): максимум 50 мА.
- Календарь на 4 года.
- Универсальный таймер с сигналом и индикацией.
- 12- или 24-часовой формат времени.
- Внешний генератор— 32.768 кГц или 50 Гц.
- Последовательная шина ввода/вывода (I²C).
- Автоматическое наращивание адреса при работе с памятью.
- Программируемые динамик, таймер и функции прерывания.
- Адреса Slave-устройств:
 - Чтение: A1 или A3;
 - Запись: A0 или A2.

Общее описание

Микросхема PCF8583, содержащая часы/календарь, содержит оперативную память на МОП-транзисторах объемом в 2048 бит, состоящую из 256 слов по 8 бит. Адреса и данные передаются последовательно через двунаправленную шину I²C. Встроенный регистр адреса автоматически наращивается после чтения или записи каждого байта данных.

Адресный вывод A0 используется для программирования адресов устройств, что позволяет подсоединять к шине два устройства без использования какого-либо дополнительного аппаратного обеспечения. Встроенная микросхема генератора, работающая на частоте 32.768 кГц, и первые 8 байт оперативной памяти используются для часов, календаря и функций счетчика. Следующие 8 байт могут быть запрограммированы на использование в качестве регистров сигнализации, или же к ним можно обращаться как к свободным адресам памяти. Остальные 240 байт относятся к оперативной памяти.

Краткие характеристики

Таблица 36. Часы-календарь PCF8583. Характеристики.

Обозн-е	Параметр	Условие	Мин.	Обычн.	Макс.	Ед.
Vdd	Напряжение питания в рабочем режиме	Шина I ² C в активном режиме	2.5		6.0	V
		Шина I ² C в неактивном режиме	1.0		6.0	V
Idd	Напряжение питания в рабочем режиме	f _{SCL} = 100 кГц			200	мкА
Iddo	Напряжение питания в режиме часов	f _{SCL} = 0 Гц; V _{DD} = 5 V f _{SCL} = 0 Гц; V _{DD} = 1 V		10 2	50 10	мкА
Tamb	Допустимые пределы температуры окружающей среды		-40		+85	°C
Tstg	Допустимые пределы температуры хранения		-65		+150	°C

Блок-схема

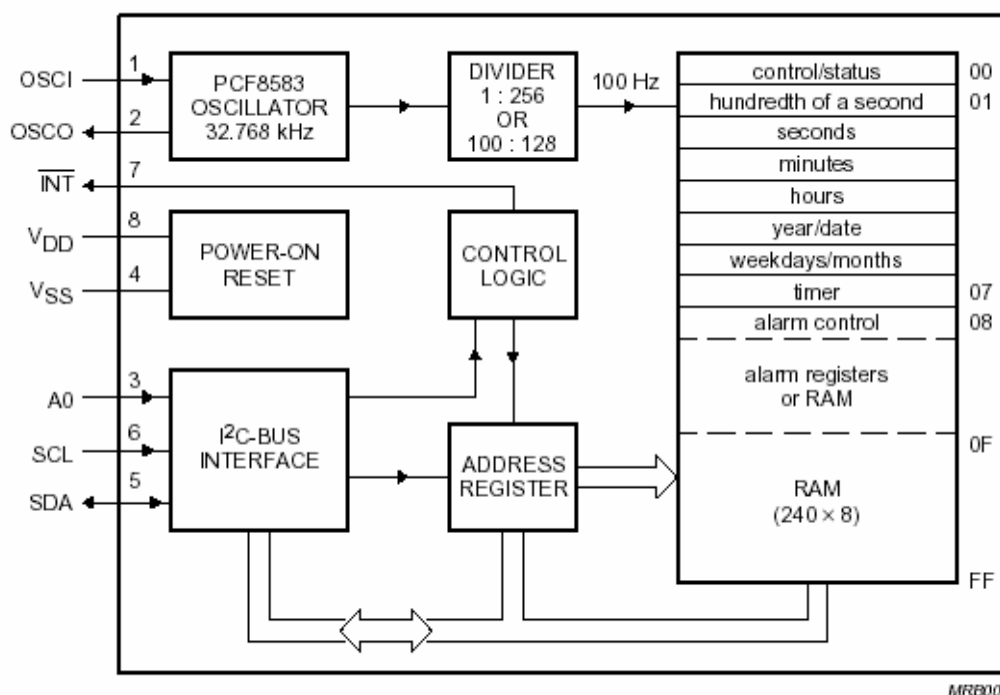


Рисунок 46. Часы / календарь с ОЗУ 240x8 бит PCF8583. Блок-схема.

Обозначения:

- PCF8583 OSCILLATOR – тактовый генератор;
- POWER-ON RESET – сброс по включению питания;
- I²C-BUS INTERFACE – интерфейс шины I²C;
- DIVIDER – делитель;
- CONTROL LOGIC – логика управления;
- ADDRESS REGISTER – адресный регистр.

Таблица 37. Устройство памяти.

Назначение ячеек	Адрес
Управление / состояние	00h
Одна сотая доля секунды	01h
Секунды	02h
Минуты	03h
Часы	04h
Год / дата	05h
Дни недели / месяцы	06h
Таймер	07h
Управление сигналом	08h
Регистры сигнала или ячейки памяти	0Fh
ОЗУ (240x8)	10h- FFh

Выводы микросхемы

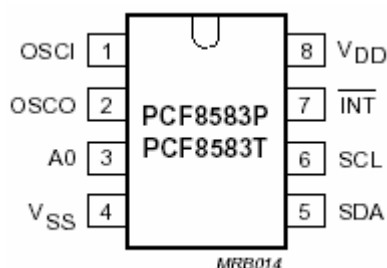


Рисунок 47. Выводы микросхемы.

Таблица 38. Назначение выводов микросхемы PCF8583.

Обозначение	Вывод	Описание
OSCI	1	Вход генератора на частоте 50 Гц или вход для импульса по событию.
OSCO	2	Выход генератора.
A0	3	Адресный вход.
Vss	4	Отрицательный импульс.
SDA	5	Последовательная линия данных.
SCL	6	Последовательная линия синхронизации.
INT	7	Выход прерывания с открытым стоком (активный низкий уровень выходного сигнала).
Vdd	8	Положительный импульс.

Описание функций

Микросхема PCF8583 содержит 8-битную оперативную память объемом 256 байт с 8-битным адресным регистром, осуществляющим автоматическое инкрементирование адреса, встроенную микросхему генератора (частота 32.768 кГц), делитель частоты, последовательную двунаправленную шину I²C и схему, осуществляющую сброс по включению питания.

Первые 16 байт ОЗУ (адреса памяти от 00 до 0F) представляют собой адресуемые 8-битовые регистры специального назначения. Первый регистр (адрес 00) используется в качестве регистра управления/состояния. Регистры по адресам с 01 по 07 – счетчики для функций часов. Регистры, расположенные по адресам с 08 по 0F, могут быть запрограммированы в качестве регистров сигнала или использованы как обычные регистры памяти (когда сигналы отключены).

Режимы счетчика

При программировании регистра управления/состояния может быть установлен режим часов на частоте 32.768 кГц, режим часов на частоте 50 Гц или режим счетчика событий. В том случае, если выбран режим часов, сотые доли секунды, секунды, минуты, часы, дата, месяц (календарь на 4 года) и дни недели хранятся в двоично-десятичном формате. Режим счетчика используется для подсчета импульсов, выдаваемых на вход генератора (к выводу OSCO ничего не подключается). Счетчик событий хранит до 6 цифр данных.

При чтении одного из счетчиков (адреса с 01 по 07) содержимое всех счетчиков стробируется в регистры-заселки в начале цикла чтения. Таким образом предотвращаются ошибки чтения счетчика. При записи в счетчик с другими счетчиками ничего не происходит.

Режим сигнализации

При установке в регистре управления/состояния бита, разрешающего сигнал, активируется регистр управления сигналом (адрес 08).

Используя настройки регистра управления сигналом, можно запрограммировать срабатывание сигнала при наступлении определенной даты, ежедневного сигнала, сигнала по дням недели и по времени. В режиме часов регистр таймера (адрес 07) может быть запрограммирован для подсчета сотых долей секунды, секунд, минут, часов и дней. Подсчет дней ведется, если не запрограммирован сигнал.

Каждый раз при наступлении "сигнального" события устанавливается соответствующий флаг регистра управления/состояния. Событие по таймеру устанавливает флаг сигнала, а в случае переполнения таймера устанавливается флаг таймера. В случае установки (разрешения) флага сигнала или таймера происходит включение вывода прерывания с открытым стоком (с активным низким уровнем выходного сигнала). Флаги остаются установленными до тех пор, пока они не будут сброшены напрямую в результате операции записи.

Если сигнал отключен (то есть бит 2 регистра управления/состояния равен 0), регистры сигналов (адреса с 08 по 0F) могут быть использованы как свободные ячейки памяти.

Регистр управления/состояния

Этот регистр расположен по адресу 00 в памяти и поддерживает свободный доступ в процессе операций чтения/записи через шину I²C. Всеми функциями можно управлять, соответствующим образом устанавливая биты регистра управления/состояния.

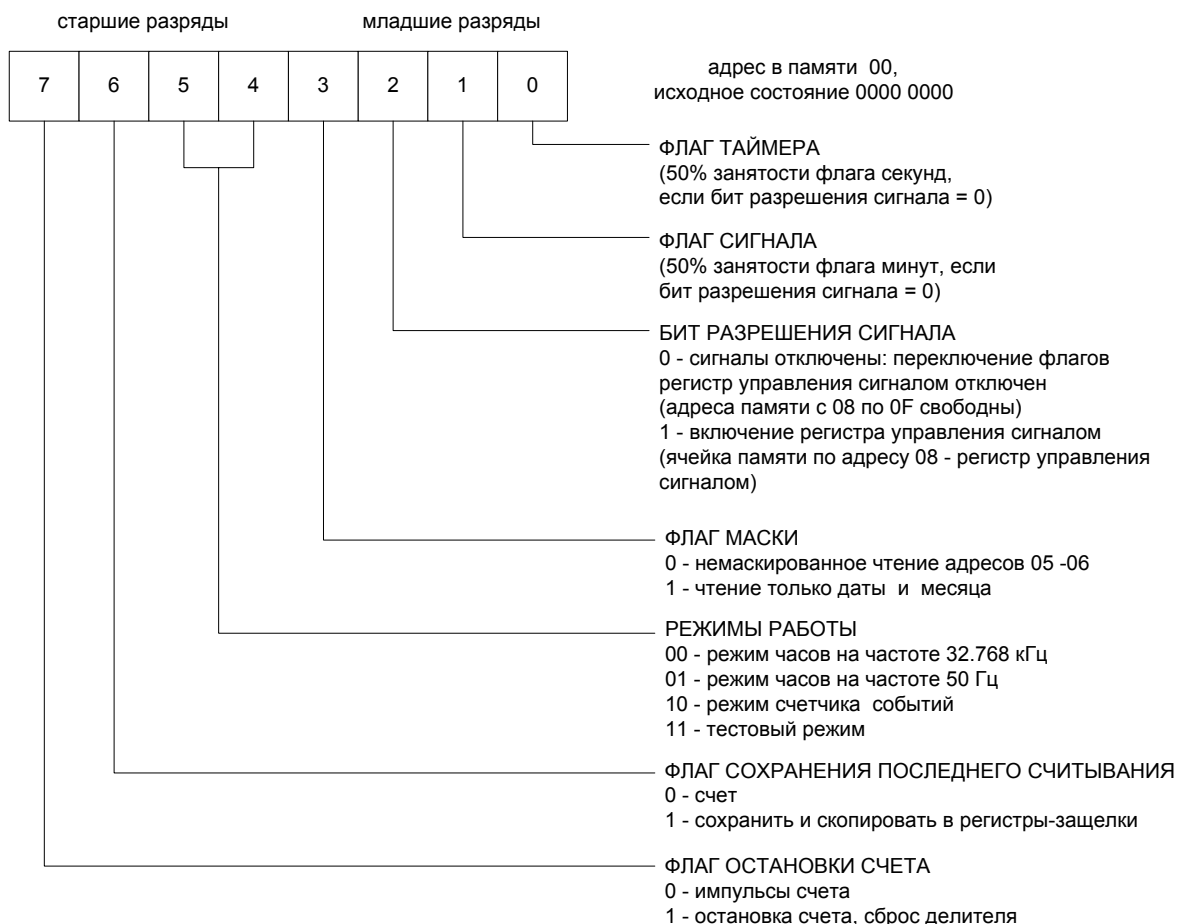


Рисунок 48

Регистры-счетчики

В режиме часов можно устанавливать 12- и 24-часовые форматы времени путем изменения соответствующих битов регистра счетчика часов. Формат счетчика часов представлен на рисунке.

Год и дата упакованы в памяти по адресу 05. Дни недели и месяцы хранятся по адресу 06. При чтении этих адресов год и дни недели маскируются (если установлен флаг маски регистра управления/состояния). Это дает возможность пользователю напрямую считывать дату и месяц.

В режиме счетчика коды событий хранятся в двоично-десятичном формате. Наиболее значимой является цифра D5, наименее значимой – D0. Ниже на рисунке представлены работа и расположение регистров счетчиков в различных режимах работы. Циклы счетчиков перечислены в таблице.

управление/состояние	
доли секунды	
1/10	1/100
секунды	
10 сек	1 сек
минуты	
10 мин	1 мин
часы	
10 часов	1 час

управление/состояние		00
D1	D0	01
D3	D2	02
D5	D4	03
свободна		04

год/дата	свободна	05
10 дней 1 день	свободна	06
день недели/месяц	таймер	07
10 мес 1 мес	T1 T0	08
таймер	управление сигналом	09
10 дней 1 день	сигнал D1 сигнал D0	0A
управление сигналом	D3 D2	0B
доли секунды	D5 D4	0C
1/10 1/10	свободна	0D
сигнал по секундам	свободна	0E
сигнал по минутам	сигнал по таймеру	0F
сигнал по часам	свободная память	
сигнал по дате		
сигнал по месяцу		
сигнал по таймеру		
свободная память		

режим часов

режим счетчика

Рисунок 49. Размещение регистров.

Таблица 39. Длины циклов счетчиков времени, режим часов.

Единицы	Счетный цикл	Переход к началу цикла	Содержимое счетчика месяцев
Сотые доли секунды	от 00 до 99	99 → 00	-
Секунды	от 00 до 59	59 → 00	-
Минуты	от 00 до 59	59 → 00	-
Часы (24 часа)	от 00 до 23	23 → 00	-
Часы (12 часов)	12 AM	-	-
	от 01 AM до 11 AM	-	-
	12 PM	-	-
	от 01 PM до 11 PM	11 PM → 12 AM	-
Дата	от 01 до 31	31 → 01	1, 3, 5, 7, 8, 10 и 12
	от 01 до 30	30 → 01	4, 6, 9 и 11
	от 01 до 29	29 → 01	2, год = 0
	от 01 до 28	28 → 01	2, год = 1, 2, 3
Месяцы	от 1 до 12	12 → 01	-
Год	до 0 до 3	-	-
Дни недели	от 0 до 6	6 → 0	-
Таймер	от 00 до 99	нет перехода	-



Рисунок 50. Формат счетчика часов.

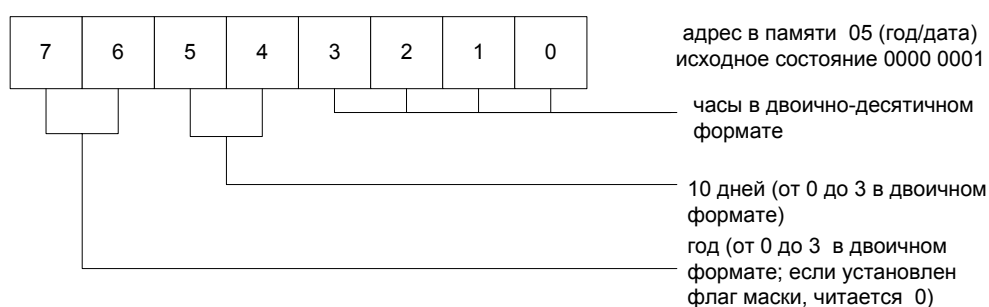


Рисунок 51. Формат счетчика года/даты.

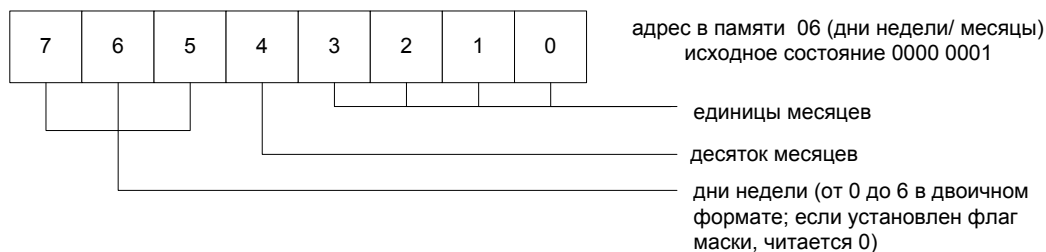


Рисунок 52. Формат счетчика дней недели/месяцев.

Регистр управления сигналом

Когда бит разрешения сигнала (регистр управления/состояния, адрес 00, бит 2) установлен, активируется регистр управления сигналом (адрес 08). Все функции, связанные с сигналом, таймером и выходом прерываний, управляются путем изменения содержимого регистра управления сигналом.

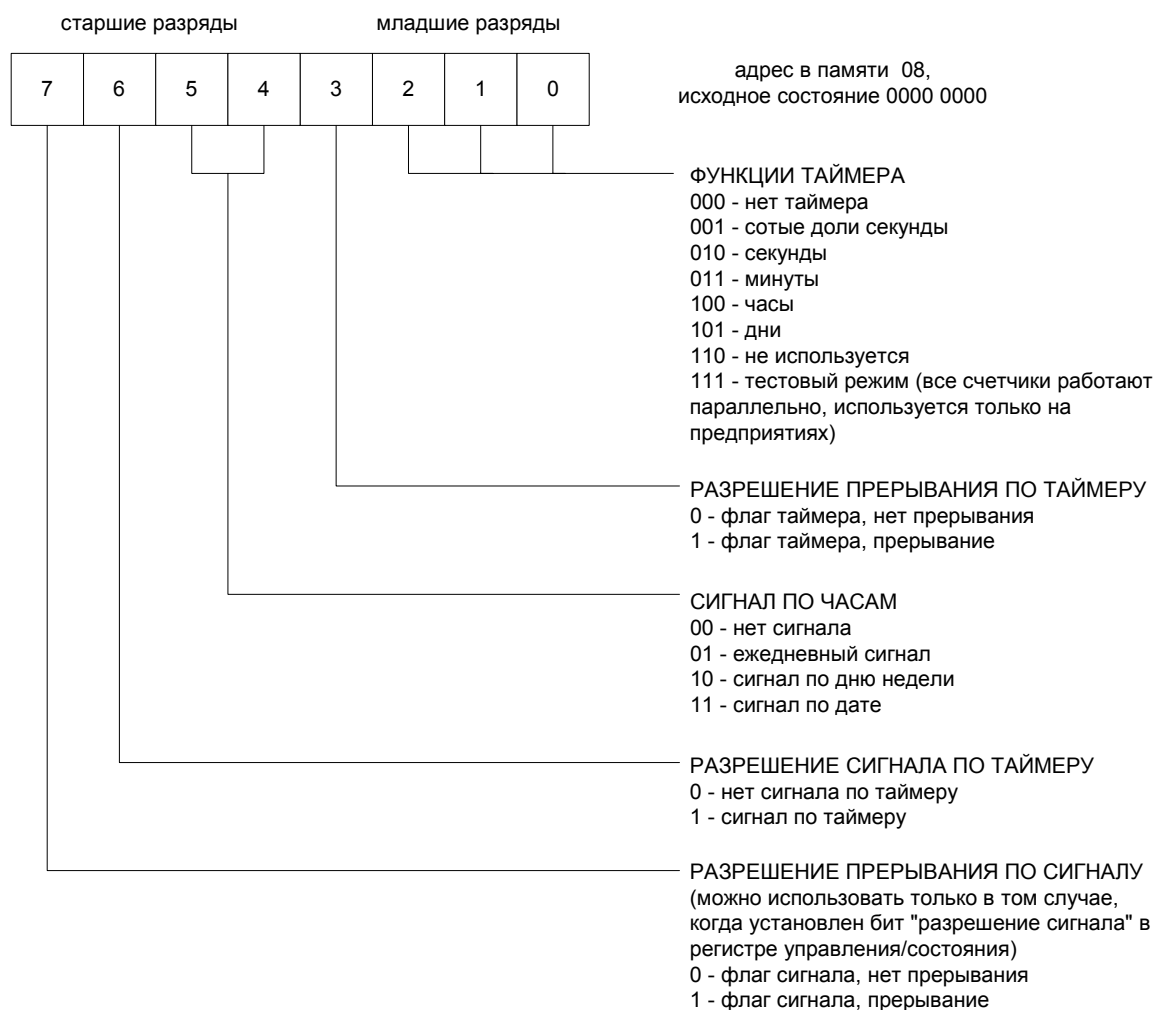


Рисунок 53. Регистр управления сигналом, режим часов.

Регистры сигналов

Все регистры сигналов размещены со смещением 08 относительно соответствующих регистров счетчиков.

Генерация сигнала происходит тогда, когда каждый бит регистра сигнала совпадает с аналогичным битом соответствующего регистра счетчика. Если речь идет о сигнале по дате, игнорируются биты года и дня недели. При генерации ежедневного сигнала игнорируются биты месяца и даты. Если выбран сигнал по дню недели, то из регистра сигнала дней недели/месяца будут выбраны соответствующие дни недели (см. рисунок ниже).

Замечание: В 12-часовом режиме времени биты 6 и 7 регистра сигнала по часам должны соответствовать таким же битам в счетчике часов.



Рисунок 54. Регистр сигнала по дням недели.

Таймер

Таймер (адрес 07) включается, если содержимое регистра управления/состояния выглядит так: XX0X X1XX. Таймер ведет отсчет от 0 (или от запрограммированного пользователем значения) до 99. При переполнении таймер устанавливается в 0. Флаг таймера (младший бит регистра управления/состояния) устанавливается при переполнении таймера. Этот флаг сбрасывается программным путем. Инвертированное значение этого флага может быть передано внешнему прерыванию путем установки бита 3 регистра управления сигналом.

Кроме того, сигнал по таймеру может быть запрограммирован установкой бита разрешения сигнала по таймеру (бит 6 регистра управления сигналом). Флаг сигнала (бит 1 регистра управления/состояния) устанавливается, когда значение таймера равно числу, указанному в регистре сигнала по таймеру (адрес 0F). Если установлен бит разрешения прерывания по сигналу (бит 6 регистра управления сигналом), то инвертированное значение флага сигнала может быть передано на внешнее прерывание.

Разрешение (точность) таймера программируется с помощью 3 младших битов регистра управления сигналом.

Режим счетчика

Режим счетчика устанавливается с помощью битов 4 и 5 регистра управления/состояния (их значения должны быть соответственно 1 и 0). Режим счетчика используется для подсчета импульсов, подаваемых на вход генератора (вывод OSC0 остается неподключенным).

Счетчик хранит до 6 цифр данных, которые располагаются в виде шестнадцатеричных чисел по адресам 1, 2 и 3. Таким образом может быть сохранено до 1 миллиона событий. Сигнал счетчика выдается в том случае, если содержимое регистра счетчика совпадает со значением, хранящимся по адресам 9, A и B, и при этом разрешен сигнал по событию (биты 4 и 5 регистра управления сигналом установлены соответственно в 0 и 1). При этом также устанавливается флаг сигнала (бит 1 регистра управления/состояния). Инвертированное значение этого флага может быть передано на вывод прерывания микросхемы (вывод 7) – для этого надо установить бит разрешения прерывания по сигналу в регистре управления сигналом.

В этом режиме таймер (адрес 07) инкрементируется при наступлении каждого первого, сотого, десятитысячного и миллионного события в зависимости от значений, установленных в битах 0, 1 и 2 регистра управления сигналом.

При наступлении всех прочих событий таймер функционирует в режиме часов.

Вывод прерывания

Условия для включения вывода INT с открытым стоком устанавливаются путем программирования регистра управления сигналом. К таким условиям относятся сигнал по часам, сигнал по таймеру, переполнение таймера, сигнал счетчика событий. Прерывание происходит в том случае, когда установлен флаг сигнала или флаг таймера и при этом разрешено соответствующее прерывание. В любом случае прерывание очищается только программным путем: очисткой флага, вызвавшего прерывание.

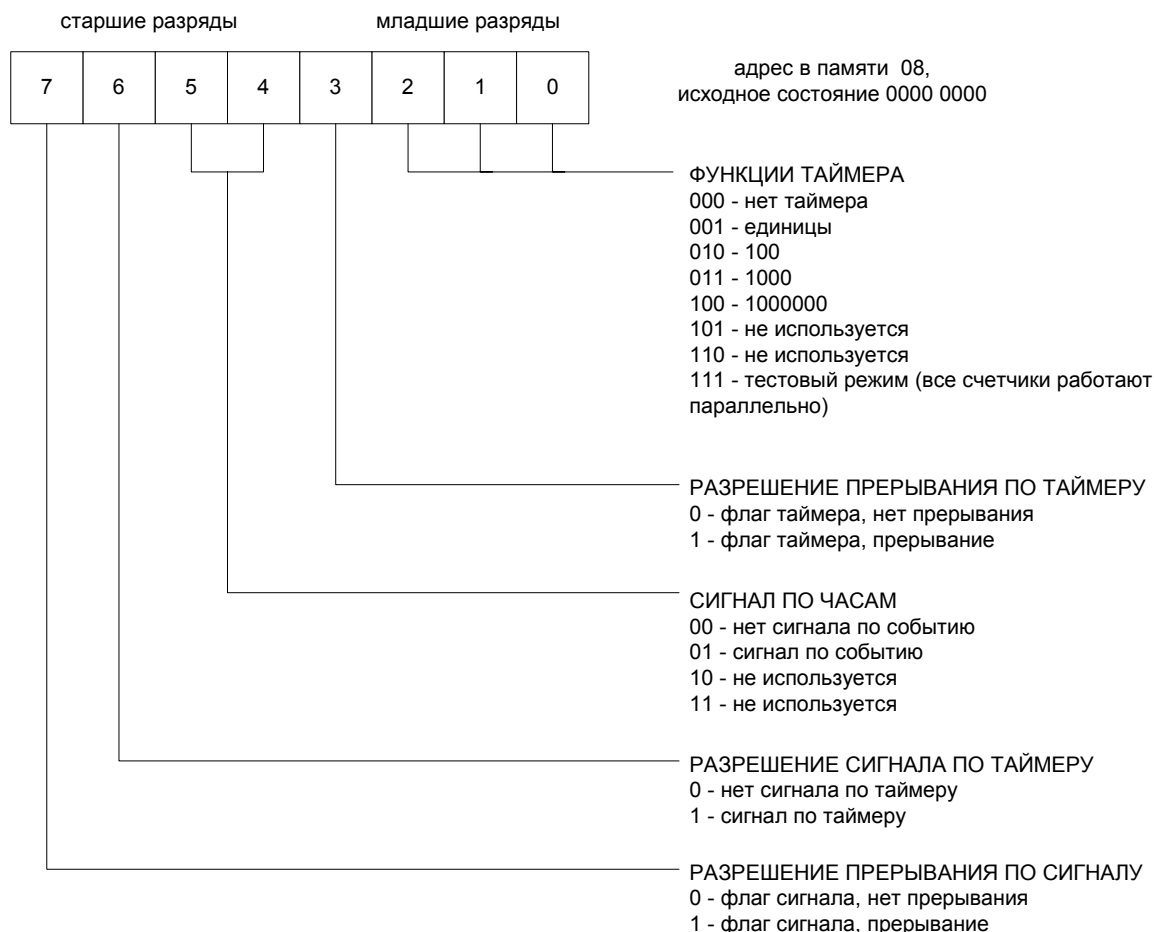


Рисунок 55. Регистр управления сигналом, режим счетчика.

В режиме счетчика, если сигнал не разрешен (то есть бит разрешения сигнала регистра управления/состояния установлен в 0), на выводе прерывания формируется сигнал с частотой 1 Гц и скважностью 2 (длительность положительного и отрицательного импульсов равна; может использоваться для калибровки). Этот режим работы устанавливается по умолчанию при включении устройства. Напряжение на выводе прерывания может превышать напряжение питания, однако оно не может быть больше 6.0 В.

Генератор и делитель

Кристалл, работающий на частоте 32.768 кГц, должен быть подключен к выводам OSCI (вывод 1) и OSCO (вывод 2). Подстроечный конденсатор между выводами OSCI и Vdd используется для настройки генератора (см. информацию о настройке частоты генератора). Сигнал 100 Гц подается с генератора для работы счетчиков часов. В режиме часов на частоте 50 Гц и в режиме счетчика событий генератор отключается, его вход переходит в состояние большого сопротивления.

Это позволяет пользователю подавать частоту обращений 50 Гц или внешний сигнал частых событий на вход OSCI.

Инициализация

При подключении питания к шине I²C сбрасываются регистр управления/состояния и все счетчики часов. Устройство начинает отсчет в режиме часов на частоте 32.768 кГц в 24-часовом формате времени и с датой и временем, установленными на 1 января в 0.00.00:00. На выводе прерывания появляется прямоугольный сигнал частотой 1 Гц и скважностью 2. Рекомендуется до загрузки в счетчики времени устанавливать флаг остановки счета (регистр управления/состояния). Загрузка некорректных значений может привести к временному сбою работы часов.

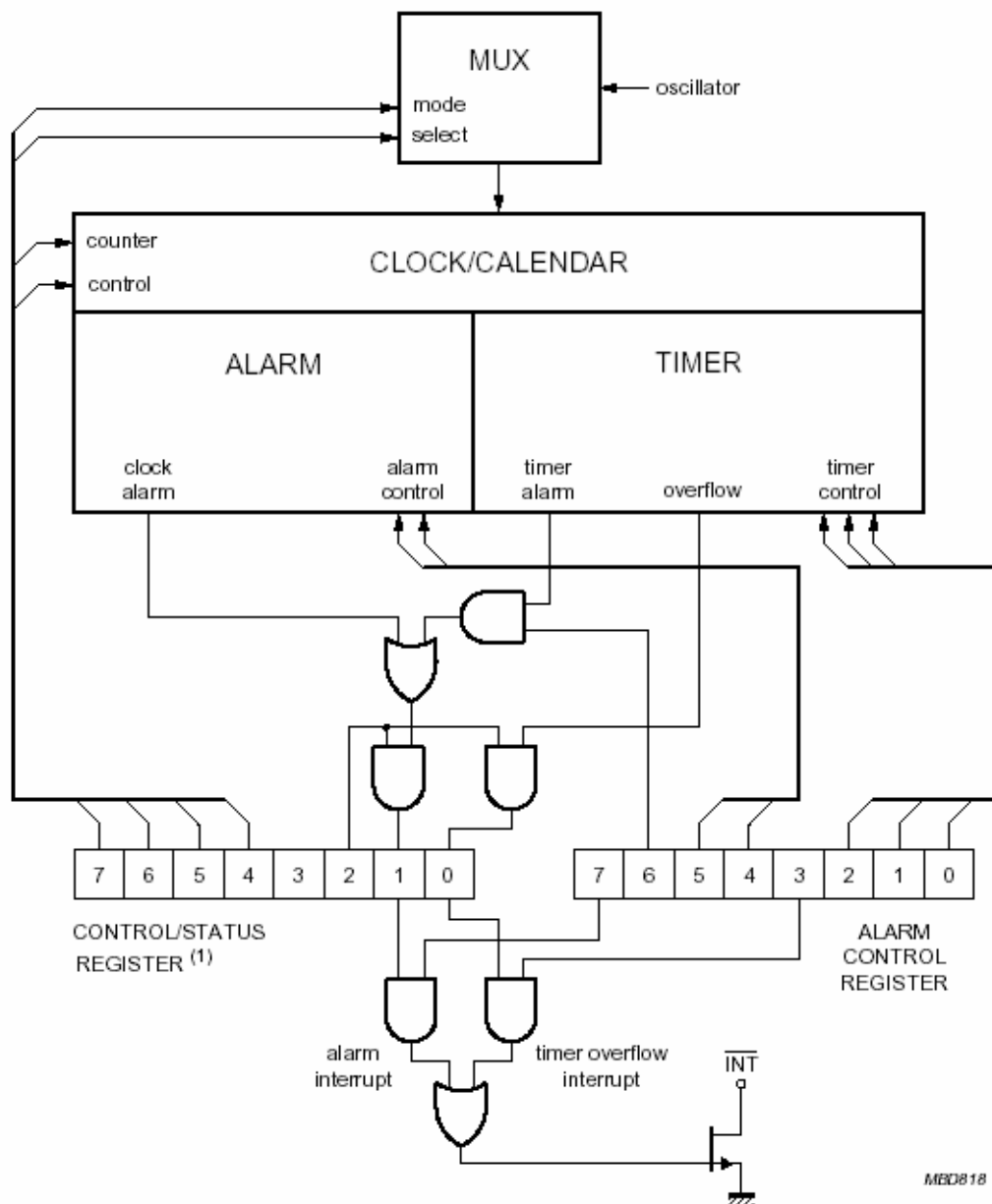


Рисунок 56. Логика работы прерываний по сигналу и по таймеру.

(1) Если бит разрешения сигнала регистра управления/состояния сброшен (то есть установлен в 0), на выводе прерывания INT может появиться сигнал частотой 1 Гц.

Обозначения:

- oscillator – генератор (mode – режим, select – выбор);
- clock/calendar – часы/календарь (counter – счетчик, control – управление);
- alarm – сигнал (clock alarm – сигнал по часам, alarm control – управление сигналом);
- timer – таймер (timer alarm – сигнал по таймеру, overflow – переполнение, timer control – управление таймером);
- CONTROL/STATUS REGISTER – регистр управления/состояния;

- ALARM CONTROL REGISTER – регистр управления сигналом;
- alarm interrupt – прерывание по сигналу;
- timer overflow interrupt – прерывание по переполнению таймера.

Характеристики шины I²C

Шина I²C – это двунаправленная шина, соединяющая между собой различные интегральные схемы или модули. Она содержит две линии: линию передачи данных (SDA) и линию синхронизации (SCL). Обе линии должны подключаться к положительному полюсу источника питания через нагрузочный резистор. Передача данных может осуществляться только в том случае, если шина не занята.

Передача бита данных

За время каждого импульса синхронизации передается один бит данных. Уровень сигнала на линии SDA должен оставаться постоянным в течение того времени, когда по линии синхронизации передается импульс высокого уровня. Любое изменение сигнала на линии данных в этот период будет воспринято как контрольный сигнал.

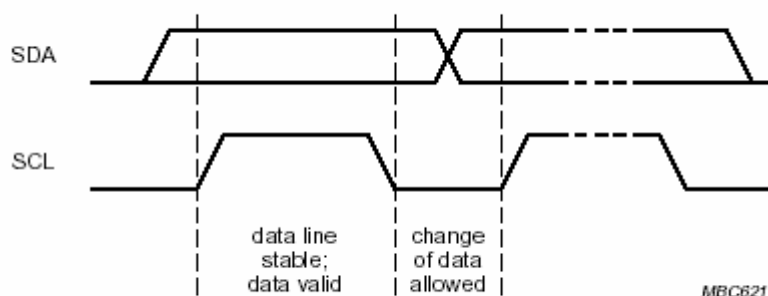


Рисунок 57. Передача бита данных.

Обозначения:

- data line stable; data valid – уровень сигнала на линии данных не изменяется; сигнал воспринимается корректно;
- change of data allowed – разрешено изменение сигнала.

Старт- и стоп-состояния

Когда шина не занята, на линиях данных и синхронизации поддерживается высокий уровень сигнала.

Старт-состояние характеризуется переходом от "1" к "0" на линии данных при наличии "1" на линии синхронизации (S).

Стоп-состояние характеризуется переходом от "0" к "1" на линии данных при наличии "1" на линии синхронизации (P).

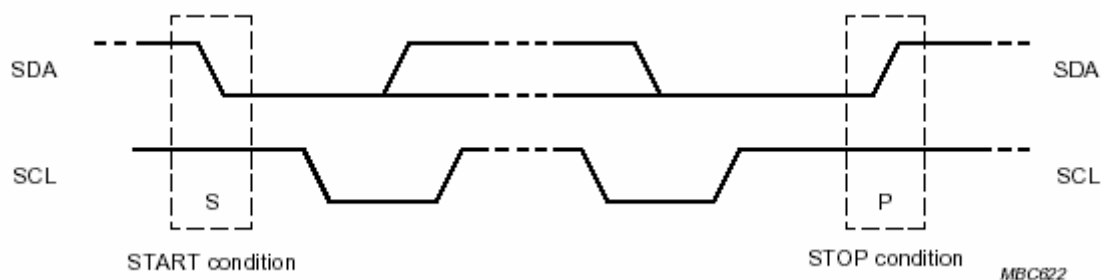


Рисунок 58. Старт- и стоп-состояние.

Конфигурация шины

Устройство, генерирующее некоторое сообщение, является передатчиком, а устройство, принимающее сообщение – приемником.

Устройство, контролирующее передачу, является главным, а устройство, контролируемое главным устройством, является подчиненным.

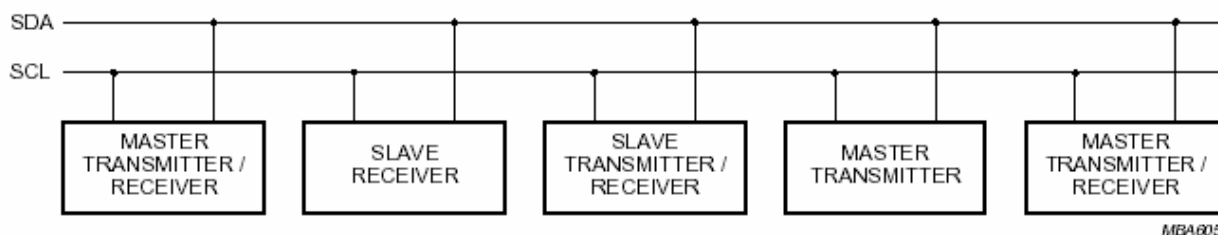


Рисунок 59. Конфигурация шины.

Обозначения:

- Master transmitter/receiver – главный передатчик / приемник.
- Slave receiver – подчиненный приемник.
- Slave transmitter/receiver – подчиненный передатчик / приемник.
- Master transmitter – главный передатчик.

Подтверждение приема сообщения

Количество байт данных, передаваемых передатчиком между стартом-состоянием и стоп-состоянием, может быть любым. После передачи каждого байта (то есть восьми бит) происходит передача бита подтверждения - сигнала уровня "1", выставяемого передатчиком на шине в течение синхронизирующего импульса. Подчиненный приемник, который был адресован, должен генерировать бит подтверждения после приема каждого байта данных. Главный приемник тоже должен генерировать бит подтверждения после приема каждого байта, переданного подчиненным передатчиком.

Устройство, подтверждающее прием (передачу), должно выставить "0" на линии SDA в течение синхронизирующего импульса. Таким образом, на линии SDA должен сохраняться "0", пока на линии SCL выставлена "1" (следует принимать во внимание временные соотношения, связанные с установкой и удержанием сигнала на линии). Главный приемник сигнализирует об окончании приема данных тем, что **не** генерирует бит подтверждения после приема последнего байта, переданного подчиненным передатчиком. В этом случае передатчик должен выдать "1" на линию SDA, тогда главное устройство сможет сгенерировать стоп-состояние.

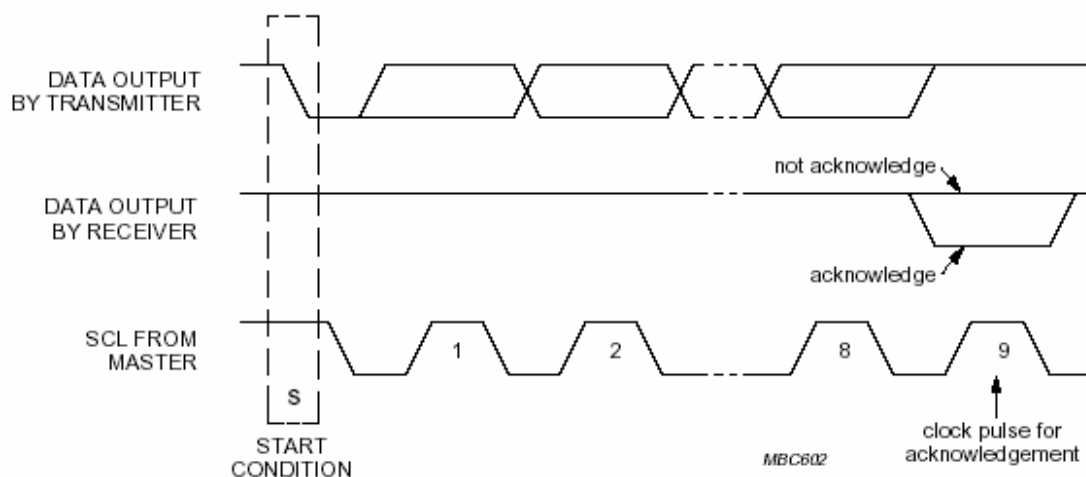


Рисунок 60. Подтверждение приема (передачи).

Обозначения:

- Data output by transmitter – данные, передаваемые передатчиком.
- Data output by receiver – данные, передаваемые приемником.
- SCL from master – линия SCL, на которую поступают данные от главного устройства.
- Start condition – старт-состояние.
- Acknowledge, not acknowledge – бит подтверждения, нет бита подтверждения.
- Clock pulse for acknowledgement – сигнал синхронизации для бита подтверждения.

Протокол передачи данных шины I²C

Адресация

До того, как на шину I²C будут переданы какие-либо данные, происходит адресация нужного устройства. Адресация осуществляется всегда при передаче первого байта после "стартовой" процедуры. Часы/календарь могут выступать в роли подчиненного приемника или подчиненного передатчика. Поэтому для них сигнал синхронизации передается только по входной линии, а данные – в обоих направлениях по линии SDA.

Адрес устройства "часы/календарь" представлен на рисунке ниже. Бит A0 относится к адресному выводу A0. При подключении этого вывода к напряжению Vdd или Vss устройство может иметь один или два различных адреса.

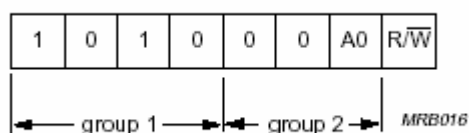


Рисунок 61. Адрес подчиненного устройства.

Циклы записи/чтения часов/календаря

Конфигурация шины I²C для циклов чтения и записи устройства PCF8583 представлена на рисунках ниже.

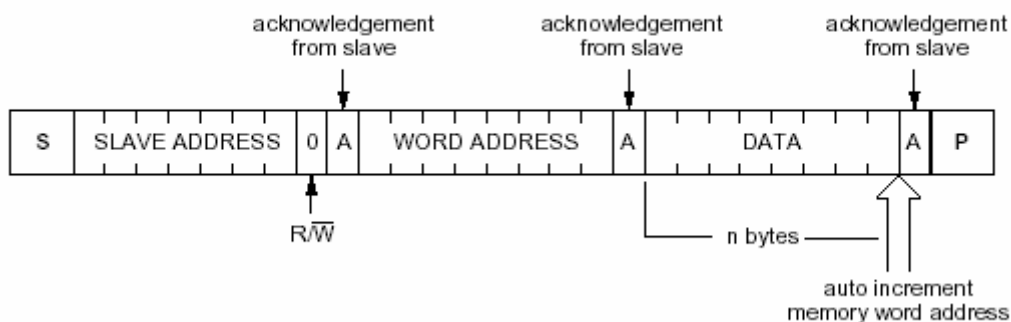


Рисунок 62. Главное устройство передает данные подчиненному приемнику (режим записи WRITE).

Обозначения:

- Slave address – адрес подчиненного устройства.
- Word address – адрес слова.
- Data – данные.
- Acknowledgement from slave – бит подтверждения, переданный подчиненным устройством.
- Auto increment memory word address – автоматическое инкрементирование (наращивание) адреса слова в памяти.

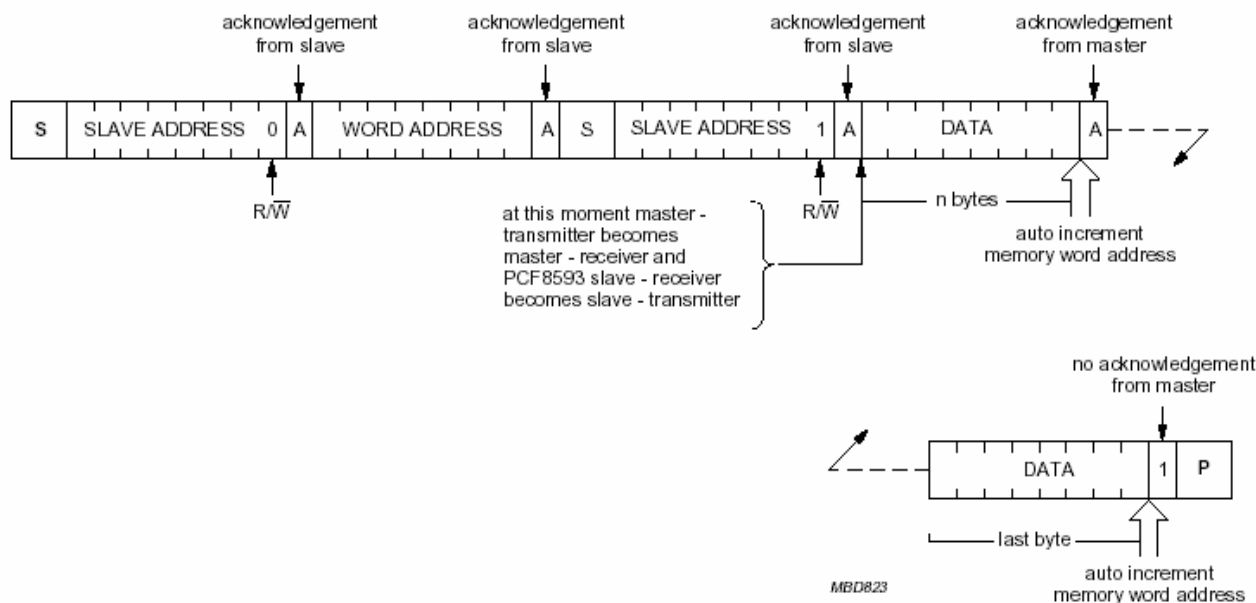


Рисунок 63. Чтение данных главным устройством после установки адреса слова (режим чтения данных).

Обозначения:

- acknowledgement from slave – бит подтверждения, переданный подчиненным устройством;
- acknowledgement from master – бит подтверждения, переданный главным устройством;
- at this moment master - transmitter becomes master - receiver and PCF8593 slave – receiver becomes slave – transmitter – в этот момент главный передатчик становится главным приемником, а устройство PCF8593, которое было подчиненным приемником, становится подчиненным передатчиком;
- auto increment memory word address – автоматическое инкрементирование (наращивание) адреса слова в памяти;
- last byte – последний байт.

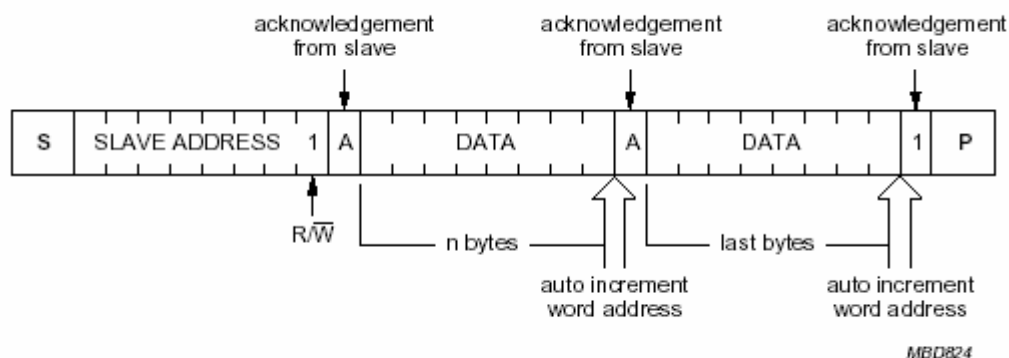


Рисунок 64. Главное устройство читает данные подчиненного устройства сразу же после передачи первого бита (режим чтения).

Обозначения:

- acknowledgement from slave – бит подтверждения, переданный подчиненным устройством;
- auto increment memory word address – автоматическое инкрементирование (наращивание) адреса слова в памяти.

Пределные величины

Таблица 40. Пределные величины (указаны в соответствии с системой AMRS - международной электротехнической комиссии).

Обозначения	Параметр	Минимум	Максимум	Единицы
V _{DD}	Напряжение питания	-0.8	+7.0	В
I _{DD}	Ток питания (вывод 8)	-	50	мА
I _{SS}	Ток питания (вывод 4)	-	50	мА
V _I	Входное напряжение	-0.8	V _{DD} + 0.8	В
I _I	Постоянный входной ток	-	10	мА
I _O	Постоянный выходной ток	-	10	мА
P _{tot}	Суммарное рассеяние мощности на один блок	-	300	мВт
P _O	Рассеяние мощности выхода	-	50	мВт
T _{amb}	Рабочая температура окружающей среды	-40	+85	°С
T _{stg}	Температура хранения	-65	+150	°С

Уход за устройством

Входы и выходы устройства защищены от воздействия электростатического заряда, однако для обеспечения полной безопасности следует соблюдать меры предосторожности, принятые при обращении с устройствами на МОП-транзисторах.

Характеристики постоянного тока

V_{DD} = 2.5...6.0 В; V_{SS} = 0 В; T_{amb} = -40...+85 °С (если не отмечено иначе).

Таблица 41. Характеристики постоянного тока.

Обозн.	Параметр	Условия	Мин.	Тип. (примеч. 1)	Макс.	Ед.
V _{DD}	Напряжение питания (рабочий режим)	Шина I ² C активна	2.5	-	6.0	В
		Шина I ² C неактивна	1.0	-	6.0	В
V _{DDosc}	Напряжение питания (на кварце генератора)	T _{amb} = 0 - 70 °С; см. примечание 2	1.0	-	6.0	В
I _{DD}	Ток питания (рабочий режим)	f _{SCL} = 100 кГц; режим часов; см. примечание 3		-	200	мкА
I _{DDO}	Ток питания (режим часов)	см. рисунок ниже f _{SCL} = 0 Гц; V _{DD} = 5 В	-	10	50	мкА
		f _{SCL} = 0 Гц; V _{DD} = 1 В	-	2	10	мкА
I _{DDR}	Сохранение данных	f _{OSCI} = 0 Hz; V _{DD} = 1 В	-	-	5	мкА
		T _{amb} = -40 - + 85 °С T _{amb} = -25 - + 70 °С	-	-	2	мкА
V _{EN}	Уровень доступ к шине I ² C	см. примечание 4	1.5	1.9	2.3	В
SDA						
V _{IL}	Входное напряжение при "0"	см. примечание 5	-0.8	-	0.3V _{DD}	В
V _{IH}	Входное напряжение при "1"	см. примечание 5	0.7V _D	-	V _{DD} + 0.8	В
I _{OL}	Выходной ток при "0"	V _{OL} = 0.4В	3	-	-	мкА
I _{LI}	Входной ток утечки	V _I = V _{DD} или V _{SS}	-1	-	+1	мА
C _i	Входная емкость	см. примечание 6	-	-	7	пкФ
A0; OSCI						
I _{LI}	Входной ток утечки	V _I = V _{DD} или V _{SS}	-250	-	+250	нА
INT						
I _{OL}	Выходной ток при "0"	V _{OL} = 0.4 В	3	-	-	мА
I _{LI}	Входной ток утечки	V _I = V _{DD} или V _{SS}	-1	-	+1	мкА
SCL						

C _i	Входная емкость	см. примечание 6	-	-	7	пкФ
I _{LI}	Входной ток утечки	V _I = V _{DD} или V _{SS}	-1	-	+1	мкА

Примечания.

1. Типичные значения были получены при температуре T_{amb} = 25 °С.
2. При включении питания V_{DD} должно превышать 1.5 В и оставаться таким до тех пор, пока не установится стабильный режим работы генератора.
3. Режим счетчика событий: ток питания зависит от входной частоты
4. Логика шины I²C отключается, если V_{DD} < V_{EN}.
5. Если значение напряжения становится выше или ниже напряжений питания V_{DD} или V_{SS}, может наблюдаться входной ток. Он не должен превышать ±0.5 мА.
6. Протестировано на образце.

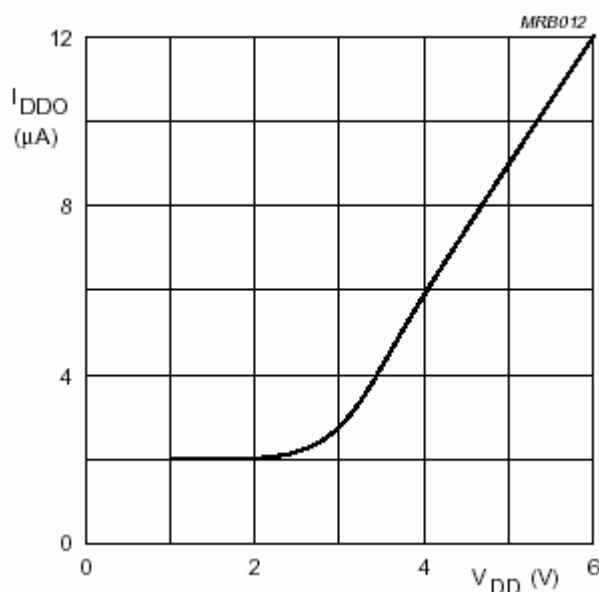


Рисунок 65. Типичные значения тока питания в режиме часов как функция зависимости от напряжения питания. f_{SCL} = 32 кГц; T_{amb} = 25 °С.

Характеристики переменного тока

V_{DD} = 2.5...6.0 В; V_{SS} = 0 В; T_{amb} = -40...+85 °С (если не отмечено иначе).

Таблица 42. Характеристики переменного тока.

Обозн.	Параметр	Условия	Мин.	Тип.	Макс.	Ед.
Генератор						
C _{osc}	Емкость генератора		-	40	-	пФ
Δf _{osc}	Частотная устойчивость генератора	для ΔV _{DD} = 100 мВ;	-	2 × 10 ⁻⁷	-	
f _i	Входная частота	T _{amb} = 25 °С; V _{DD} = 1.5 В	-	-	1	МГц
Параметры кварца (f = 32.768 кГц)						
R _s	Последовательное сопротивление		-	-	40	кОм
C _L	Емкость параллельной загрузки		-	10	-	пФ
C _T	Емкость подстроечного конденсатора		5	-	25	пФ
Синхронизация шины I ² C (см примечания 2 и 3, а также рисунок ниже)						
f _{SCL}	Частота синхронизации SCL		-	-	100	кГц
t _{SP}	Допустимая ширина		-	-	100	нс

	импульса на шине					
t_{BUF}	Время простоя шины		4.7	-	-	мкс
$t_{SU;STA}$	Время установки старт-состояния		4.7	-	-	мкс
$t_{HD;STA}$	Время удержания старт-состояния		4.0	-	-	мкс
t_{LOW}	Время "0" на SCL		4.7	-	-	мкс
t_{HIGH}	Время "1" на SCL		4.0	-	-	мкс
t_r	Время нарастания сигнала на SDA и SCL		-	-	1.0	мкс
t_f	Время спада сигнала на SDA и SCL		-	-	0.3	мкс
$t_{SU;DAT}$	Время установки импульса данных		250	-	-	нс
$t_{HD;DAT}$	Время удержания импульса данных		0	-	-	нс
$t_{VD;DAT}$	Время выдачи "0" для корректной передачи данных		-	-	3.4	мкс
$t_{SU;STO}$	Время удержания стоп-состояния		4.0	-	-	мкс

Примечания

1. Только в режиме счетчика событий.
2. Все временные характеристики корректны для напряжения питания и температуры окружающей среды, не выходящих за установленные пределы, и относятся к инжекционным логическим схемам с вертикальной геометрией, имеющим перепад входного напряжения от V_{SS} до V_{DD} .
3. Подробное описание спецификации шины I²C с приложениями представлено в брошюре "Шина I²C и ее использование" ("The I²C-bus and how to use it").

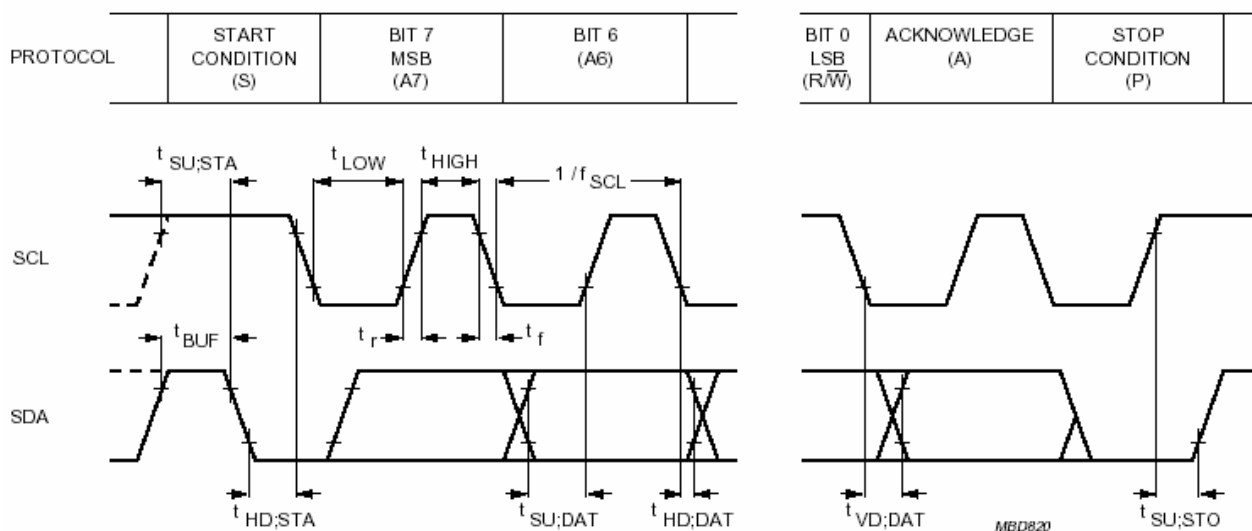


Рисунок 66. Временная диаграмма работы шины I²C; время нарастания и спада относится к логическим инжекционным схемам с вертикальной геометрией.

Информация по использованию

Настройка частоты кварца

Способ 1: С помощью конденсатора постоянной емкости на OSCI

Для вычисления среднего значения емкости, необходимого в данной схеме применения, может быть использован конденсатор постоянной емкости. Частоту лучше всего рассчитывать через сигнал частотой 1 Гц после подключения вывода прерываний (вывод 7). Допустимое отклонение частоты зависит от устойчивости кристалла, конденсатора и сопряжения устройство-устройство (в среднем $\pm 5 \times 10^{-6}$). Допустимо отклонение ± 5 минут за год.

Способ 2: С помощью подстроечного конденсатора на OSC1

При использовании функций сигнала (через шину I²C) для быстрой настройки подстроечного конденсатора на выводе прерываний может быть сгенерирован сигнал с частотой, превышающей 1 Гц.

Порядок действий:

- Включение в сеть.
- Инициализация (функций сигнала).

Регулярные действия:

- Установить часы на время T и сигнал на время T + dT.
- В момент времени T + dT (прерывание) повторить процедуру.

Способ 3: Непосредственные измерения на выходе OSC (с учетом сопротивления пробника)

Адрес подчиненного устройства PCF8583 имеет постоянную комбинацию 1010 в группе 1.

Программное обеспечение стенда SDK-1.1

Резидентный загрузчик HEX202

Резидентный загрузчик HEX202 располагается во Flash-памяти ADuC812 начиная с адреса 0100h. Он обеспечивает начальную инициализацию системы, загрузку программ в HEX-формате в память SDK-1.1 и передачу им управления.

Начальная инициализация. При включении питания или передаче управления на ячейку с адресом 0 происходит повторная инициализация всех регистров специального назначения их значениями по умолчанию. Это сделано для того, чтобы при случайной передаче управления на ячейку с адресом 0 вследствие возможной ошибки в пользовательской программе не происходило сбоя системы, а сама система вела себя так же, как при включении питания. Эта же процедура повторяется непосредственно перед передачей управления загруженной программе. В случае успешной инициализации на ЖКИ на мгновение выводится надпись «SDK-1.1, 2001 ©LMT Ltd» и на резонатор выдается короткий сигнал.

Загрузка программ в память SDK-1.1. После процедуры инициализации системы последовательный канал настраивается в режим 9600 бит/сек, 8 бит данных, 1 стоп-бит, без контроля четности и в него выдается строка «HEX202-XX», где XX – номер версии загрузчика. Далее с интервалом примерно в 200 мс выдается символ ‘.’ и ожидается появление символа со стороны инструментальной системы на РС. При появлении символа, если это первый символ строки в HEX-формате, то есть двоеточие (‘:’), выдача символа ‘.’ прекращается и производится прием остальной части HEX-строки. После завершения приема очередной HEX-строки вычисляется ее контрольная сумма. Если она не совпадает с принятой, то в последовательный канал выдается символ ‘-’, сигнализирующий об ошибке приема. В противном случае выдается ‘+’ и принятая строка обрабатывается в соответствии с указанной в ней командой (запись данных в память, конец блока или передача управления). Далее, если не было команды передачи управления, вывод в последовательный канал символа ‘.’ возобновляется и ожидается следующая HEX-строка.

Передача управления загруженной программе. Передача управления происходит по приему HEX-строки вида :02AAAA060000SS<cr>, где AAAA – это HEX-адрес, по которому необходимо передать управление, SS – контрольная сумма HEX-строки, <cr> - символ возврата каретки. Такая строка должна быть добавлена в конец каждого HEX-файла, загружаемого в SDK-1.1. Для этого в поставляемых с SDK-1.1 инструментальных системах есть команда `addhexstart` (см. соответствующие разделы).

Важное замечание. В процессе своей работы HEX202 использует область статической памяти в диапазоне адресов F000h-FFFFh, что следует учитывать при написании программ для стенда. Расположение части кода или статических данных по этим адресам может привести к некорректной работе загрузчика и неработоспособности загружаемой программы.

Инструментальная система для MS-DOS (T167B)

Инструментальная система T167B призвана решать следующие задачи:

1. Преобразование HEX- и BIN- файлов.
2. Передача загрузочных модулей различных форматов в целевую систему с протоколами разного уровня сложности.
3. Получение информации из целевой системы.
4. Обеспечение элементарных операций с последовательным каналом (прием и передача байта, эмуляция терминала, настройка скорости).
5. Обеспечение быстрой адаптации к целевой системе.

Управляющие клавиши.

- CTRL+BREAK – аварийный выход; в большинстве случаев приводит к корректному завершению работы T167B при зависаниях.
- Alt+X – выход в DOS.
- Up, Down – перелистывание команд; командная строка в T167B имеет историю, записываемую в файл.

Основные команды. Система T167B представляет собой набор средств для работы с различными системами (а не только с SDK-1.1). Она имеет множество команд, добавлявшихся для взаимодействия с очередными разработками ООО «ЛМТ». В данном руководстве описаны лишь основные команды, необходимые для работы с SDK-1.1.

Обозначение команд дано в виде (a, b, c -> d, e), где a, b, c – *числовые* параметры (их число не должно быть обязательно равно трем), “->” соответствует команде, d и e – *строковые (символьные)* параметры. Следует учесть, что все параметры разделяются пробелами, а не запятыми.

Пример обозначения: PRIMER (a,b,c->d) filename.ext

Команда PRIMER имеет 3 числовых параметра, разделяемых пробелом и указываемых перед командой, и один строковый, указываемый после команды. Пример вызова команды PRIMER:

```
Ok 0x10 12 145 primer test.txt
Ok
```

Здесь: a = 0x10, b = 12, c = 145, filename.ext = test.txt

Таблица 43. Перечень основных команд системы T167B.

Команда	Действие
OPENCHANNEL	Открытие COM-порта.
OPENCHANNELRTS	Открытие COM-порта с установкой сигнала RTS.
CLOSECHANNEL	Закрытие COM-порта.
TERM	Включение эмулятора терминала.
+ECHO	Включение режима копирования консольного вывода в файл echo.txt.
-ECHO	Выключение режима копирования консольного вывода в файл echo.txt.
LOADHEX+	HEX-загрузка.
ADDHEXSTART	Добавление стартового адреса в конец HEX-файла.
BYE	Выход из T167B.
LFIL	Интерпретация командного файла.

OPENCHANNEL (com,baud->)

Открытие COM-порта:

Аргументы: com – номер COM-порта (1, 2, 3, 4); baud – делитель частоты для получения нужной скорости обмена в порту.

Наиболее часто используемые делители:

- 24: 4800 бит/сек;
- 12: 9600 бит/сек;
- 6: 19200 бит/сек;
- 3: 38400 бит/сек;
- 2: 57600 бит/сек;
- 1: 115200 бит/сек.

Пример:

```
2 12 openchannel – открытие COM2 на скорости 9600.
```

Команда `openchannelrts` аналогична `openchannel`. Отличие состоит в том, что `openchannelrts` устанавливает сигнал RTS (сигнал интерфейса RS-232). Этот сигнал используется для питания гальванически развязанного последовательного порта на целевой системе.

CLOSECHANNEL (->)

Заккрытие ранее открытого COM-порта.

Пример:

```
closechannel
```

TERM (w->)

Включение эмулятора терминала:

- 0 – бинарный;
- 1 – HEX;
- 2 – десятичный.

Пример:

```
1 term
```

+ECHO (->)

-ECHO (->)

Включение/выключение режима копирования консольного вывода в файл `echo.txt`.

Пример:

```
+echo
```

LOADHEX+ (-> filename.hex)

Загрузка HEX-файла в целевую систему (SDK-1.1) по протоколу загрузчика HEX202. Этот протокол предполагает последовательную пересылку строк из HEX-файла `filename.hex`. После отправки очередной строки ожидается подтверждение со стороны HEX202 в виде символа '+' или запрос на повторную отсылку в виде '-'. Далее ожидается символ '.' и производится либо отсылка следующей строки (если были приняты '+' и '.'), либо повторная отсылка данной строки. Если снова были приняты '-' и '.', то попытка повторяется во второй раз и т.д. – всего 9 раз, после чего производится аварийный выход.

Отсылка HEX-файла производится только при наличии периодической индикации работоспособности HEX202, проявляющейся в отсылке символа '.'.

Необходимо заметить, что перед отсылкой HEX-файла, сгенерированного в какой-либо среде разработки (например, в `µVision`), необходимо добавить в его конец стартовый адрес (то есть адрес в памяти RAM, на который передается управление после загрузки в SDK-1.1) командой `addhexstart`.

Пример:

```
loadhex+ myfile.hex
```

ADDHEXSTART (addr,seg->) filename.hex

Добавление в конец файла `filename.hex` строки, которая нужна для передачи управления загрузчиком HEX202 по адресу `addr` после загрузки в целевую систему (SDK-1.1). Поле `seg` необходимо указывать, но в данный момент оно не используется.

Пример:

```
0x9000 0x0 addrhexstart myfile.hex
```

BYE (->)

Выход из T167B.

Пример:

```
bye
```

LFILE (-> filename.ext)

Интерпретация командного файла filename.ext. Файл представляет собой набор строк текста, содержащих команды T167B в том же виде, в котором они представлены в командной строке T167B.

Пример:

```
lfile myfile.ini
```

Инструментальная система для Win 9x/NT

Инструментальная система T2 для Windows 9x/NT с точки зрения команд для работы с SDK-1.1 является аналогом системы T167B для MS-DOS (за исключением нескольких команд).

Таблица 44. Перечень основных команд системы T2.

Команда	Действие
OPENCHANNEL	Открытие COM-порта с установкой сигнала RTS.
OPENCOM1, OPENCOM2	Открытие портов COM1 или COM2 с установкой сигнала RTS.
CLOSECHANNEL	Закрытие COM-порта.
TERM	Включение эмулятора терминала.
+ECHO	Включение режима копирования консольного вывода в файл echo.txt.
-ECHO	Выключение режима копирования консольного вывода в файл echo.txt.
LOADHEX	HEX-загрузка.
ADDHEXSTART	Добавление стартового адреса в конец HEX-файла.
BYE	Выход из T2.
LFILE	Интерпретация командного файла.

Описание и синтаксис большинства команд совпадают с их аналогами в T167B. Ниже дано описание только тех команд, которые не присутствуют в T167B или имеют другой синтаксис.

OPENCHANNEL (baud -> com)

Открытие последовательного порта на заданной скорости. Числовой параметр baud определяет скорость в бодах, например, 19200. Параметр com может иметь два значения: «com1» или «com2».

Пример:

```
9600 openchannel com1
```

OPENCOM1, OPENCOM2 (->)

Открытие COM1 или COM2 на скорости 9600 бод.

Пример:

```
opencom1
```

TERM (w->)

Команда аналогична команде term в T167B, за исключением того, что параметр w может принимать лишь значения 0 или 1.

Пример:

```
0 term
```

LOADHEX (-> filename.hex)

Аналогична команде loadhex+ системы T167B.

Программатор Flash для ADuC812 (DL)

Программное обеспечение (ПО) программирования Flash-памяти ADuC812 – это модифицированная версия программного обеспечения ADuC Downloader Version 2.07, свободно распространяемого фирмой Analog Devices. Изменение, внесенное в программу, заключалось в соответствующем конфигурировании COM-порта для обеспечения питания оптически развязанного от остальной части платы порта SDK-1.1.

При запуске программа dl.exe полностью стирает Flash-память ADuC812 и записывает туда образ программы, содержащейся в HEX-файле, имя которого указывается в командной строке. Протокол загрузки HEX-файла во Flash-память микроконтроллера детально описан в документе MicroConverter™ Technical Note – uC004, прилагаемом к настоящему руководству в виде pdf-файла.

Формат командной строки. Командная строка dl.exe имеет следующие параметры:

- /C:X, где X – номер COM-порта (1 или 2). Если параметр не указан, по умолчанию принимается 1.
- /F:n.n, где n.n – частота, на которой работает микроконтроллер, в МГц (по умолчанию 11.0592). Так как микроконтроллер ADuC812, установленный на стенде SDK-1.1, работает от кварца 11.0592 МГц, этот параметр можно не указывать.
- /D – указывает программе не стирать память данных (RAM). Этот параметр не является обязательным.
- /R или /R:XXXX, где XXXX – адрес в HEX-формате. Указывает загрузчику запустить передаваемую программу либо с определенного адреса, либо с адреса по умолчанию (FF00h). Этот параметр необходимо пропустить.
- *filename.hex* – единственный обязательный параметр: имя файла с программой в HEX-формате.

Замечание. Для загрузки программы во Flash-память необходимо установить переключатель JP1 на плате SDK-1.1 (находится слева от клавиатуры примерно на уровне клавиши «1»). После этого необходимо нажать кнопку сброса или включить питание (если стенд был отключен) и запустить программу dl.exe, указав необходимые параметры в командной строке. Затем необходимо снять переключатель и нажать кнопку сброса – записанная во Flash-память программа начнет работать.

Набор средств тестирования стенда SDK-1.1

В комплект поставки SDK-1.1 входит программа SDKtest, обеспечивающая тестирование основных компонент стенда и заодно поясняющая работу с устройствами в SDK-1.1. Вместе с загрузочным модулем поставляются исходные тексты программы на языке Си. Для запуска программы достаточно запустить одну из прилагаемых инструментальных систем (T167B или T2) и с их помощью загрузить HEX-образ программы в память стенда. Затем необходимо перейти в режим эмуляции терминала с параметром 0 (см. описание команды term в соответствующих разделах).

При запуске программа на короткое время зажжет сигнальные светодиоды и выдаст в последовательный канал текстовое меню. В дальнейшем необходимо следовать инструкциям программы, передаваемым по последовательному каналу. Программа выводит часть результатов на ЖКИ.

Простейшая программа на языке Си

В этом разделе приведена простейшая программа на языке Си для стенда SDK-1.1, инициализирующая последовательный канал на скорости 9600 бод и выдающая в него строчку «SDK1.1». Текст программы:

```
#include "ADuC812.h"    // Включение в текст описания регистров
                        // специального назначения ADuC812

void main(void)
{
    TH1 = 0xFD;        // Скорость 9600
    TMOD = 0x20;       // Таймер 1 в режиме autoreload
    TCON=0x40;         // Запуск таймера 1
    SCON=0x50;         // 8 bit UART, разрешение приема
    PCON&=0x7F;       // Отключение дублирования скорости, установленной в TH1
    EA=0;              // Запрещение прерываний

    TI=0;              // Обнуление флага завершения посылки
    SBUF='S';          // Инициация посылки символа «S»
    while(!TI);       // Ожидание завершения посылки
    TI=0;
    SBUF='D';
    while(!TI);
    TI=0;
    SBUF='K';
    while(!TI);
    TI=0;
    SBUF='1';
    while(!TI);
    TI=0;
    SBUF='.';
    while(!TI);
    TI=0;
    SBUF='1';
    while(!TI);

    while(1);         // «Завершение» программы.
}
```

Необходимо обратить внимание на последний оператор в теле функции main(). Бесконечный цикл `while(1);` играет роль оператора завершения программы. Так как SDK-1.1 не находится под управлением операционной системы, то простой выход из пользовательской программы приведет к неконтролируемой выборке команд микроконтроллером из памяти, что может вызвать нежелательные последствия и даже привести к выходу стенда из строя. Поэтому рекомендуется все программы либо «завершать» бесконечным циклом, либо строить их таким образом, чтобы они работали по бесконечному алгоритму.

Для трансляции программы необходим компилятор C51 фирмы Keil Software. Соответствующий командный файл для получения загрузочного HEX-модуля прилагается к исходным текстам.

Инструментальные средства фирмы Keil Software

Keil Software поддерживает все стадии разработки приложения: создание исходного файла на С или Ассемблере, трансляцию, исправление ошибок, линкование объектных файлов, тестирование приложения.

В пакете Keil Software содержатся следующие средства разработки для микроконтроллера 8051:

- C51 – компилятор С;
- Макроассемблер А51;
- Динамический загрузчик/компоновщик BL51;
- Конвертер объектных файлов OC51;
- Конвертер объектных и HEX-файлов OH51;
- Менеджер библиотек LIB51;
- Симулятор dScope-51 (для Windows);
- Отладчик/компилятор μ Vision/51 (для Windows);
- Операционная система реального времени (Real-Time Operating System - RTX).

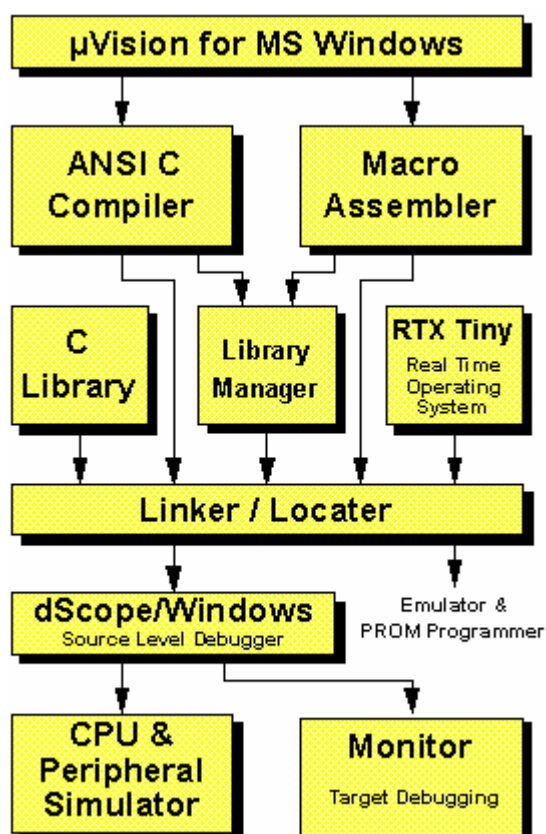


Рисунок 67

Компилятор языка Си фирмы Keil Software

Кросс-компилятор языка С предназначен для создания приложений для процессоров семейства 8051 на языке программирования С. Компилятор C51 поддерживает стандарт ANSI C, разработан специально для 8051 семейства и позволяет создавать программы на

языке C, сохраняя эффективность и скорость оптимизации Ассемблера. Расширения, включенные в инструментальные средства Keil, обеспечивают полный доступ к ресурсам микроконтроллеров 8051. Для локализации ошибок полезно воспользоваться выходными данными C-компилятора или средствами PC-Link. Отличительные черты компилятора:

- Девять основных типов данных, включая быструю 32-разрядную IEEE-арифметику с плавающей точкой.
- Разнообразие селекторов для различных областей памяти: code, data, bdata, idata, xdata и pdata, которые позволяют осуществить эффективный доступ к различным областям памяти и сгенерировать компактный код.

Таблица 45. Селекторы для различных областей памяти.

Селектор	Область памяти
data	128 байт во встроенном ОЗУ – непосредственная байтовая адресация
bit	128 бит во встроенном ОЗУ – непосредственная битовая адресация
bdata	128 бит/16 байт во встроенном ОЗУ – непосредственная битовая/байтовая адресация
idata	256 байт во встроенном ОЗУ – непосредственная адресация
pdata	256 байт в страничной внешней памяти XDATA
xdata	64 Кбайт памяти XDATA
code	64 Кбайт памяти CODE

- Область размещения переменных и функций и соответственно время доступа к переменной в памяти определяют модели памяти. Выбор модели зависит от размера области памяти: Small - 128 байт, Compact - 256 байт, Large - 64 Кбайт.
- Разнообразие общих и специальных указателей данных для различных областей памяти:
 - Основные указатели (generic pointers) позволяют получить доступ ко всем областям памяти 8051 (code, data, idata, xdata и pdata). Эти указатели используются в функциях типа memsrcu и printf для обеспечения максимальной совместимости.
 - Специальные указатели (memory-specific pointers) указывают на определенную область памяти 8051 и позволяют осуществить эффективный доступ к различным областям и сгенерировать компактный код.

Например, область data занимает только 128 байт и указатель data размещается в одном байте. При использовании модели Small с селектором data программа выполняется быстрее и генерируемый код короче, однако для редко используемых переменных вполне подходит модель Large с селектором xdata. Сочетание моделей достигается путем явного указания селектора при описании переменной.

- Наиболее эффективное управление прерываниями при написании функций прерываний на C.
- Прямое управление банками регистров и полное их использование, битовая адресация данных. Для доступа к регистрам специального назначения и их отдельным битам используются ключевые слова sfr, sfr16 и sbit. Доступ к отдельным битам побитно-адресуемых регистров осуществляется с помощью ключевого слова bit.
- Доступ на C к регистрам специальных функций для всех МК 8051, простой интерфейс с ассемблером.
- Полная отладочная информация для последующей высокоуровневой отладки с помощью эмулятора.
- Использование инструкций AJMP и ACALL.
- Встроенный интерфейс для операционной системы реального времени RTX51.
- Сложная синтаксическая проверка и подробные предупреждающие сообщения.
- Более сотни библиотечных функций ANSI C, учитывающих особенности различных членов семейства 8051.

- Эффективность и скорость С-программы за счет использования ряда методов оптимизации: регистровой оптимизации, общей оптимизации кода и специальной оптимизации для 8051.
- В поставку C51 входят 6 оптимизированных библиотек, предоставляющих пользователю свыше 100 библиотечных функций, помимо стандартного набора из ANSI C.

chkfloat_	atan2	isalnum	memccpy	sinh	strchr
crol	atof	isalpha	memchr	sprintf	strpbrk
cror	atoi	isctrl	memcmp	sqrt	strpos
_getkey	atol	isdigit	memcpy	srand	strspn
irol	cabs	isgraph	memmove	sscanf	tan
iror	calloc	islower	memset	streat	tanh
lrol	ceil	isprint	modf	strchr	toascii
lror	cos	ispunct	pow	strcmp	toint
nop	cosh	isspace	printf	strcpy	tolower
testbit	exp	isupper	putchar	strcspn	toupper
_tolower	fabs	isxdigit	puts	strlen	ungetchar
_toupper	floor	labs	rand	strncat	va_arg
abs	free	log	realloc	strncmp	va_end
acos	getchar	log10	scanf	strncpy	va_start
asin	gets	longjmp	setjmp	strpbrk	vprintf
atan	init_mempool	malloc	sin	strpos	vsprintf

Расширение языка С до C51

Компилятор C51 очень гибок. Он поддерживает стандарт ANSI и все аспекты программирования на С, описанные этим стандартом. Все дополнения к стандарту предназначены для поддержки контроллера 8051. Дополнения касаются следующих понятий:

- Типы данных;
- Типы памяти;
- Модели памяти;
- Указатели;
- Реентерабельные функции;
- Функции обработки прерываний;
- Операционные системы реального времени;
- Поддержка PL/M и A51.

Все эти дополнения будут описаны ниже.

Типы данных

Компилятор C51 поддерживает все типы данных, приведенные ниже в таблице. Скалярные переменные могут быть объединены в структуры, объединения и массивы. За некоторым исключением (что указано ниже) доступ к значениям переменных может быть получен с помощью указателей.

Таблица 46. Типы данных.

Типы данных	Биты	Байты	Область значений
bit*	1		от 0 до 1
signed char	8	1	от -128 до +127
unsigned char	8	1	от 0 до 255
enum	16	2	от -32768 до +32767
signed short	16	2	от -32768 до +32767
unsigned short	16	2	от 0 до 65535

signed int	16	2	от -32768 до +32767
unsigned int	16	2	от 0 до 65535
signed long	32	4	от -2147483648 до 2147483647
unsigned long	32	4	от 0 до 4294967295
float	32	4	от ±1.175494E-38 до ±3.402823E+38
sbit*	1	0	от 0 до 1
sfr*	8	1	от 0 до 255
sfr16*	16	2	от 0 до 65535

* Типы *bit*, *sbit*, *sfr* и *sfr16* являются специфичными для процессора 8051 и компилятора C51. Они не описаны стандартом ANSI, поэтому к ним нельзя обращаться через указатели.

Типы данных *sbit*, *sfr* и *sfr16* используются для обращения к специальным регистрам процессора 8051. Например, строка `sfr P0 = 0x80`; объявляет переменную P0 и назначает ей адрес специального регистра 0x80. Это адрес порта P0. Компилятор C51 производит автоматическое преобразование типов в случае, если результат относится к другому типу. Кроме того, можно выполнить принудительное приведение типов. В процессе приведения расширение знака к переменным знаковых типов добавляется автоматически.

Модификаторы памяти

Компилятор C51 поддерживает контроллеры типа 8051 и обеспечивает доступ ко всем областям памяти контроллера. Для каждой переменной можно точно указать область ее размещения в памяти.

Таблица 47

Модификаторы памяти	Описание
code	Память программ (64 Кб); выполняется следующий код: <code>MOVC @A+DPTR</code> .
data	Внутренняя память данных с прямой адресацией; самая быстрая работа с переменными (128 байт).
idata	Внутренняя память данных с косвенной адресацией; доступ ко всему адресному пространству (256 байт).
bdata	Бит-адресуемая внутренняя память данных; возможность обращаться как к битам, так и к байтам (16 байт).
xdata	Внешняя память данных (64 Kbytes); выполняется код: <code>MOVX @DPTR</code> .
pdata	Внешняя память данных со страничной организацией (256 байт); выполняется код: <code>MOVX @Rn</code> .

Обращение к внутренней памяти данных происходит гораздо быстрее, чем к внешней. Поэтому переменные, которые используются чаще других, следует размещать во внутренней памяти, а остальные – во внешней.

Применяя модификаторы памяти при объявлении переменной, можно указать, где именно она будет размещена.

Можно включить сведения о модели памяти в объявление переменной (точно так же, как атрибуты `signed` и `unsigned`), например:

```
char data var1;
char code text[] = "ENTER PARAMETER:";
unsigned long xdata array[100];
float idata x,y,z;
unsigned int pdata dimension;
unsigned char xdata vector[10][4][4];
char bdata flags;
```

Если в объявлении переменной модификатор памяти не указан, выбирается модель памяти, установленная по умолчанию. Аргументы функции и переменные класса памяти `auto`, которые не могут быть размещены в регистрах, также хранятся в области памяти, установленной по умолчанию.

Модель памяти, выбираемая в качестве модели по умолчанию, устанавливается с помощью директив компилятора SMALL, COMPACT и LARGE.

Модели памяти

Вы можете назначить модель памяти для тех аргументов функций, автопеременных и переменных, в объявлении которых не указана соответствующая модель. Это делается с помощью директив SMALL, COMPACT и LARGE. Явное задание модели отменяет использование модели памяти по умолчанию.

Таблица 48. Модели памяти.

Название	Описание
SMALL	В этой модели все переменные по умолчанию размещаются во внутренней памяти данных. Достигается тот же эффект, как при явном использовании модификатора data. При такой модели обращение к переменным оказывается очень эффективным. Но все объекты данных и стек должны размещаться во внутренней памяти. Размер стека имеет решающее значение, так как пространство, занимаемое стеком, зависит от глубины вложенности различных функций. Обычно, если компоновщик BL51 сконфигурирован для оверлейной загрузки переменных во внутреннюю память данных, лучше всего использовать малую модель памяти (small).
COMPACT	Если используется компактная модель памяти, все переменные по умолчанию загружаются во внешнюю память данных – точно так же, как при явном задании модификатора памяти rdata. В этой памяти может быть размещено до 256 байтов. Ограничения появляются вследствие использования косвенной адресации, когда обращение происходит через регистры R0 и R1. Данная модель памяти не так эффективна, как малая, и обращение к переменным происходит медленнее. Но компактная модель все же быстрее, чем большая. Старший бит адреса обычно устанавливается через порт 2, причем делается это вручную программистом.
LARGE	При использовании большой модели памяти все переменные по умолчанию размещаются во внешней памяти данных. Можно также явно указать модель памяти с помощью модификатора xdata. Для адресации используется указатель DPTR. Обращение к памяти через этот указатель не является эффективным, особенно если переменная имеет длину 2 или более байт. При использовании данной модели памяти код получается длиннее, чем при малой или компактной модели.

Примечание:

Почти всегда следует использовать модель памяти SMALL: в этом случае вы получите самый быстрый, компактный и наиболее эффективный код. Однако у вас есть возможность явно указать нужную модель памяти. Модели, использующие внешнюю память, стоит использовать только в том случае, если ваше приложение никаким образом не может работать при модели памяти SMALL.

Указатели

Компилятор C51 поддерживает объявление указателей с использованием символа звездочки (*). Указатели можно использовать для выполнения всех доступных в стандартном C операций.

Вследствие уникальности архитектуры контроллера 8051 и его производных компилятор C51 поддерживает 2 вида указателей: память-зависимые и нетипизированные.

Нетипизированные указатели

Нетипизированные указатели объявляются точно так же, как указатели в стандартном C, например:

```
char *s;          /* string ptr */
int *numptr;     /* int ptr */
```

```
long *state; /* long ptr */
```

Нетипизированные указатели имеют размер 3 байта. В первом байте указывается модель памяти, во втором – старший байт смещения, в третьем – младший байт смещения.

Нетипизированные указатели используются для обращения к любым переменным независимо от их размещения в памяти контроллера. Многие библиотечные функции используют эти указатели, при этом им совершенно неважно, в какой именно области памяти они размещаются.

Память-зависимые указатели

В объявления память-зависимых указателей всегда включается модификатор памяти. Обращение всегда происходит к указанной области памяти, например:

```
char data *str; /* ptr to string in data */
int xdata *numtab; /* ptr to int(s) in xdata */
long code *powtab; /* ptr to long(s) in code */
```

Поскольку модель памяти определяется во время компиляции, типизированным указателям не нужен байт, в котором указывается модель. Типизированные указатели могут иметь размер в 1 байт (указатели `idata`, `data`, `bdata`, и `pdata`) или в 2 байта (указатели `code` и `xdata`).

Сравнение память-зависимых и нетипизированных указателей

Можно существенно ускорить выполнение кода путем использования типизированных указателей. Приведенные ниже примеры программ показывают различия в размерах кода и данных, а также времени выполнения для двух видов указателей.

Таблица 49. Сравнение память-зависимых и нетипизированных указателей.

Описание	Указатель <code>Idata</code>	Указатель <code>Xdata</code>	Нетипизированный указатель
Пример программы	<code>char idata *ip;</code> <code>char val;</code> <code>val = *ip;</code>	<code>char xdata *xp;</code> <code>char val;</code> <code>val = *xp;</code>	<code>char *p;</code> <code>char val;</code> <code>val = *p;</code>
Код, сгенерированный компилятором 8051	<code>MOV R0,ip</code> <code>MOV val,@R0</code>	<code>MOV DPL,xp +1</code> <code>MOV DPH,xp</code> <code>MOV A,@DPTR</code> <code>MOV val,A</code>	<code>MOV R1,p + 2</code> <code>MOV R2,p + 1</code> <code>MOV R3,p</code> <code>CALL CLDPTR</code>
Размер указателя	1 байт	2 байта	3 байта
Размер кода	4 байта	9 байт	11 байт кода + библиот.
Время выполнения	4 цикла	7 циклов	13 циклов

Реентерабельные функции

Реентерабельные функции могут быть вызваны одновременно разными процессами. При выполнении такой функции другой процесс может прервать ее выполнение и начать выполнять ту же функцию. В большинстве случаев функции C51 не могут быть вызваны рекурсивно или повторно до окончания выполнения. Причина этого в том, что аргументы функций и локальные переменные хранятся в строго определенных областях памяти. Однако атрибут `reentrant` позволяет объявить функцию как реентерабельную, которая, может быть вызвана рекурсивно, например:

```
int calc (char i, int b) reentrant
{
    int x;
    x = table [i];
    return (x * b);
}
```

Реентерабельные функции могут вызываться рекурсивно, а также *одновременно* двумя и более процессами. Подобные функции часто используются в приложениях, работающих в режиме реального времени, или в тех случаях, когда функция вызывается как из кода, так и по прерыванию.

Стек каждой реентерабельной функции размещается во внутренней или внешней памяти в зависимости от модели памяти.

Примечание:

Добавление атрибута reentrant при объявлении функции позволит сделать реентерабельными только те участки программ, которые могут быть вызваны повторно или рекурсивно. При этом вся программа не становится реентерабельной. Если вы объявите реентерабельной целую программу, она займет больше места в памяти.

Функции – обработчики прерываний

Компилятор C51 предоставляет возможность вызова функций при возникновении прерываний. Это дает возможность написания на языке C собственных обработчиков прерываний. Однако следует соблюдать осторожность в выборе номера прерывания и банка регистров. Компилятор автоматически генерирует вектор прерывания, а также код входа в обработчик и выхода из него. Атрибут interrupt, включенный в объявление функции, указывает на то, что данная функция обрабатывает прерывание. Кроме того, можно указать банк регистров для данного прерывания с помощью атрибута using.

```
unsigned int interruptcnt;
unsigned char second;

void timer0 (void) interrupt 1 using 2
{
    if (++interruptcnt == 4000)
    {
        /* count to 4000 */

        second++;          /* second counter */
        interruptcnt = 0; /* clear int counter */
    }
}
```

Передача параметров

Компилятор C51 размещает в регистрах до 3 аргументов функций. Это значительно повышает системную производительность, поскольку аргументы не требуется сохранять в памяти и считывать оттуда. Передачей параметров можно управлять с помощью специальных директив – REGPARMS и NOREGPARMS.

Ниже приведен список регистров, используемых для размещения аргументов разных типов.

Таблица 50. Регистры, используемые для размещения аргументов разных типов.

Количество аргументов	char, 1-байтный указатель	int, 2-байтный указатель	long, float	Нетипизированный указатель
1	R7	R6 & R7	R4 - R7	R1 - R3
2	R5	R4 & R5		
3	R3	R2 & R3		

Если в данный момент нет свободных регистров или аргументов слишком много, то для этих аргументов используются фиксированные области памяти.

Значения, возвращаемые функцией

Регистры микроконтроллера всегда используются для значений, возвращаемых функциями. Ниже приведена таблица типов возвращаемых данных и соответствующих им регистров.

Таблица 51. Типы возвращаемых данных и регистры.

Тип возвращаемого значения	Регистры	Описание
bit	Флаг переноса	
char, unsigned char, 1-byte pointer	R7	
int, unsigned int, 2-byte pointer	R6 & R7	MSB в R6, LSB в R7
long, unsigned long	R4 - R7	MSB в R4, LSB в R7
float	R4 - R7	32-битный формат IEEE
generic pointer	R1 - R3	Тип памяти в R3, MSB R2, LSB, R1

Оптимальное выделение регистров

Компилятор C51 выделяет для регистровых переменных до 7 регистров процессора (в зависимости от контекста программы). Компилятору известны все регистры, содержимое которых изменилось в процессе выполнения функции. Компоновщик / загрузчик генерирует один файл на весь проект, содержащий информацию обо всех регистрах, измененных внешними функциями. В результате компилятор знает об изменениях в каждом регистре и может оптимально распределить регистры среди всех функций.

Добавление ассемблерного кода

К процедурам, написанным на ассемблере, можно легко обращаться из кода на языке C (и наоборот). Параметры функций передаются через регистры контроллера или, в случае использования директивы `NOREGPARMS`, путем загрузки их в фиксированные области памяти. Значения, возвращаемые функциями, всегда передаются через регистры.

Чтобы сгенерировать не объектный файл, а код, который может быть обработан ассемблером A51, нужно использовать директиву компилятора C51 `SRC`. Например, приведенный ниже текст на C,

```
unsigned int asmfunc1 (unsigned int arg)
{
    return (1 + arg);
}
```

откомпилированный с использованием директивы `SRC`, создает следующий код на ассемблере:

```
?PR?_asmfunc1?ASM1 SEGMENT CODE
PUBLIC _asmfunc1
RSEG ?PR?_asmfunc1?ASM1
USING 0
_asmfunc1:
;---- Variable 'arg?00' assigned to Register 'R6/R7' ----
MOV A,R7 ; load LSB of the int
ADD A,#01H ; add 1
MOV R7,A ; put it back into R7
CLR A
ADDC A,R6 ; add carry & R6
MOV R6,A
?C0001:
RET ; return result in R6/R7
```

Чтобы включить инструкции ассемблера в программу на C, следует использовать директивы препроцессора `#pragma asm` и `#pragma endasm`.

Работа с языком PL/M-51

PL/M-51 – разработанный Intel популярный язык программирования, который во многом схож с языком С. Программа, написанная на С, может легко взаимодействовать с функциями PL/M-51. Непосредственно в программе на С можно объявить функции PL/M-51, включив модификатор `alien`. Все глобальные переменные, объявленные в модуле на PL/M-51, доступны программе на С. Например:

```
extern alien char plm_func (int, char);
```

Поскольку компилятор PL/M-51 и инструментальные средства Keil Software генерируют объектный файл в формате OMF51, внешние символические ссылки являются разрешенными.

Оптимизация кода

Компилятор C51 обладает мощными оптимизирующими свойствами. Это означает, что компилятор сам предпринимает определенные действия к тому, чтобы повысить эффективность генерируемого кода: уменьшить его объем или увеличить скорость выполнения. Компилятор производит оценку полученного кода с точки зрения скорости работы, эффективности и размера.

Компилятор C51 обеспечивает несколько уровней оптимизации. Каждый из уровней оптимизации обладает возможностями нижележащих уровней. Ниже приведен список всех уровней.

- Общая оптимизация:
 - "Склеивание констант": несколько константных значений, включенных в одно выражение или участвующих в одной адресной операции, рассматриваются как одна константа.
 - Оптимизация переходов: если это помогает повысить эффективность работы программы, на место переходов подставляются реальные адреса.
 - Удаление пассивных участков программы: из программы удаляются все участки, к которым ни разу не происходит обращения.
 - Регистровые переменные: автоматические переменные и аргументы функций размещаются по возможности в регистрах, место в памяти данных для этих переменных не резервируется.
 - Передача параметров через регистры: аргументы функций (не более трех аргументов) передаются через регистры.
 - Оптимизация глобальных выражений: идентичные выражения или операции с адресами, встречающиеся в функции неоднократно, запоминаются и выполняются по возможности только один раз.
- Методы оптимизации, специфичные для контроллеров 8051:
 - Оптимизация по методу "глазка": сложные операции замещаются простыми, если это поможет уменьшить затраты памяти или сократить время выполнения программы.
 - Оптимизация доступа: константы и переменные рассчитываются, и их значения подставляются при выполнении операций.
 - Перекрытие данных: данные и сегменты функций считаются перекрываемыми - компоновщик BL51 замещает их другими данными и сегментами.
 - Оптимизация конструкций Case/Switch: в зависимости от числа, порядка следования и размещения выполнение операторов switch может быть оптимизировано с помощью таблицы или строки переходов.

- Директивы для оптимизации кода:
 - OPTIMIZE(SIZE). Большинство операций языка C замещаются подпрограммами. Однако при этом несколько падает скорость работы программы.
 - OPTIMIZE(SPEED). Многие операции языка C рассматриваются как встраиваемые (in-line, на их место в каждом случае подставляется код). При этом возрастает скорость работы программы, но увеличивается размер кода.
 - NOAREGS. Компилятор C51 не использует абсолютный доступ к регистрам. Поэтому код программы не зависит от выбранного банка регистров.
 - NOREGPARMS. Передача параметров производится через сегмент данных, а не через специальные регистры. Код, использующий директиву #pragma, совместим с более ранними версиями компиляторов C51 и PL/M-51 и ассемблера ASM-51.
- Оптимизация глобальных регистров.
Компилятор C51 обеспечивает оптимизацию глобальных регистров. В приведенном ниже примере код, сгенерированный компилятором C51 версии 5.0 (использующим оптимизацию глобальных регистров), сравнивается с кодом, который сгенерирован компилятором версии 3.4. В первом случае компилятору C известны регистры, используемые внешними функциями. Регистры, содержимое которых не изменяется внешними функциями, используются для регистровых переменных. Полученный код использует меньше данных, занимает меньше памяти и быстрее выполняется. Ниже приведены две внешние функции *input* и *output*, использующие всего несколько регистров.

Таблица 52. Сравнение кода с оптимизацией глобальных регистров и без нее.

Оптимизация глобальных регистров	Без оптимизации
<pre>main () { unsigned char i; unsigned char a; while (1) { i = input (); /* get number of values */</pre>	
<pre>?C0001: LCALL input ;- 'i' assigned to 'R6' - MOV R6,AR7</pre>	<pre>?C0001: LCALL input MOV DPTR,#i MOV A,R7 MOV @DPTR,A</pre>
<pre>do { a = input (); /* get input value */</pre>	
<pre>?C0005: LCALL input ;- 'a' assigned to 'R7' - MOV R5,AR7</pre>	<pre>?C0005: LCALL input MOV DPTR,#a MOV A,R7 MOVX @DPTR,A</pre>
<pre>output (a); /* output value */</pre>	
<pre>LCALL _output</pre>	<pre>LCALL _output</pre>
<pre>} while (--i); /* decrement values */</pre>	

DJNZ R6,?C0005 MOV DPTR,#i	MOVX A,@DPTR DEC A MOVX @DPTR,A JNZ ?C0005
}	
SJMP ?C0001	SJMP ?C0001
}	
RET	RET
Размер кода: 18 байт	Размер кода: 30 байт

Отладка программ

Компилятор C51 использует для объектных файлов разработанный Intel формат OMF51 и сохраняет текстовые данные. Кроме того, компилятор может включить в код и другую информацию: имена переменных, функций, количество строк и т.д., что позволяет производить полноценные анализ и отладку кода в эмуляторе dScope или других Intel-совместимых эмуляторах. При использовании управляющей директивы OBJECTTEXTEND в объектный файл включается дополнительная информация о типах переменных, благодаря чему в различных эмуляторах будут отображены соответствующие переменные и структуры. Следует поинтересоваться у поставщика, поддерживает ли данный эмулятор формат OMF51, и, в частности, поддерживает ли он объектные модули Keil.

Файлы библиотек

Компилятор C51 использует семь подключаемых во время компиляции библиотек, удовлетворяющих различным функциональным требованиям.

Таблица 53. Файлы библиотек.

Файл библиотеки	Описание
C51S.LIB	Предназначена для малой модели памяти и не включает операции над числами с плавающей запятой.
C51FPS.LIB	Предназначена для малой модели памяти; производит операции над числами с плавающей запятой.
C51C.LIB	Предназначена для компактной модели памяти и не включает операции над числами с плавающей запятой.
C51FPC.LIB	Предназначена для компактной модели памяти; включает операции над числами с плавающей запятой.
C51L.LIB	Предназначена для большой модели памяти и не включает операции над числами с плавающей запятой.
C51FPL.LIB	Предназначена для большой модели памяти; производит операции над числами с плавающей запятой.
80C751.LIB	Библиотека для контроллера Philips 8xC751 и его производных.

Исходные коды используются выполняющими аппаратно-зависимый ввод-вывод библиотечными модулями и размещаются в каталоге \C51\LIB. С помощью этих файлов можно использовать библиотеки для выполнения операций ввода-вывода при работе с любыми периферийными устройствами.

Встроенные библиотеки

Библиотеки, включаемые компилятором, содержат массу функций, которые выполняются как встроенные (подставляемые). Другим функциям приходится использовать инструкции ACALL или LCALL, чтобы вызвать библиотечные функции. Встроенные функции с этой целью генерируют подставляемый (in-line) код, который более эффективен и выполняется быстрее.

Таблица 54. Встроенные функции.

Встроенные функции	Описание
crol	Циклический сдвиг символа влево.
cror	Циклический сдвиг символа вправо.
irol	Циклический сдвиг целого числа влево.
iror	Циклический сдвиг целого числа вправо.
lrol	Циклический сдвиг длинного целого числа влево.
lror	Циклический сдвиг длинного целого числа вправо.
nop	Нет операции (инструкция 8051 NOP).
testbit	Проверить и очистить бит (инструкция 8051 JBS).

Командная строка

Вызов:

C51 sourcefile [directives]

C51 @commandfile

При этом:

- sourcefile** Имя исходного файла на языке C.
- commandfile** Имя файла, содержащего командную строку компилятора, включающую sourcefile и directives. Можно использовать командный файл для более простой компоновки исходного файла или в случаях, когда все директивы не помещаются в командной строке.
- directives** Параметры директив, описываемые ниже.

Таблица 55

Элементы управления C51	Описание
<u>C</u> ODE	Включает ассемблерный код в файл листинга.
<u>C</u> OMPACT	Установка компактной модели памяти.
<u>D</u> EBUG	Включает отладочную информацию в объектный файл.
<u>D</u> EFINE	Определяет препроцессорные идентификаторы в командной строке.
<u>F</u> LOAT <u>F</u> UZZY	Назначает число бит округления при операциях с плавающей запятой.
<u>I</u> NT <u>VE</u> CTOR	Назначает интервал для векторов прерываний.
<u>I</u> NT <u>VE</u> CTOR(<i>n</i>), <u>NO</u> INT <u>VE</u> CTOR	Назначает смещение для таблицы прерываний (<i>n</i>) или исключает вектора прерываний из объектного файла.
<u>L</u> ARGE	Устанавливает большую модель памяти.
<u>L</u> IST <u>I</u> NG <u>C</u> ON <u>U</u> SE	Включает содержимое подключаемого файла в файл листинга.
<u>M</u> AX <u>A</u> RGS(<i>n</i>)	Назначает число байт, зарезервированных для списка аргументов переменной длины.
<u>M</u> OD517	Разрешает поддержку дополнительного оборудования Siemens 80C517 и производных.
<u>M</u> ODDP2	Разрешает поддержку дополнительного оборудования Dallas Semiconductor 80C320/520/530 и AMD 80C521.
<u>N</u> O <u>A</u> MAKE	Исключает AutoMAKE-информацию из объектного файла.
<u>N</u> O <u>A</u> REGS	Отключает абсолютную адресацию регистра с использованием инструкции ARn.

<u>NOCOND</u>	Исключает из файла листинга код, к которому не осуществляется перехода по условию.
<u>NOEXTEND</u>	Запрещает расширения и процессы 8051 и использует только конструкции, описанные в стандарте ANSI C.
<u>NOINTPROMOTE</u>	Запрещает правила приведения целых чисел по стандарту ANSI.
<u>NOREGPparms</u>	Запрещает передачу параметров через регистры.
<u>OBJECT [(filename)], NOOBJECT</u>	Разрешает или запрещает создание объектного файла. Объектный файл сохраняется под указанным именем (filename).
<u>OBJECTEXTEND †</u>	Включает дополнительную информацию о типе переменных в объектный файл.
<u>OPTIMIZE</u>	Назначает уровень оптимизации, выполняемой компилятором.
<u>ORDER</u>	Размещает переменные в памяти в том же порядке, в каком они объявлены в исходном файле.
<u>PAGELLENGTH(n)</u>	Назначает максимальное число строк на каждой странице файла листинга.
<u>PAGEWIDTH(n)</u>	Назначает максимальное число символов в каждой строке файла листинга.
<u>PREPRINT [(filename)]</u>	Создает препроцессорный файл листинга со всеми дополнительными макросами. Препроцессорный файл листинга сохраняется под указанным именем (filename).
<u>PRINT [(filename)], NOPRINT</u>	Разрешает или запрещает создание файла листинга. Файл листинга сохраняется под указанным именем (filename).
<u>REGFILE(filename)</u>	Назначает имя файла, в котором будет содержаться информация об использовании регистров.
<u>REGISTERBANK</u>	Назначает банк регистров для использования функций исходного файла.
<u>ROM({SMALL COMPACT LARGE})</u>	Контролирует генерирование инструкций AJMP и ACALL.
<u>SMALL</u>	Выбирает малую модель памяти.
<u>SRC</u>	Создает ассемблерный файл вместо объектного модуля.
<u>SYMBOLS</u>	Включает список символических ссылок, используемых в файле листинга.
<u>WARNINGLEVEL(n)</u>	Контролирует тип и уровень генерируемых предупреждений.

Пример файла листинга

```

C51 COMPILER V5.02, SAMPLE 07/01/95 08:00:00 PAGE 1

DOS C51 COMPILER V5.02, COMPILATION OF MODULE SAMPLE
OBJECT MODULE PLACED IN SAMPLE.OBJ
COMPILER INVOKED BY: C:\C51\BIN\C51.EXE SAMPLE.C CODE

stmt level source
1 #include <reg51.h> /* SFR definitions for 8051 */
2 #include <stdio.h> /* standard i/o definitions */
3 #include <ctype.h> /* defs for char conversion */
4
5 #define EOT 0x1A /* Control+Z signals EOT */
6
7 void main (void) {
8 1 unsigned char c;
9 1
10 1 /* setup serial port hwd (2400 Baud @12 MHz) */
11 1 SCON = 0x52; /* SCON */
12 1 TMOD = 0x20; /* TMOD */
13 1 TCON = 0x69; /* TCON */
14 1 TH1 = 0xF3; /* TH1 */
15 1
16 1 while ((c = getchar ()) != EOF) {
17 2 putchar (toupper (c));
18 2 }
19 1 P0 = 0; /* clear Output Port to signal ready */
20 1 }

ASSEMBLY LISTING OF GENERATED OBJECT CODE

```

В файле листинга указываются номера страниц, а также дата и время компиляции. В файле также содержится информация о компиляторе и объектном файле.

В листинг включаются номера строк и номер уровня для каждого блока, заключенного в фигурные скобки { и }.

В файл включаются сообщения об ошибках и предупреждения.

Опция компилятора CODE включает в файл ассемблерный код, в котором

```
; FUNCTION main (BEGIN)
; SOURCE LINE # 7
; SOURCE LINE # 11
0000 759852 MOV SCON,#052H
; SOURCE LINE # 12
0003 758920 MOV TMOD,#020H
; SOURCE LINE # 13
0006 758869 MOV TCON,#069H
; SOURCE LINE # 14
0009 758DF3 MOV TH1,#0F3H
000C ?C0001:
; SOURCE LINE # 16
000C 120000 E LCALL getchar
000F 8F00 R MOV c,R7
0011 EF MOV A,R7
0012 F4 CPL A
0013 6008 JZ ?C0002
; SOURCE LINE # 17
0015 120000 E LCALL _toupper
0018 120000 E LCALL _putchar
; SOURCE LINE # 18
001B 80EF SJMP ?C0001
001D ?C0002:
; SOURCE LINE # 19
001D E4 CLR A
001E F580 MOV P0,A
; SOURCE LINE # 20
0020 22 RET
; FUNCTION main (END)
```

также указаны номера строк исходной программы, соответствующие коду.

```
MODULE INFORMATION: STATIC OVERLAYABLE
CODE SIZE          =      33 ----
CONSTANT SIZE      = ---- ----
XDATA SIZE         = ---- ----
PDATA SIZE         = ---- ----
DATA SIZE          = ---- 1
IDATA SIZE         = ---- ----
BIT SIZE           = ---- ----
END OF MODULE INFORMATION.
```

В файл включена информация об используемых областях памяти.

C51 COMPILATION COMPLETE. 0 WARNING(S), 0 ERROR(S)

В последних строках файла указано общее количество ошибок и предупреждений.

Макроассемблер A51

Ассемблер фирмы Keil A51 специально разработан для семейства микроконтроллеров 8051. Ассемблер в основном применяется при написании фрагментов программ, наиболее критичных к скорости, размеру кода и возможностям аппаратного управления. В ассемблер включен макроязык, использование которого ускоряет разработку и экономит общее время проектирования. Макроязык позволяет также осуществлять доступ ко всем ресурсам микроконтроллеров с использованием символьных обозначений регистров.

Отладка написанного на Ассемблере модуля осуществляется с помощью отладчика/симулятора dScore или внутрисхемного эмулятора. Специальная директива DEBUG обеспечивает включение в загрузочный модуль полной информации для символьной отладки. В дополнение к объектному файлу Ассемблер генерирует list-файл, состав которого задается соответствующими директивами для включения таблиц символов и переменных, информации о перекрестных ссылках и другой отладочной информации.

Ассемблер поддерживает все разновидности соответствующего семейства микроконтроллеров. В ассемблер встроено стандартное описание регистров специальных функций SFR. В то же время специальная директива NOMOD позволяет аннулировать эти определения путем подключения файла заголовка для конкретного микроконтроллера. Ассемблер содержит файлы заголовков для всех основных членов семейств микроконтроллеров, на основе которых можно легко создать файл для любого члена

семейства. Заметим также, что при задании директивы SRC С-компилятор Keil генерирует из кода на языке С ассемблерный код.

Обзор функций

Ассемблер A51 преобразовывает исходный ассемблерный код в перемещаемый объектный модуль. Если используется директива **DEBUG**, объектный файл содержит весь код, который можно отлаживать в dScore или встроенном эмуляторе. Помимо объектного модуля ассемблер генерирует таблицу имен и словарь перекрестных ссылок. Ассемблер A51 полностью поддерживает исходный код на языке Intel ASM-51.

Конфигурация

Ассемблер A51 поддерживает все микроконтроллеры семейства 8051. Набор регистров специального назначения (SFR) является стандартным. Однако директива **NOMOD51** позволяет подавлять эти определения с помощью специальных подключаемых файлов. В поставку ассемблера A51 входят файлы для микроконтроллеров 8051, 8051Fx, 8051GB, 8052, 80152, 80451, 80452, 80515, 80C517, 80C515A, 80C517A, 8x552, 8xC592, 8xCL781, 8xCL410 и 80C320. Можно легко создавать подключаемые файлы для других контроллеров семейства 8051.

Командная строка

Вызов:

**A51 sourcefile [directives]
A51 @commandfile**

При этом:

sourcefile Имя исходного файла на ассемблере.
commandfile Имя файла, содержащего командную строку ассемблера, включающую sourcefile и directives. Вы можете использовать командный файл для более простой компоновки исходного файла или в том случае, когда все директивы не помещаются в командной строке.
directives Параметры, описываемые ниже.

Таблица 56

Элементы управления A51	Описание
<u>DATE</u> (date)	Размещает строку данных (date) в заголовке (максимум 9 символов).
<u>DEBUG</u>	Включает информацию по отладке в объектный файл.
<u>ERRORPRINT</u> [(filename)]	Выводит сообщение об ошибке в файл с указанным именем (filename).
<u>INCLUDE</u> (filename)	Включает содержимое указанного файла (filename) в файл на ассемблере.
<u>MACRO</u>	Объявляет стандартный макрос.
<u>MPL</u>	Объявляет макрос формата Intel.
<u>NOAMAKE</u>	Исключает AutoMAKE-информацию из объектного файла.
<u>NOCOND</u>	Исключает некомпонуемый код из листинга файла.
<u>NOGEN</u>	Отключает выполнение дополнительных (нестандартных) макрокоманд из листинга файла.
<u>NOLINES</u>	Исключает номера строк из объектного файла.
<u>NOLIST</u>	Исключает ассемблерный код исходника из файла листинга.
<u>NOMACRO</u>	Отключает обработку макроса.
<u>NOMOD51</u>	Отключает регистры специального назначения, специфичные для 8051.

<u>NOSYMBOLS</u>	Исключает таблицу имен из листинга файла.
<u>NOSYMLIST</u>	Исключает таблицу объявлений из листинга файла.
<u>OBJECT</u> [(<i>filename</i>)], <u>NOOBJECT</u>	Запрещает или разрешает создание объектного файла. Объектный файл сохраняется под указанным именем (<i>filename</i>).
<u>PAGELength</u> (<i>n</i>)	Назначает максимальное число строк на каждой странице файла листинга.
<u>PAGEWidth</u> (<i>n</i>)	Назначает максимальное число символов в каждой строке файла листинга.
<u>PRINT</u> [(<i>filename</i>)], <u>NOPRINT</u>	Запрещает или разрешает создание файла листинга. Листинг сохраняется в файле с назначенным именем (<i>filename</i>).
<u>REGISTERBANK</u> (<i>num</i> , ...), <u>NOREGISTERBANK</u>	Устанавливает использование одного или нескольких регистров банков или указывает на то, что регистры банков не используются.
<u>RESET</u> (<i>symbol</i> , ...)	Присваивает значение 0000h символической ссылке (имени).
<u>SET</u> (<i>symbol</i> , ...)	Присваивает значение 0FFFFh символической ссылке (имени).
<u>TITLE</u> (<i>title</i>)	Включает название (<i>title</i>) в заголовок листинга.
<u>XREF</u>	Включает список символических перекрестных ссылок в файл листинга.

Пример файла листинга

Ниже приведен пример файла листинга, сгенерированный ассемблером A51. Файл содержит исходный код, сгенерированный машинный код, информацию о директивах и таблицу символических ссылок.

```
A51 MACRO ASSEMBLER Test Program 07/01/95 08:00:00 PAGE 1

DOS MACRO ASSEMBLER A51 V5.02
OBJECT MODULE PLACED IN SAMPLE.OBJ
ASSEMBLER INVOKED BY: C:\C51\BIN\A51.EXE SAMPLE.A51 XREF

LOC OBJ LINE SOURCE
1 $TITLE ('Test Program')
2 NAME SAMPLE
3
4 EXTRN CODE (PUT_CRLF, PUTSTRING, InitSerial)
5 PUBLIC TXTBIT
6
7 PROG SEGMENT CODE
8 CONST SEGMENT CODE
9 BITVAR SEGMENT BIT
10
----          11 CSEG AT 0
12
0000 020000 F 13 Reset: JMP Start
14
----          15 RSEG PROG
16 ; *****
0000 120000 F 17 Start: CALL InitSerial ;Init Serial
Interface
18
19 ; This is the main program. It is an endless
20 ; loop which displays a text on the console.
0003 C200 F 21 CLR TXTBIT ; read from CODE
0005 900000 F 22 Repeat: MOV DPTR,#TXT
0008 120000 F 23 CALL PUTSTRING
000B 120000 F 24 CALL PUT_CRLF
000E 80F5 25 SJMP Repeat
26 ;
----          27 RSEG CONST
0000 54455354 28 TXT: DB 'TEST PROGRAM',00H
0004 2050524F
0008 4752414D
000C 00
29
30
31
----          32 RSEG BITVAR ; TXTBIT=0 read from CODE
0000          33 TXTBIT: DBIT 1 ; TXTBIT=1 read from XDATA
```

Ассемблер A51 генерирует файл листинга, в котором указываются номера страниц, а также дата и время ассемблирования. В комментариях также приводятся данные о вызове ассемблера и объектном файле.

Обычно код начинается с директив EXTERN, PUBLIC и SEGMENT.

Для каждой строки кода указан номер.

Если в строке происходит обращение к переменным, в начале строки указываются их значения в шестнадцатеричном формате.

В файл также включаются сообщения об ошибках и предупреждения. Место каждой ошибки четко обозначено.


```

34
35 END
XREF SYMBOL TABLE LISTING
-----
NAME          TYPE VALUE  ATTRIBUTES / REFERENCES
BITVAR . . . . . B SEG  0001H    REL=UNIT 9# 32
CONST. . . . . C SEG  000DH    REL=UNIT 8# 27
INITSERIAL . . . . C ADDR  -----  EXT 4# 17
PROG . . . . . C SEG  0010H    REL=UNIT 7# 15
PUTSTRING. . . . . C ADDR  -----  EXT 4# 23
PUT_CRLF . . . . . C ADDR  -----  EXT 4# 24
REPEAT . . . . . C ADDR  0005H R  SEG=PROG 22# 25
RESET. . . . . C ADDR  0000H A  13#
SAMPLE . . . . . N NUMB  -----  2
START. . . . . C ADDR  0000H R  SEG=PROG 13 17#
TXT. . . . . C ADDR  0000H R  SEG=CONST 22 28#
TXTBIT . . . . . B ADDR  0000H.0 R  SEG=BITVAR 5 5 21
33#

REGISTER BANK(S) USED: 0

ASSEMBLY COMPLETE. 0 WARNING(S), 0 ERROR(S)

```

Опция ассемблера XREF создает словарь перекрестных ссылок, в котором перечислены все ссылки и номера строк, в которых они встречаются. Номер строки, в которой объявляются ссылки, отмечен значком "#".

В конце файла приведены все используемые банки регистров, а также количество предупреждений и ошибок.

Загрузчик/компоновщик BL51

Загрузчик/компоновщик BL51 объединяет один или несколько объектных модулей в один исполняемый файл. Компоновщик также разрешает внешние и глобальные ссылки и назначает абсолютные адреса перемещаемым сегментам программ.

Компоновщик работает с объектным модулем, созданным компилятором Keil C51 и ассемблером A51, а также компилятором Intel PL/M-51 и ассемблером ASM-51. Компоновщик автоматически выбирает подходящие библиотеки рабочих программ и связывает только нужные модули библиотек. Как правило, компоновщик вызывается из командной строки, где указываются имена объектных модулей, которые надо скомпоновать. Директивы управления, используемые по умолчанию, подобраны так, чтобы большинство приложений могло обходиться без специальных директив. Но при необходимости можно задать специальные настройки для конкретного приложения.

Управление доступом к данным

Компоновщик BL51 обходится ограниченным объемом внутренней памяти контроллера 8051 путем сохранения переменных одних функций поверх переменных других (речь идет о функциях, которые не зависят друг от друга). Это значительно снижает требования к памяти для большинства приложений. Для осуществления перекрытия переменных компоновщик анализирует связи между функциями. Чтобы вручную контролировать ссылки, используемые компоновщиком для выявления областей памяти, которые могут быть перекрыты, нужно использовать директиву OVERLAY. Директива NOOVERLAY запрещает перекрытие переменных в памяти. Обе эти директивы могут пригодиться при использовании косвенно вызываемых функций или при отладке, когда необходимо запретить перекрытие.

Распределение кода в памяти

Загрузчик/компоновщик BL51 предоставляет возможность создания программ размером больше 64 Кб. В таких случаях, поскольку микроконтроллер физически имеет только 64 Кб памяти, необходимо использовать внешние устройства, которые будут осуществлять подкачку кода. Эти устройства не обязательно должны управляться программой, загруженной в микроконтроллер 8051. Процесс подкачки еще называют переключением банков.

Компоновщик позволяет использовать одну общую область и 32 банка размером до 64 Кб, составляющих вместе до 2 Мб памяти. Управление переключением банков

осуществляется с помощью специального ассемблерного файла, который можно отредактировать под конкретную платформу. Компоновщик позволяет указать банк, в который нужно загрузить отдельный модуль программы. Продуманно распределяя функции по разным банкам, можно создавать большие эффективные приложения.

Общая область памяти

Компоновщик значительно расширяет возможности создания программ для микроконтроллеров семейства 8051. BL51 позволяет управлять общей областью памяти и 32 банками размером до 64 Кбайт, что составляет до 2 Мбайт коммутируемого адресного пространства. Общая область в программе с переключением банков - это область памяти, доступная в любое время всем банкам, которая не может физически выгружаться и загружаться снова. Код общей области или дублируется в каждом банке, или располагается в отдельной области памяти. Общая область содержит участки программы и константы, которые могут использоваться в любой момент, а также часто используемый код. По умолчанию в общей области автоматически размещаются:

- Векторы прерываний и перезагрузки;
- Константы;
- Обработчики прерываний C51;
- Таблица переходов при переключении банков;
- Некоторые функции библиотеки рабочих программ.

Программная поддержка внешнего устройства переключения банков состоит из короткого редактируемого ассемблерного файла.

Выполнение функций в других банках

BL51 генерирует таблицу переходов для функций, расположенных в разных кодовых банках. Банки памяти выбираются дополнительными программно-управляемыми адресными шинами, работу которых симулируют шины порта ввода-вывода или защелки распределения памяти. Компоновщик генерирует таблицу переходов для функций в других банках. Если вызывается функция, расположенная в другом банке, программа переключает банк и переходит к нужной функции, а по завершении ее работы возвращается в предыдущий банк и продолжает выполнение.

Поскольку переключение банка требует приблизительно 50 циклов и 2 байта стекового пространства, следует группировать взаимосвязанные функции в одном банке. Функции, часто вызываемые из различных банков, должны располагаться в общей области. BL51 эффективно управляет данными, размещаемыми во внутреннем ОЗУ, замещая переменные одной функции переменными другой, если исключено их взаимное и одновременное использование. Это существенно уменьшает требования к памяти для большинства приложений.

Командная строка

Вызов:

BL51 inputlist [TO outputfile] [directives]

L51 inputlist [TO outputfile] [directives]

BL51 @commandfile

L51 @commandfile

При этом:

sourcefile Имя исходного файла на C.

- commandfile** Имя файла, содержащего командную строку компилятора, включая sourcefile и directives. Вы можете использовать командный файл для более простой компоновки исходного файла или в том случае, когда все директивы не помещаются в командной строке.
- directives** Параметры элементов управления, описываемые ниже.

Таблица 57

Элементы управления BL51	Описание
<u>BANK</u> AREA	Назначает область адресов для размещения банков кода.
<u>BANK</u> <i>x</i>	Назначает начальный адрес, сегменты и объектные модули для банков 0-31.
<u>BIT</u>	Определяет расположение и порядок сегментов BIT.
<u>CODE</u> **	Определяет расположение и порядок сегментов CODE.
<u>COMMON</u> *	Назначает начальный адрес, сегменты и объектные модули для размещения в общем банке.
<u>DATA</u>	Определяет расположение и порядок сегментов DATA.
<u>IDATA</u>	Определяет расположение и порядок сегментов IDATA.
<u>IXREF</u>	Включает таблицу перекрестных ссылок в файл листинга.
<u>NAME</u>	Назначает имя модуля для объектного файла.
<u>NOAMAKE</u>	Исключает AutoMAKE-информацию из объектного файла.
<u>NODEBUGLINES</u>	Не включает информацию о номерах строк в объектный файл.
<u>NODEBUGPUBLICS</u>	Не включает информацию о глобальных именах в объектный файл.
<u>NODEBUGSYMBOLS</u>	Не включает информацию о локальных именах в объектный файл.
<u>NODEFAULTLIBRARY</u>	Исключает модули из библиотеки рабочих программ.
<u>NOLINES</u>	Не включает информацию о номерах строк в файл листинга.
<u>NOMAP</u>	Не включает карту распределения памяти в файл листинга.
<u>NOOVERLAY</u>	Запрещает перекрытие локальных сегментов BIT и DATA.
<u>NOPUBLICS</u>	Не включает информацию о глобальных ссылках в файл листинга.
<u>NOSYMBOLS</u>	Не включает информацию о локальных именах в файл листинга.
<u>OVERLAY</u>	Разрешает перекрытие локальных сегментов BIT и DATA и позволяет изменять ссылки между сегментами.
<u>PAGELENGTH</u> (<i>n</i>)	Назначает максимальное число строк на каждой странице файла листинга.
<u>PAGewidth</u> (<i>n</i>)	Назначает максимальное число символов в каждой строке файла листинга.
<u>PDATA</u>	Назначает начальный адрес сегмента PDATA.
<u>PRECEDE</u>	Определяет расположение и порядок следования сегментов, предшествующих всем остальным сегментам внутренней памяти данных.
<u>PRINT</u>	Назначает имя файла листинга.
<u>RAMSIZE</u>	Назначает размеры внутренней памяти данных.
<u>REGFILE</u> (FILENAME)	Назначает имя файлу, который будет содержать информацию об использовании регистров.
<u>RTX51</u> *	Включает поддержку ядра реального времени RTX-51 FULL.
<u>RTX51TINY</u> *	Включает поддержку ядра реального времени RTX-51 TINY.
<u>STACK</u>	Определяет расположение и порядок сегментов STACK.
<u>XDATA</u> **	Определяет расположение и порядок сегментов XDATA.

* Эти директивы доступны только в компоновщике BL51.

** При использовании этих директив следует учитывать особенности распределения памяти в SDK-1.1 и особенности работы резидентного загрузчика HEX202. См. соответствующие разделы.

Пример файла листинга

Ниже приведен пример файла распределения памяти, сгенерированный компоновщиком BL51.

```
BL51 BANKED LINKER/LOCATER V3.52 07/01/95 08:00:00 PAGE
1
```

Компоновщик BL51 генерирует
файл распределения памяти, в

```
MS-DOS BL51 BANKED LINKER/LOCATER V3.52, INVOKED BY:
C:\C51\BIN\BL51.EXE SAMPLE.OBJ
```

котором указываются дата и время загрузки/компоновки.

```
MEMORY MODEL: SMALL
```

```
INPUT MODULES INCLUDED:
SAMPLE.OBJ (SAMPLE)
C:\C51\LIB\C51S.LIB (?C_STARTUP)
C:\C51\LIB\C51S.LIB (PUTCHAR)
C:\C51\LIB\C51S.LIB (GETCHAR)
C:\C51\LIB\C51S.LIB (TOUPPER)
C:\C51\LIB\C51S.LIB (_GETKEY)
```

Приведен список вызовов и выбранный модуль памяти.

```
LINK MAP OF MODULE: SAMPLE (SAMPLE)
```

```
TYPE      BASE LENGTH RELOCATION SEGMENT NAME
-----
* * * * * D A T A M E M O R Y * * * * *
REG       0000H 0008H ABSOLUTE "REG BANK 0"
DATA      0008H 0001H UNIT ?DT?GETCHAR
DATA      0009H 0001H UNIT _DATA_GROUP_
          000AH 0016H *** GAP ***
BIT       0020H.0 0000H.1 UNIT ?BI?GETCHAR
          0020H.1 0000H.7 *** GAP ***
IDATA    0021H 0001H UNIT ?STACK
* * * * * C O D E M E M O R Y * * * * *
CODE      0000H 0003H ABSOLUTE
CODE      0003H 0021H UNIT ?PR?MAIN?SAMPLE
CODE      0024H 000CH UNIT ?C_C51STARTUP
CODE      0030H 0027H UNIT ?PR?PUTCHAR?PUTCHAR
CODE      0057H 0011H UNIT ?PR?GETCHAR?GETCHAR
CODE      0068H 0018H UNIT ?PR?_TOUPPER?TOUPPER
CODE      0080H 000AH UNIT ?PR?_GETKEY?_GETKEY
```

Карта связей содержит таблицу использования физической памяти контроллера 8051.

```
OVERLAY MAP OF MODULE: SAMPLE (SAMPLE)
```

```
SEGMENT          DATA_GROUP
+--> CALLED SEGMENT START LENGTH
-----
?C_C51STARTUP    -----
+--> ?PR?MAIN?SAMPLE
?PR?MAIN?SAMPLE  0009H 0001H
+--> ?PR?GETCHAR?GETCHAR
+--> ?PR?_TOUPPER?TOUPPER
+--> ?PR?PUTCHAR?PUTCHAR
?PR?GETCHAR?GETCHAR -----
+--> ?PR?_GETKEY?_GETKEY
+--> ?PR?PUTCHAR?PUTCHAR
```

Оверлейная карта отображает структуру программы и размещение сегментов данных и кода каждой функции.

```
LINK/LOCATE RUN COMPLETE. 0 WARNING(S), 0 ERROR(S)
```

В конце файла приводится список сообщений об ошибках и предупреждений. Сообщения содержат информацию о проблемах, возникших в процессе загрузки/компоновки.

Утилиты

Все пакеты инструментальных средств разработки включают в себя комплекты утилит, предназначенные для объединения объектных файлов и их преобразования в различные форматы.

Конвертер объектных файлов ОС51

Конвертер объектных модулей ОС51 создает объектные модули с абсолютными адресами для каждого банка модуля (если он размещен по частям в разных банках). Это происходит при создании приложения с переключением банков. Информация о символических ссылках

копируется в объектный файл, размещенный по абсолютному адресу, и может быть использована для отладки в dScore или встроенном эмуляторе.

Конвертер объектных файлов можно использовать для создания объектных модулей в абсолютных адресах для области команд и каждого из банков, задействованных в создании модуля. Затем с помощью конвертера OH51 генерируются шестнадцатеричные файлы для каждого модуля.

Вызов:

OC51 banked_file

При этом:

banked_file Имя исходного объектного файла.

Шестнадцатеричный конвертер OH51

Конвертер OH51 конвертирует объектные модули в абсолютных адресах в шестнадцатеричные файлы в формате Intel. Модули перед этим создаются с помощью компоновщика BL51 или конвертера OC51. Шестнадцатеричные файлы формата Intel – это файлы в ASCII-кодах, содержащие шестнадцатеричное представление приложения.

Вызов:

OH51 absfile | HEXFILE (hexfile)

При этом:

absfile Имя объектного модуля, построенного в абсолютных адресах.

hexfile Имя шестнадцатеричного файла в формате Intel, который должен быть создан.

Менеджер библиотек LIB51

Менеджер библиотек LIB51 позволяет создавать библиотечные файлы и управлять ими. Файл библиотеки – это форматированный набор одного или более объектных файлов. Библиотечный файл представляет собой удобный способ объединить большое количество объектных модулей, сохранив при этом все ссылки. Библиотеки эффективно используются загрузчиком/компоновщиком BL51. С помощью менеджера библиотек вы можете создать библиотечный файл, добавлять объектные модули в файл библиотеки, удалять модули из библиотеки, а также просматривать содержимое библиотечного файла. Управление менеджером библиотек ведется либо в интерактивном режиме, либо из командной строки.

Во все пакеты средств разработки включены библиотеки функций, оптимизированных по времени выполнения. Доступно более 100 библиотечных программ, включая все функции ANSI C, адаптированные для микроконтроллерных приложений. Остановимся только на двух библиотеках. Это библиотека программ, обеспечивающих поддержку конкретного микроконтроллера, и библиотека встроенных (Intrinsic) функций. Особенностью этих библиотек является то, что при вызове других функций генерируется команда перехода (по CALL), здесь же происходит только слияние общего потока программы с текстом встроенной функции, что значительно быстрее и эффективнее. Встроенные функции предусмотрены для всех специфических команд микроконтроллеров (например, einit, diswdt, testbit, trap, idle).

Вызов:

LIB51 [command]

При этом:

command Управляющая команда, описанная в таблице ниже. Если команда не указана, LIB51 запускается в интерактивном командном режиме.

Таблица 58

Команда LIB51	Описание
<u>A</u> DD	Добавляет объектный модуль в файл библиотеки.
<u>C</u> REATE	Создает новый файл библиотеки.
<u>D</u> ELETE	Удаляет объектный модуль из библиотеки.
<u>E</u> XIT	Завершает работу в интерактивном режиме.
<u>H</u> ELP	Отображает справку.
<u>L</u> IST	Выводит информацию о модулях и символических ссылках, содержащуюся в файле библиотеки.

Отладчик/симулятор Keil dScope

В пакет инструментальных средств Keil 8051 входит высокоуровневый отладчик/симулятор dScope, который позволяет вести отладку исходных текстов программ, написанных на C и ассемблере фирмы Keil. dScope51 поддерживает также язык PL/M-51 и ассемблер ASM-51 фирмы Intel. dScope - это чисто программный продукт, который симулирует большинство ресурсов микроконтроллера без отлаживаемого устройства.

dScope позволяет проводить пошаговую отладку программ, просматривая в специальном окне Debug исходный текст программы на языке C, на ассемблере или в смешанном формате. Трассировщик запоминает до 512 инструкций и позволяет их просматривать в окне Trace. Изменение заранее заданных переменных отслеживает окно Watch. Последовательность вызова процедур можно просмотреть в специальном окне Call-Stack.

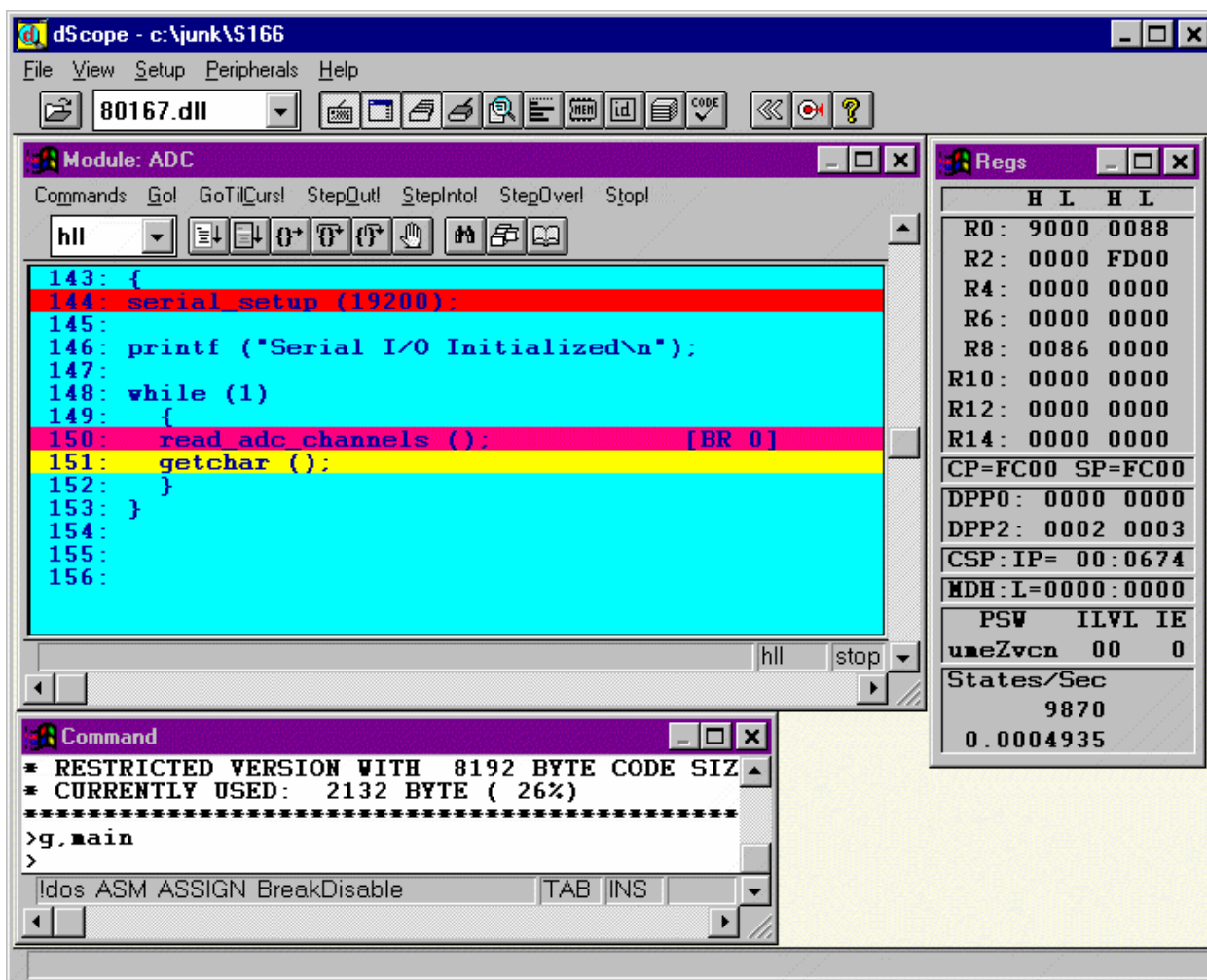


Рисунок 68. Отладчик/симулятор Keil dScope.

dScope предлагает широкие возможности по заданию простых и условных точек останова. Условием останова может быть или результат выражения, или операция обращения к ячейке памяти (чтение, запись, доступ). Для редактирования и просмотра параметров контрольных точек служит окно Breakpoint.

Для автоматизации типовых выполняемых при отладке операций могут быть созданы специальные файлы из собственных команд dScope. Для ввода этих команд служит окно Command или окно Toolbox. Кроме того, dScope поддерживает C как макроязык. При отладке могут использоваться следующие типы функций: встроенные (printf, memset, rand), сигнальные для симуляции аналоговых и цифровых входных/выходных сигналов (исполняются в фоновом режиме), пользовательские для расширения команд отладчика.

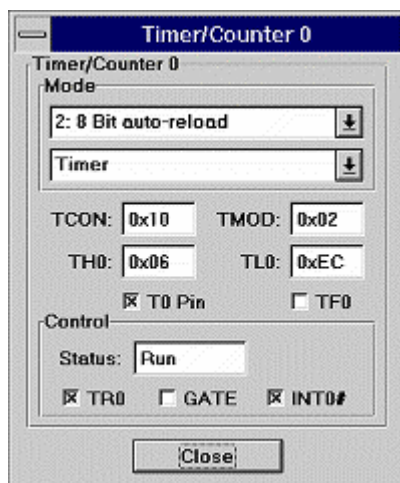


Рисунок 69

dScore полностью симулирует периферийные устройства семейства микроконтроллеров 8051 с помощью специального драйвера. Перед загрузкой отлаживаемого приложения выбирается специальный драйвер DLL, и в распоряжение пользователя предоставляется набор окон для работы с периферией, отображающих состояния таймеров, портов, прерываний, сторожевого таймера, последовательного порта, аналогово-цифрового преобразователя и т.д. Параметры этих устройств могут быть установлены в соответствии с контекстом приложения.

В распоряжении пользователя также имеются окна для просмотра областей памяти Memory и состояний регистров Register. С помощью окна Serial I/O становится возможной наглядная симуляция последовательного ввода/вывода.

В dScore встроен анализатор производительности, фиксирующий время исполнения программных модулей. Задавая список модулей для анализа, пользователь получает диаграмму затрат времени на каждую часть программы. dScore позволяет также провести анализ эффективности кода, локализуя части программы, к которым редко происходит обращение, что позволяет удалить ненужный код. Столбцовая диаграмма отображается в специальном окне Code Coverage.

Интегрированная среда разработки Keil μ Vision

Интегрированная среда разработки μ Vision включает полнофункциональный текстовый редактор, менеджер проектов и встроенную утилиту Make и работает в среде Windows 95/NT. При этом достигается простая интеграция различных исходных файлов в файл проекта. Интегрированный редактор значительно облегчает подготовку исходного текста за счет многооконности, цветового выделения синтаксиса и исправления ошибок в режиме диалога. Предусмотрена возможность настройки редактора в соответствии с нуждами конкретного проекта и вкусами пользователя.

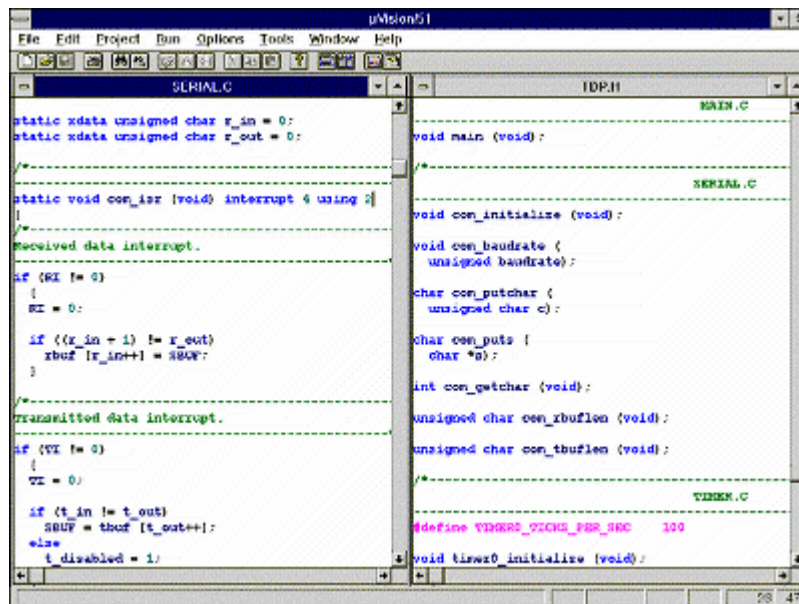


Рисунок 70

Как правило, разрабатываемые программные приложения содержат большое число связанных друг с другом программных файлов, которые часто обрабатываются индивидуально. Например, часть файлов подлежит С-компиляции, другие следует ассемблировать, а третьи требуют некоторой специальной обработки пользователем. Здесь на помощь приходит Менеджер проекта, который предлагает разработчику методику создания и эксплуатации проекта.

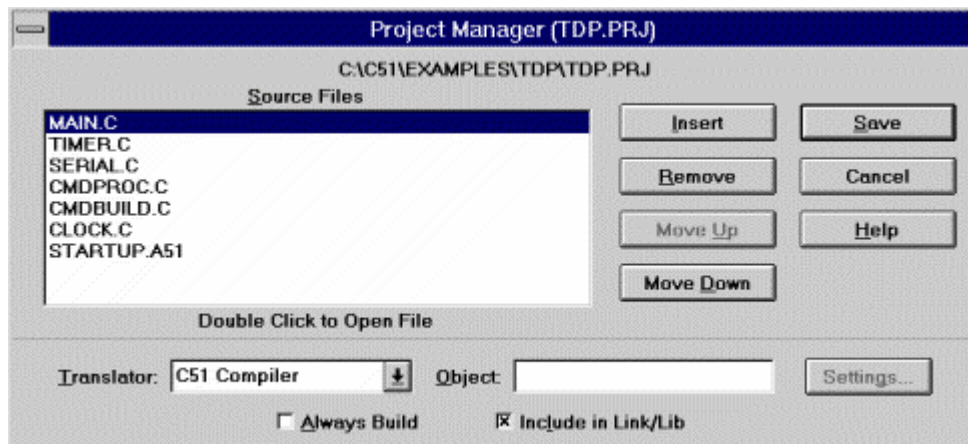


Рисунок 71. Менеджер проекта.

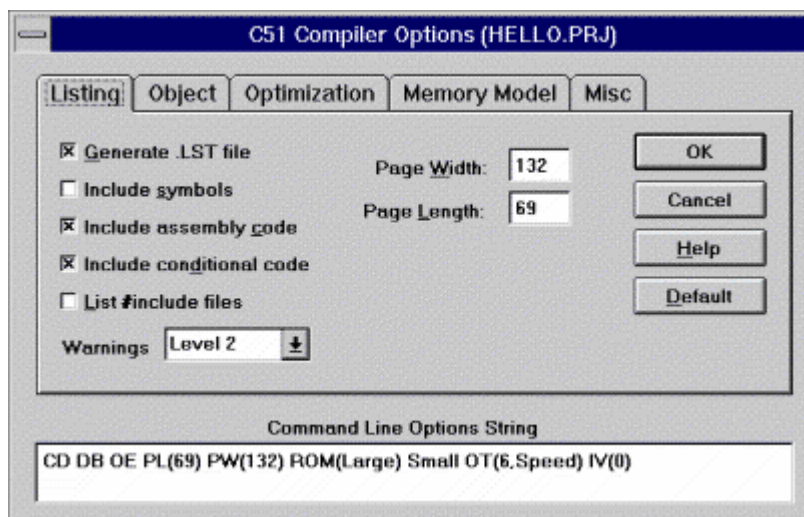


Рисунок 72

С помощью μ Vision легко осуществить настройку всех программ пакета: С-компилятора, ассемблера или встроенной утилиты Make. Для этого с помощью клавиатуры или мыши выбирается соответствующий пункт меню и корректируются значения опций.

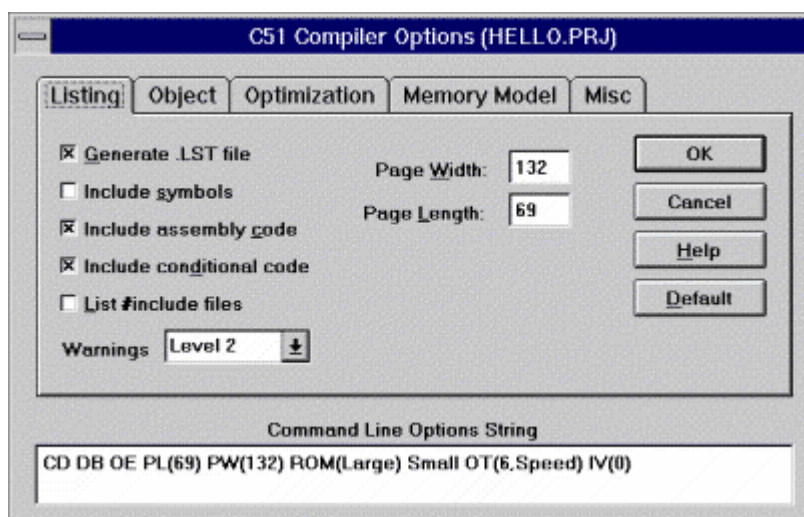


Рисунок 73

Все параметры проекта сохраняются в специальном файле, который содержит список исходных файлов, командные строки компилятора, ассемблера, редактора, отладчика, симулятора и утилиты Make. Встроенная утилита Make, используя данные этого файла, компилирует и связывает проект по нажатию одной клавиши, причем после внесения исправлений в исходный текст перекомпилируются лишь те файлы, которые были изменены. μ Vision позволяет непосредственно вызвать симулятор dScore и другие внешние средства.

Операционная система реального времени RTX51

При создании микропроцессорных приложений разработчик часто сталкивается со следующими проблемами:

- Задача должна быть выполнена в короткий отрезок времени, и через гарантированное время от нее должен быть получен ответ (режим работы в реальном времени).
- Различные по времени и логике работы задачи подчинены друг другу и должны выполняться одновременно.

С помощью операционной системы реального времени RTX такие задачи могут быть организованы как независимые программные модули, поскольку RTX берет на себя решение проблем реального времени и многозадачности. RTX работает под управлением таймера, который измеряет относительное время с момента старта. В качестве временных отсчетов служат прерывания от одного из встроенных таймеров микроконтроллера. Интервал между прерываниями называется квантом системного времени и задается пользователем. Каждая задача выполняется на заданном кванте времени. Таким образом упрощается логическая структура многозадачных приложений и сокращается время разработки.

Облегченная версия операционной системы RTX Tiny интегрирована в C-компилятор Keil. Для работы с RTX также необходимы Macro Assembler и Linker/Locator. При таком подходе генерация приложений осуществляется достаточно просто. Расширенный синтаксис описания функции позволяет объявить задачу и описать процедуры обработки прерывания. Возможности RTX Tiny оптимальны для разработки однопроцессорных многозадачных систем. RTX Tiny:

- Осуществляет только циклическое (Round-Robin) переключение задач.
- Поддерживает только стандартные задачи, использующие общий банк регистров и область стека, что приводит к потерям времени при переключении задач.
- Для синхронизации задач с внешними событиями использует систему прерываний, но не содержит функций управления прерываниями. Прерывания обрабатываются стандартными процедурами на C. Процедуры при этом могут обмениваться сигналами и данными с задачами RTX.
- По истечению кванта времени происходит переключение на другую задачу с тем же приоритетом.
- Выполнение задачи инициируется одним из следующих событий: сигнал от задачи или прерывания, временной интервал, истечение кванта времени или любая комбинация этих событий. Отслеживанием событий занимается функция `os_wait`. Пока одна задача ожидает появления события, другие задачи выполняются.

Кроме RTX Tiny доступна полная версия операционной системы реального времени RTX Full, поставляемая в виде самостоятельного программного продукта. Возможности RTX Full значительно шире, чем у RTX Tiny:

- Кроме циклического Round-Robin доступно также предупреждающее (preemptive) переключение задач при нескольких уровнях приоритета.
- Помимо стандартных задач поддерживает быстрые задачи с собственным банком регистров и областью стека, что обеспечивает малое время ответа и переключения при прерываниях.
- Поддерживает параллельную работу и управление функциями прерывания. Прерывания обрабатываются стандартными процедурами на C, а также быстрыми и стандартными задачами RTX.
- Инициированная задача с наивысшим приоритетом поступает на выполнение вне очереди, прерывая текущую. Если этого не произошло, то после истечения кванта времени происходит переключение на другую задачу с тем же приоритетом.
- Помимо сигналов поддерживает систему почтовых ящиков и семафоров. Выполнение задачи инициируется одним из следующих событий: сигналом от задачи или прерывания, сообщением от задачи или прерывания, разрешением от семафора, истечением кванта времени или любой комбинацией этих событий. Отслеживанием событий занимается функция `os_wait`. Пока одна задача ожидает появления события, другие задачи выполняются.
- Поддерживает CAN-интерфейс: программирование отдельных CAN-контроллеров (например, Siemens 81C90 и 81C91) и кристаллов 8051 со встроенным CAN (например, Siemens C505C, C505CA и C515C).

Следует пояснить суть механизмов синхронизации задач. RTX Full поддерживает три метода синхронизации.

- Сигналы (Signals) - самый быстрый способ синхронизации, при котором одна задача для запуска на выполнение другой задачи посылает в нужный момент сигнал. RTX Tiny поддерживает только этот механизм синхронизации, являющийся самым простым.
- Сообщения (Messages) - это данные, пересылаемые через почтовые ящики. Почтовый ящик хранит множество сообщений по принципу FIFO, т.е. поступившее первым сообщение первым выдается на обработку. Если сообщения ожидает несколько задач, то его забирает задача с наивысшим приоритетом.
- Семафоры (Semaphores) - простой механизм для разделения общих ресурсов, таких, как последовательный порт. При потребности в ресурсе задача выставляет требование к соответствующему семафору. Семафор предоставляет ресурс, если нет других нуждающихся в нем задач.

Таблица 59. Характеристики RTX.

Параметр	RTX51 Tiny	RTX51 Full
Максимальное число задач	16	256
Максимальное число почтовых ящиков	-	8
Максимальное число семафоров	-	8
Уровни приоритета	1	4
Время контекстного переключения, циклов	100 - 700	70 - 700
Размер кода, Кбайт	0,9	6 - 8
ОЗУ, байт	data:7 idata: 3 * task_count	data:40-46 idata:20-200 xdata:>650