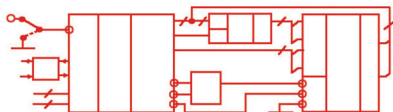
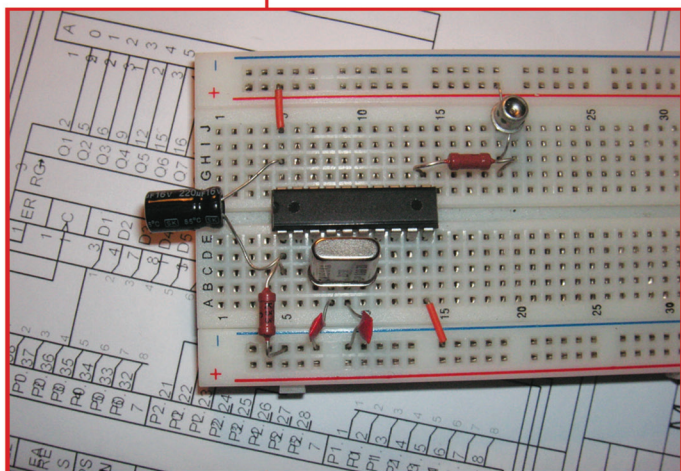


Схемотехника: аппаратура и программы



Аверченков О. Е.

О. Е. Аверченков

СХЕМОТЕХНИКА: аппаратура и программы

*Посвящается 50-летию Смоленского филиала МЭИ,
человеку, который придумал, и людям, его создавшим –
Пищикову В.И., Лейтману М.Б., Дьяконову В.П., Мамедову Ф.А –
и всем первым преподавателям, сотрудникам и студентам*



Москва, 2012

УДК 004.3:621.382
ББК 32.973.26-04+32.844
A19

Рецензент: доктор технических наук, профессор филиала МЭИ
в г. Смоленске Борисов В. В.

Аверченков О. Е.

A19 Схемотехника: аппаратура и программы. – М.: ДМК Пресс, 2012. – 588 с.

ISBN 978-5-94074-402-3

Данное пособие предназначено для первоначального ознакомления с элементной базой и функциональными типовыми узлами микропроцессорных систем. В нем описываются ключевые и логические элементы, комбинационные устройства, триггеры, регистры, счетчики, микросхемы памяти. Большое внимание уделено усилителям и аналого-цифровым устройствам, без которых не может обойтись процессор при общении с внешними датчиками.

Главной особенностью пособия, ориентированного на самостоятельное усвоение материала, является попытка совместить изучение схемотехнической аппаратуры с практическими приемами подключения ее к однокристалльной вычислительной машине (ОВМ), являющейся ядром современных систем обработки данных. В ряде случаев приведены программы, программно реализующие схемотехнические функции и позволяющие исключить или упростить внешнюю аппаратуру.

Рассмотрены не только принципы действия типовых устройств, но и приведены конкретные примеры их реализации и практического использования, что позволяет рекомендовать данное пособие для курсового проектирования, особенно, если оно осуществляется одновременно с чтением лекций.

Несмотря на то, что пособие предназначено в первую очередь для первоначального ознакомления с основами схемотехники, некоторые материалы могут представлять интерес и для подготовленных читателей.

УДК 004.3:621.382

ББК 32.973.26-04+32.844

Все права защищены. Любая часть этой книги не может быть воспроизведена в какой бы то ни было форме и какими бы то ни было средствами без письменного разрешения владельцев авторских прав.

Материал, изложенный в данной книге, многократно проверен. Но, поскольку вероятность технических ошибок все равно существует, издательство не может гарантировать абсолютную точность и правильность приводимых сведений. В связи с этим издательство не несет ответственности за возможные ошибки, связанные с использованием книги.

ISBN 978-5-94074-402-3

© Аверченков О. Е., 2012

© Оформление, издание, ДМК Пресс, 2012

Оглавление

Введение.....	24
ГЛАВА 1. Ключевые схемы.....	27
§1.1. Базовая ключевая схема.....	27
1.1.1. Схема с идеальным ключом.....	27
1.1.2. Временная диаграмма выходного напряжения.....	28
1.1.3. Ограничение величины внешнего резистора снизу.....	28
1.1.4. Влияние сопротивления нагрузки.....	29
1.1.5. Мощность, выделяемая на ключе.....	29
1.1.6. Реальный ключ.....	30
§1.2. Влияние емкостей на форму импульсов.....	31
1.2.1. Влияние емкости нагрузки.....	31
1.2.2. Интегрирующая цепочка.....	32
1.2.3. Дифференцирующая цепочка.....	32
§1.3. Свойства полупроводникового диода.....	34
1.3.1. Вольтамперная характеристика.....	34
1.3.2. Эквивалентная схема проводящего диода.....	35
1.3.2. Эквивалентная схема непроводящего диода.....	36
§1.4. Диодные логические элементы.....	37
1.4.1. Схема диодного ключа.....	37
1.4.2. Диодный логический элемент «И» (AND).....	38
1.4.3. Диодный логический элемент «ИЛИ» (OR).....	39
1.4.4. «Монтажное ИЛИ».....	40
§1.5. Диодные ограничители уровня сигнала.....	41
1.5.1. Ограничитель отрицательного сигнала.....	41
1.5.2. Параллельный ограничитель отрицательной полярности.....	42
1.5.3. Ограничитель положительного напряжения сверху.....	42
1.5.4. Двусторонний ограничитель на стабилитроне.....	43
§1.6. Свойства биполярного транзистора в ключевом режиме	44
1.6.1. Режимы работы биполярного транзистора.....	44
1.6.2. Токи транзистора в режиме глубокой отсечки.....	46
1.6.3. Токи транзистора в режиме с оборванной базой.....	47
1.6.4. Токи транзистора в режиме «заземленной» базы.....	47
1.6.5. Схемы замещения непроводящего транзистора.....	48
1.6.6. Токовое условие насыщения биполярного транзистора.....	49
1.6.7. Схема замещения проводящего транзистора.....	50
§1.7. Ключ на биполярном транзисторе.....	51
1.7.1. Схема и работа ключа.....	51
1.7.2. Условие работоспособности ключа в режиме отсечки.....	52

1.7.3. Условие работоспособности ключа в режиме насыщения.....	54
1.7.4. Упрощенный порядок расчета ключа.....	55
§1.8. Переходный процесс при включении биполярного транзистора	55
1.8.1. Метод заряда.....	55
1.8.2. Процесс включения транзистора.....	56
1.8.3. Нахождение времени включения	57
1.8.4. Режим сильного запирающего сигнала	58
1.8.5. Режим слабого запирающего сигнала	59
§1.9. Переходный процесс при выключении биполярного транзистора	59
1.9.1. Временные диаграммы	59
1.9.2. Нахождение времени выключения.....	60
1.9.3. Режим сильного запирающего сигнала	61
1.9.4. Режим слабого запирающего сигнала	61
1.9.5. Учет заряда в области коллектора.....	62
§1.10. Повышение быстродействия ключа на биполярном транзисторе	62
1.10.1. Оптимальная форма базового тока	62
1.10.2. Ключ с ускоряющим конденсатором	63
1.10.3. Выбор емкости ускоряющего конденсатора.....	64
1.10.4. Экспериментальная оценка правильности выбора C_y	66
1.10.5. Ключ с нелинейной обратной связью (ОС)	66
1.10.6. Преимущества и недостатки ключа с нелинейной ОС.....	68
ГЛАВА 2. Интегральные логические элементы	69
§2.1. Разновидности логических интегральных элементов на биполярных транзисторах	69
2.1.1. Резистивно-транзисторная логика (РТЛ)	69
2.1.2. Диодно-транзисторная логика (ДТЛ)	70
2.1.3. ДТЛ элемент с одним источником питания	71
2.1.4. Многоэмиттерный транзистор	72
§2.2. Структура ТТЛ элемента.....	73
2.2.1. Иллюстративная схема ТТЛ элемента	73
2.2.2. Двухтранзисторный выходной каскад.....	73
2.2.3. «Сквозной» ток двухтранзисторного выходного каскада.....	74
2.2.4. Программное управление двухтранзисторным каскадом	76
§2.3. Базовая схема ТТЛ элемента.....	76
2.3.1. Функциональная схема.....	76
2.3.2. «Фазоразделитель»	76
2.3.3. Базовая схема ТТЛ элемента	77
2.3.4. Работа ТТЛ элемента.....	78

2.3.5. Работа ТТЛ элементов на общую информационную линию.....	79
2.3.6. ТТЛ элемент с открытым коллектором	80
2.3.7. ТТЛ элемент с третьим состоянием	81
§2.4. Параметры и характеристики ТТЛ элемента	82
2.4.1. Основные параметры.....	82
2.4.2. Входная характеристика	82
2.4.3. Определение входных токов ТТЛ элемента.....	83
2.4.4. Определение порогового напряжения.....	84
2.4.5. Передаточная характеристика	85
2.4.6. Особенности выходных (нагрузочных) характеристик	85
2.4.7. Единичная нагрузочная характеристика.....	86
2.4.8. Нулевая нагрузочная характеристика.....	87
2.4.9. Временные параметры.....	87
§2.5. Нормализация входного сигнала для ТТЛ элемента	89
2.5.1. Двухключевая схема	89
2.5.2. Схема с одним верхним ключом	89
2.5.3. Схема с верхним ключом и резистором	90
2.5.4. Схема с нижним ключом и резистором	90
2.5.5. Преобразователь двуполярного напряжения в ТТЛ уровень	91
2.5.6. Стабилитронный ограничитель двуполярного сигнала.....	92
§2.6. Ключи на полевых транзисторах	93
2.6.1. Общие сведения о МОП транзисторах	93
2.6.2. Ключ на полевом транзисторе	94
2.6.3. Схемы замещения полевого транзистора.....	95
2.6.4. Ключ на полевом транзисторе с транзисторной нагрузкой	97
2.6.5. Ключ на разнотипных МОП транзисторах	97
2.6.6. Быстродействие ключей на полевых транзисторах	98
§2.7. Логика на полевых транзисторах	99
2.7.1. Общие сведения	99
2.7.2. МОПТЛ элементы на однородных транзисторах	100
2.7.3. Базовая схема кМОПТЛ элемента «И-НЕ»	101
2.7.4. Характеристики кМОПТЛ элемента	102
ГЛАВА 3. Подключение нагрузки к выходу микросхем.....	104
§3.1. Подключение нагрузки без преобразования уровня сигнала ...	104
3.1.1. Нагрузка в виде входов логических элементов	104
3.1.2. Нагрузка в виде светодиода.....	105
3.1.3. Параллельная работа элементов	106
§3.2. Однокаскадные ключи для преобразования уровня	
выходного сигнала	107
3.2.1. Использование элемента с открытым электродом.....	107
3.2.2. Внешний ключ на n - p - n транзисторе	107
3.2.3. Внешний ключ без источника смещения	108

3.2.4. Внешний ключ на $p-n-p$ транзисторе	109
§3.3. Двухкаскадные ключевые усилители	109
3.3.1. Ключ на составном транзисторе	110
3.3.2. Устранение недостатков составного транзистора	111
3.3.3. Двухкаскадный ключ	111
3.3.4. Защита мощных выходных транзисторов	112
3.3.5. Ключ с двуполярным выходным напряжением	113
3.3.6. Микросхемы с наборами мощных ключей	114
§3.4. Ключи на мощных полевых транзисторах	115
3.4.1. Ключ на полевом транзисторе с низковольтным управлением	115
3.4.2. Ключ на полевом транзисторе с преобразованием управляющего уровня	116
3.4.3. Биполярный транзистор с изолированным затвором	117
ГЛАВА 4. Ключевые устройства с гальваническим разделением входа и выхода	118
§4.1. Ключевые устройства с оптической связью	118
4.1.1. Транзисторный оптрон	118
4.1.2. Коэффициент передачи тока оптрона	120
4.1.3. Параметры транзисторного оптрона	120
4.1.4. Оптонный преобразователь тока в ТТЛ сигнал	121
§4.2. Оптически управляемые тиристорные ключи	123
4.2.1. Основные свойства тиристора	123
4.2.2. Недостатки тиристора	124
4.2.3. Оптически управляемый тиристорный ключ	124
4.2.4. Особенности релейного режима	125
4.2.5. Фазоимпульсное управление тиристором	126
4.2.6. Управление мощностью за счет пропуска полупериодов	127
§4.3. Особенности электромагнитных цепей	127
4.3.1 Основные магнитные параметры	128
4.3.2 Закон полного тока	129
4.3.3. Закон электромагнитной индукции (закон Фарадея)	130
4.3.4. Магнитные материалы	130
§4.4. Уравнения импульсного трансформатора	131
4.4.1. Общие сведения	131
4.4.2. Интегральное уравнение импульсного трансформатора	132
4.4.3. Оценка вторичного напряжения	133
4.4.4. Оценка токов трансформатора	133
4.4.5. Временные диаграммы	134
§4.5. Режимы работы импульсного трансформатора	134
4.5.1. Воздействие на сердечник однополярных импульсов	134
4.5.2. Введение немагнитного зазора	136
4.5.3. Действие импульса большой длительности	136

§4.6. Искажения, вносимые трансформатором.....	137
4.6.1. Эквивалентная схема трансформатора	137
4.6.2. Формирование вершины импульса	138
4.6.3. Разрыв индуктивной цепи	140
4.6.5. Колебания при разрыве индуктивной цепи.....	141
4.6.6. Экспериментальная оценка параметров трансформатора.....	142
4.6.7. Формирование фронта выходного сигнала	142
§4.7. Ключ с импульсным трансформатором	143
4.7.1. Схема	143
4.7.2. Процесс формирования импульса	144
4.7.3. Процесс формирования выброса.....	145
4.7.4. Параметры выброса.....	145
4.7.5. Оценка числа витков трансформатора	146
4.7.6. Расчет сопротивлений ключа.....	147
ГЛАВА 5. Основы схемотехники ИОУ	148
§5.1. Обратная связь в усилителях.....	148
5.1.1. Общие сведения об усилителях	148
5.1.2. Разновидности обратной связи	150
5.1.3. Коэффициент усиления усилителя с ПОС.....	151
5.1.4. Коэффициент усиления усилителя с ООС.....	152
5.1.5. Общие свойства отрицательной ОС.....	153
5.1.6. Самовозбуждение усилителя.....	154
5.1.7. Методы срыва генерации	154
5.1.8. Основные требования к усилительным каскадам	155
§5.2. Параметры усилительного каскада	156
5.2.1. Схемы включения транзисторов	156
5.2.2. Эквивалентная схема усилительного каскада с ОЭ	156
5.2.3. Входное сопротивление схемы с ОЭ.....	157
5.2.4. Коэффициент усиления напряжения для схемы с ОЭ	158
5.2.5. Выходное сопротивление схемы с ОЭ.....	158
5.2.6. Схема и входное сопротивление эмиттерного повторителя.....	159
5.2.7. Коэффициент усиления ЭП по напряжению.....	160
5.2.8. Выходное сопротивление ЭП	160
§5.3. Простейшие УПТ	161
5.3.1. УПТ на одном транзисторе.....	161
5.3.2. Дрейф нулевого уровня УПТ	162
5.3.3. Усилительный каскад с термокомпенсацией.....	163
5.3.4. Дифференциальный каскад.....	164
§5.4. Структура и параметры ИОУ.....	165
5.4.1. Описание иллюстративной схемы ИОУ	165
5.4.2. Условное графическое обозначение (УГО)	166
5.4.3. Параметры входной цепи	168

5.4.4. Параметры выходной цепи	169
5.4.5. Передаточные параметры	169
5.4.6. Частотные и временные параметры	170
5.4.7. Типы ИОУ	170
§5.5. Неинвертирующий УПТ на основе ИОУ	171
5.5.1. Схема	171
5.5.2. Коэффициент усиления	172
5.5.3. Входное сопротивление	173
5.5.4. Главные особенности неинвертирующего УПТ	173
5.5.5. Порядок расчета	174
5.5.6. Числовой пример расчета	174
§5.6. Инвертирующий УПТ на основе ИОУ	175
5.6.1. Схема	175
5.6.2. Коэффициент усиления	176
5.6.3. Входное сопротивление	177
5.6.4. Зависимость коэффициента β от сопротивления R_y	177
5.6.5. Главные особенности инвертирующего УПТ	178
5.6.6. Порядок расчета	179
5.6.7. Числовой пример расчета	179
§5.7. Погрешности УПТ	180
5.7.1. Общие сведения	180
5.7.2. Потенциальная аддитивная составляющая	181
5.7.3. Токовая аддитивная составляющая	182
5.7.4. Оценка общей аддитивной погрешности	183
5.7.5. Мультипликативная погрешность	184
ГЛАВА 6. Примеры использования УПТ на ИОУ	186
§6.1. Использование инвертирующего УПТ	186
6.1.1. Инвертирующий сумматор напряжений	186
6.1.2. Интегратор	187
6.1.3. Преобразователь тока в напряжение	188
6.1.4. Устранение неинформационного нулевого уровня	190
§6.2. Использование неинвертирующего УПТ	191
6.2.1. Повторитель напряжения	191
6.2.2. Неинвертирующий сумматор напряжений	192
6.2.3. Усилитель с программируемым усилением	192
§6.3. Дифференциальные УПТ с обратной связью	193
6.3.1. Усилитель разности однополярных напряжений	193
6.3.2. Усилитель разности с повышенным входным сопротивлением	195
6.3.3. Усилитель разности с повышенным коэффициентом усиления	195
6.3.4. Измерительный усилитель разности	196
§6.4. Усилитель с мощным выходным каскадом	197
6.4.1. Общие сведения	197

6.4.2. Эмиттерный повторитель класса А	198
6.4.3. Двухтактный эмиттерный повторитель класса В	199
6.4.3. Достоинства и недостатки каскада класса В	199
6.4.4. Усилитель с мощным выходным каскадом	200
6.4.6. Выбор типа транзисторов выходного каскада	201
6.4.7. Усилительный каскад класса D	202
§6.5. Усилители переменного напряжения	203
6.5.1. Расчет разделительного конденсатора	203
6.5.2. Инвертирующий УНЧ	204
6.5.3. Влияние нулевого уровня УПТ	204
6.5.4. Порядок расчета инвертирующего УНЧ	205
6.5.5. Неинвертирующий УНЧ	205
6.5.6. Неинвертирующий УНЧ с раздельной ООС по переменному и постоянному токам	207
6.5.6. Неинвертирующий УНЧ с повышенным входным сопротивлением	208
§6.6. Однополярное питание усилителей	209
6.6.1. Смещение информационной составляющей	209
6.6.2. Учет смещения сигнала при обработке	210
6.6.3. Инвертирующий УНЧ с однополярным питанием	211
6.6.4. Параметры ИОУ при однополярном питании	212
6.6.5. Получение отрицательного напряжения из импульсов	214
§6.7. Генератор синусоидальных колебаний	215
6.7.1. Общие сведения	215
6.7.2. Условия возникновения генерации в усилительной схеме	216
6.7.3. Схема генератора Вина	216
6.7.4. Коэффициент передачи звена обратной связи	217
6.7.5. Нахождение частоты генерации	217
6.7.6. Нахождение требуемого коэффициента усиления	218
§6.8. Кварцевые генераторы	218
6.8.1. Свойства кварцевого резонатора	219
6.8.2. Кварцевый генератор на неинвертирующем усилителе	220
6.8.3. Кварцевый генератор на инвертирующем усилителе	222
 ГЛАВА 7. Внутренние ресурсы ОВМ семейства x51	 224
§7.1. Общее описание ОВМ x51	224
7.1.1. Внутренняя структура ОВМ	224
7.1.2. Процессорное ядро	224
7.1.3. Внутренняя периферия	225
7.1.4. Условное графическое обозначение	226
7.1.5. Служебные выводы общего назначения	227
7.1.6. Выводы для обслуживания внешней памяти	229
7.1.7. Альтернативные функции порта P3	230

§7.2. Особенности выходных цепей портов	230
7.2.1. Выходные цепи порта P0.....	230
7.2.2. Выходные цепи портов P1, P2, P3.....	231
7.2.3. Особенности маловыводного варианта x51	232
7.2.4. Подключение нагрузки к выводу порта.....	233
7.2.5. Внешний ключ на <i>n-p-n</i> транзисторе	235
7.2.6. Внешний ключ на <i>p-n-p</i> транзисторе	237
§7.3. Работа ОВМ и структура памяти	237
7.3.1. Общие сведения.....	237
7.3.2. Структура памяти команд (ПЗУ)	238
7.3.3. Способы занесения кодов во внутреннее ПЗУ	239
7.3.4. Структура памяти данных	240
7.3.5. Навигация по ячейкам памяти	241
§7.4. Спецрегистры ОВМ x51	242
7.4.1. Аккумулятор.....	242
7.4.2. Регистр признаков	242
7.4.3. «Нефиксируемые» признаки результата	244
7.4.4. Особые биты регистра признаков	244
7.4.5. Регистры указатели	244
7.4.6. Спецрегистры внутренней периферии (SFR).....	245
§7.5. Память данных – регистры и операнды	246
7.5.1. Структура основного ОЗУ	246
7.5.2. Определение операндов пользователя	247
7.5.3. Использование зарегистрированных имен	248
§7.6. Методы адресации операндов	248
7.6.1. Непосредственная адресация.....	248
7.6.2. Регистровая адресация	249
7.6.3. Прямая адресация.....	250
7.6.4. Косвенная адресация	250
7.6.5. Относительная адресация	251
§7.7. Особенности ассемблерных команд.....	251
7.7.1. Мнемокоды команд	251
7.7.2. Особенности записи мнемокода команды.....	252
§7.8. Команды пересылки данных	253
7.8.1. Общий вид команд пересылки	253
7.8.2. Регистровые пересылки	253
7.8.3. Пересылки с косвенно-регистровой адресацией.....	254
7.8.4. Пересылки с прямой адресацией	255
7.8.5. Пересылка (загрузка) констант	256
7.8.6. Загрузка констант из ПЗУ с адресацией по сумме регистров.....	257
§7.9. Арифметические команды	257
7.9.1. Общие особенности	257
7.9.2. Команды сложения (ADD)	258

7.9.3. Команды вычитания	259
7.9.4. Команда десятичной коррекции аккумулятора	259
7.9.5. Команда умножения	260
7.9.6. Команда деления	260
§7.10. Логические команды	260
7.10.1. Команды логического умножения	261
7.10.2. Команды логического сложения	261
7.10.3. Команды «исключающего ИЛИ»	261
7.10.4. Команды ротации (сдвига)	262
7.10.5. Команды очистки и инвертирования аккумулятора	263
7.10.6. Команды побитовой обработки	263
§7.11. Команды передачи управления	264
7.11.1. Общие сведения	264
7.11.2. Разновидности команд перехода	265
7.11.3. Безусловные переходы	265
7.11.4. Условные переходы	267
7.11.5. Сдвоенные условные команды перехода	268
7.11.6. Вызовы подпрограмм	269
§7.12. Оформление ассемблерной программы	270
7.12.1. Структура ассемблерной программы	270
7.12.2. Определение констант и распределение внешних выводов	271
7.12.3. Резервирование переменных в памяти данных	272
7.12.4. Заполнение особых ячеек в ПЗУ	272
7.12.5. Текст основной части программы	273
7.12.6. Подпрограмма и макрос	273
7.12.7. Тексты используемых процедур	274
7.12.8. Размещение массивов констант в ПЗУ	274
7.12.9. Общие советы	275
 ГЛАВА 8. Управление внутренней аппаратурой ОВМ x51	 276
§8.1. Вывод информации битовыми командами	276
8.1.1. Особенности адресации битов	276
8.1.2. Установка, сброс и инвертирование отдельных битов	277
8.1.3. Вывод (копирование) групп битов	277
8.1.4. Вывод битов в последовательном формате	278
§8.2. Ввод информации битовыми командами	279
8.2.1. Ввод (чтение) одиночного бита	279
8.2.2. Чтение и анализ бита	280
8.2.3. Чтение групп битов из разных портов	281
8.2.4. Чтение последовательного формата битов	281
§8.3. Ввод-вывод информации байтовыми командами	282
8.3.1. Вывод (запись) байтовых констант и переменных	282
8.3.2. Установка групп битов байтовыми командами	283

8.3.3. Сброс групп битов байтовыми командами	284
8.3.4. Инвертирование групп битов байтовыми командами	285
8.3.5. Ввод (чтение) байтовой информации.....	285
8.3.6. Чтение групп битов байтовыми командами	285
§8.4. Управление прерываниями.....	286
8.4.1. Общие сведения.....	286
8.4.2. Регистр разрешения прерываний IE	286
8.4.3. Регистр управления приоритетами IP.....	287
8.4.4. Управление типом внешних прерываний	287
8.4.5. Запросы прерываний	288
§8.5. Пример программы для внешнего прерывания.....	288
8.5.1. Организация внешнего прерывания	288
8.5.2. Пример процедуры внешнего прерывания	289
8.5.3. Использование процедуры прерывания	290
§8.6. Регистры и возможности таймеров.....	290
8.6.1. Регистр управления таймерами TCON.....	290
8.6.2. Формат регистра режима TMOD.....	291
8.6.3. Инициализация (загрузка) регистра TMOD	292
8.6.4. Загрузка регистров данных.....	293
8.6.5. Чтение регистров данных.....	293
§8.7. Счет времени при помощи таймеров.....	294
8.7.1. Общие сведения.....	294
8.7.2. 16-битный счет времени	295
8.7.3. 8-битный счет времени с перезагрузкой	297
8.7.4. Отсчет времени без использования прерывания	297
8.7.5. Отсчет времени с использованием прерываний	298
§8.8. Управление последовательным портом.....	300
8.8.1. Общие сведения.....	300
8.8.2. Назначение битов регистра управления SCON	301
8.8.3. Управление режимами последовательного порта	302
8.8.4. Режим аппаратного различения 9 бита при приеме	304
8.8.5. Бит последовательного порта в регистре управления мощностью PCON.....	305
§8.9. Инициализация последовательного порта	306
8.9.1. Выбор режима тактирования последовательного порта	306
8.9.2. Использование таймера T1 для тактирования последовательного порта.....	307
8.9.3. Погрешность задания стандартной скорости.....	308
8.9.4. Пример инициализации последовательного порта.....	309
§8.10. Процедуры для последовательного порта	310
8.10.1. Программное ожидание посылки.....	310
8.10.2. Прием по прерыванию	310
8.10.3. Передача байта без использования прерываний	311

8.10.4. Передача с ожиданием окончания посылки.....	312
8.10.5. Передача пакета данных по прерыванию	312

ГЛАВА 9. Особенности программирования ОВМ х51 на языке Си 314

§9.1. Директивы #include и #define	314
9.1.1. Общие сведения	314
9.1.2. Особенности включаемых файлов.....	315
9.1.3. Директива препроцессора #define	316
§9.2. Ресурсы ОВМ х51 для языка Си	317
9.2.1. Общие сведения.....	317
9.2.2. Указание места размещения переменных.....	318
9.2.3. Размещение локальных переменных	319
9.2.4. Работа со спецрегистрами.....	319
9.2.5. Ассемблерные вставки	320
§9.3. Объявления переменных и констант	321
9.3.1. Объявления переменных.....	321
9.3.2. Объявление не удаляемых локальных переменных	322
9.3.3. «Изменчивые» переменные типа volatile	322
9.3.4. Объявления констант	323
9.3.5. Объявления многобайтовых переменных и массивов	323
§9.4. Объявления функций	324
9.4.1. Общие сведения.....	324
9.4.2. Функция, не получающая и не возвращающая данные.....	325
9.4.3. Объявление функции, получающей параметры.....	325
9.4.4. Объявление функции, возвращающей значение	325
9.4.5. Бесконечный цикл в главной функции	326
9.4.6. Пример записи простейшей программы	326
§9.5. Адресуемые биты ОВМ х51.....	327
9.5.1. Общие сведения.....	327
9.5.2. Объявления битовых переменных.....	328
9.5.3. Операции с битовыми переменными	329
§9.6. Типовые преобразования данных	330
9.6.1. Битовые операции для целочисленных операндов.....	330
9.6.2. Сдвиги переменных	331
9.6.3. Преобразование «коротких» типов переменных в «длинные».....	332
9.6.4. Преобразование бита в целое число	332
9.6.5. Преобразование «длинных» типов переменных в «короткие».....	333
9.6.6. Преобразование целого числа в бит	334
§9.7. Работа с отдельными битами целых чисел	334
9.7.1. Указание места бита в целочисленной переменной.....	334
9.7.2. Образование маски для нескольких значащих битов	335
9.7.3. Установка отдельных битов целочисленного операнда.....	335
9.7.4. Сброс незначащих битов	336

9.7.5. Инвертирование отдельных битов.....	336
9.7.6. Обмен частей переменных	337
9.7.7. Объединение (упаковка) битов разных переменных	337
9.7.8. Разъединение (распаковка) переменной на биты.....	338
§9.8. Процедуры задержки	338
9.8.1. Общие сведения.....	338
9.8.2. Реализация микросекундных задержек	339
9.8.3. Реализация задержки при помощи оператора for.....	340
9.8.4. Оценка времени задержки mDelayFOR.....	340
9.8.5. Реализация задержки при помощи оператора do-while	341
9.8.6. Эмуляция «паскалевской» процедуры Delay.....	342
9.8.7. Использование и недостатки процедур задержки.....	342
§9.9. Примеры программ преобразования кодов.....	343
9.9.1. Вычисление контрольной суммы массива во внешнем ОЗУ	343
9.9.2. Прием данных в буфер со сдвигом.....	344
9.9.3. Скользящее осреднение результатов оцифровки	344
 ГЛАВА 10. Управление ресурсами x51 на языке Си.....	 346
§10.1. Вывод-ввод информации битовыми командами	346
10.1.1. Генерация пачки импульсов.....	346
10.1.2. Генерация звука «бип-бип»	347
10.1.3. Ввод сигнала от контактного датчика	347
10.1.4. Счет числа нажатий кнопки с «дребезгом»	348
§10.2. Вывод-ввод байтовыми командами	349
10.2.1. Управление простейшим светофором	349
10.2.2. Программа для простейшего светофора	350
10.2.3. Вычисление скан-кода матрицы ключей.....	351
10.2.4. Двухнаправленный опрос матрицы ключей.....	352
10.2.5. Текст программы для двухнаправленного опроса	353
§10.3. Регистры и биты системы прерывания.....	353
10.3.1. Управление разрешениями и приоритетами.....	353
10.3.2. Биты типа внешних прерываний	354
10.3.3. Биты запросов прерываний	355
§10.4. Оформление прерывающих процедур	356
10.4.1. Объявление прерывающей процедуры	356
10.4.2. Результат работы прерывающих процедур.....	357
10.4.3. Инициализация прерываний	357
10.4.4. Пример процедуры для счета внешних импульсов	357
10.4.5. Измерение частоты внешних импульсов	358
§10.5. Регистры, биты и режимы таймеров	358
10.5.1. Регистр TCON и пуск-останов счета	359
10.5.2. Формат регистра TMOD и задание режима.....	359
10.5.3. Задание режима счета	360

10.5.4. Изменение режима только одного из таймеров	361
10.5.5. Загрузка 16-разрядных регистров данных таймеров	362
10.5.6. Чтение регистров данных таймеров	362
§10.6. Формирование интервалов времени при помощи таймеров ...	363
10.6.1. Общие сведения	363
10.6.2. Определение кода загрузки таймера.....	363
10.6.3. Отсчет заданного времени без использования механизма прерываний	364
10.6.4. Пример инициализации таймера при работе в режиме прерывания.....	364
10.6.5. Отсчет времени с использования механизма прерываний	365
10.6.6. Использование таймера для отсчета одной секунды	366
10.4.7. Использование флажков-семафоров.....	367
§10.7. Измерение длительности и частоты импульсов	367
10.7.1. Внешнее управление счетом таймера	367
10.7.2. Программа для измерения длительности импульса соответствует вышеописанной последовательности действий:.....	368
10.7.3. Счет переполнений таймера при измерении длинных импульсов	369
10.7.4. Использование механизма прерываний для определения начала и окончания импульса.....	370
10.7.5. Измерение частоты импульсов	370
§10.8. Ресурсы последовательного порта	371
10.8.1. Регистры данных приемника и передатчика.....	372
10.8.2. Регистр управления последовательным портом.....	372
10.8.3. Биты задания режима работы последовательного порта (SM0, SM1, SM2, REN).....	373
10.8.4. Особенности работы с девятым битом.....	374
§10.9. Инициализация последовательного порта.....	375
10.9.1. Общие сведения	375
10.9.2. Задание режима работы последовательного порта битовыми командами.....	375
10.9.3. Задание режима работы последовательного порта байтовой командой	376
10.9.4. Задание скорости и инициализация таймера 1	376
10.9.5. Удвоение скорости приема-передачи.....	377
10.9.6. Пример инициализации последовательного порта	378
§10.10. Процедуры приема для последовательного порта	378
10.10.1. Программное ожидание послыки	378
10.10.2. Прием по прерыванию одного байта	379
10.10.3. Прием по прерыванию заданного количества байтов	380
10.10.4. Прием строки символов по прерыванию	381
10.10.5. Примеры простейшей обработки принятой информации	381

§ 10.11. Процедуры передачи для последовательного порта	382
10.11.1. Передача одного байта без прерывания.....	382
10.11.2. Передача байта с ожиданием окончания посылки	382
10.11.3. Передача пакета данных без прерывания	383
10.11.4. Передача пакета по прерыванию.....	383
10.11.5. Передача строки символов по прерыванию	384
10.11.6. Примеры подготовки пакета для передачи	385
 ГЛАВА 11. Комбинационные узлы.....	386
§11.1. Дешифраторы	386
11.1.1. Общие сведения	386
11.1.2. Иллюстративная модель дешифратора	387
11.1.3. Таблица и уравнения для простейшего дешифратора	387
11.1.4. Схема линейного дешифратора	388
11.1.5. УГО и микросхемы дешифраторов	389
11.1.6. Каскадное соединение дешифраторов.....	390
11.1.7. Прямоугольный (матричный) дешифратор.....	392
§11.2. Применение микросхем дешифраторов	393
11.2.1. Дешифратор в микропроцессорной системе.....	393
11.2.2. Программная реализация дешифратора.....	394
11.2.3. Реализация табличной функции	395
11.2.4. Дешифратор для управления банками памяти	396
11.2.5. Дешифратор адреса для внешних устройств	397
§11.3. Дешифраторы для управления светодиодными индикаторами	399
11.3.1. Семисегментные индикаторы	399
11.3.2. Типы семисегментных светодиодных индикаторов	400
11.3.3. Дешифратор K514ИД1	401
11.3.4. Дешифратор КР514ИД2, ...	402
11.3.5. Обслуживание однокладного индикатора	403
11.3.6. Обслуживание двухккладного индикатора	403
§11.4. Дешифраторы в динамических индикаторах.....	404
11.4.1. Схема динамического индикатора.....	404
11.4.2. Временные диаграммы динамического индикатора.....	406
11.4.3. Реализация периодического обслуживания динамического индикатора.....	406
11.4.4. Подготовка данных для индикации	407
11.4.5. Двоично-десятичное преобразование байта.....	407
11.4.6. Процедура индикации знакоместа mIndZn3.....	408
§11.5. Программное управление сегментами одиночного индикатора	409
11.5.1. Схема.....	409

11.5.2. Таблицы для преобразования тетрады в код управления семисегментным индикатором	410
11.5.3. Ассемблерная программа для табличного преобразования кодов	411
11.5.4. Программа управления сегментами индикатора	411
§11.6. Динамический индикатор с программно-управляемыми сегментами	412
11.6.1. Схема четырехзначного динамического индикатора, ...	412
11.6.2. Подготовка данных в буфере индикации	413
11.6.4. Программа индикации знакоместа на Си	413
§11.7. Шифраторы	414
11.7.1. Общие сведения	414
11.7.2. Схема линейного шифратора на 8 входов	415
11.7.3. Каскадный шифратор с 16 входами	415
11.7.4. Микросхема КР155ИВ1	417
11.7.5. Программная реализация функции шифратора	418
§11.8. Коммутаторы	419
11.8.1. Общие сведения	419
11.8.2. Аналоговый и логический ключи	419
11.8.3. Реализация коммутатора	420
11.8.4. Каскадное соединение коммутаторов	420
11.8.5. Неявный коммутатор на основе элементов с открытым коллектором	421
11.8.6. Неявный коммутатор на основе элементов с третьим состоянием	422
11.8.7. Реализация логической табличной функции	423
11.8.8. Управление внешним коммутатором	423
§11.9. Сумматоры	424
11.9.1. Полусумматор (сумматор по модулю 2)	425
11.9.2. «Исключающее ИЛИ», как управляемый инвертор	426
11.9.3. Полный сумматор	426
11.9.4. Многоразрядные сумматоры	427
11.9.5. Арифметико-логическое устройство	428
§11.10. Узлы контроля	428
11.10.1. Цифровой компаратор	428
11.10.2. УГО микросхемы цифрового компаратора	429
11.10.3. Схемы контроля по четности	429
11.10.4. Искатели старшей единицы	431
11.10.5. Мажоритарный элемент	432
ГЛАВА 12. Триггеры и регистры	434
§12.1. Триггеры RS-типа	434
12.1.1. Общие сведения	434

12.1.2. Асинхронный RS-триггер	435
12.1.3. Применение RS-триггера для подавления дребезга.....	436
12.1.4. Тактируемый уровнем RS-триггер	437
12.1.5. Тактируемый фронтом (перепадом) RS-триггер.....	437
12.1.6. RS-триггеры с приоритетными входами	438
§12.2. Триггеры D-типа	438
12.2.1. Тактируемый уровнем D-триггер.....	438
12.2.2. Тактируемый фронтом триггер D-типа.....	439
12.2.3. Микросхема триггера D-типа, тактируемого фронтом	440
§12.3. Триггеры Т- и JK-типов.....	440
12.3.1. Асинхронный триггер Т-типа.....	440
12.3.2. Синхронный счетный триггер.....	441
12.3.3. JK-триггер.....	442
12.3.4. Микросхема JK триггера.....	442
§12.4. Регистры памяти.....	443
12.4.1. Общие сведения	443
12.4.2. Обозначение регистра памяти.....	444
12.4.3. Управление приемом информации.....	444
12.4.4. Управление выдачей информации.....	445
§12.5. Регистры сдвига	446
12.5.1. Общие сведения	446
12.5.2. Регистр сдвига вправо	446
12.5.3. Реверсивный регистр сдвига.....	447
12.5.4. Микросхема реверсивного регистра КР155ИР13	448
12.5.5. Регистровая память типа очередь	448
§12.6. Регистры сдвига в асинхронном приемопередатчике	450
12.6.1. Передатчик асинхронного последовательного порта.....	450
12.6.2. Программная передача асинхронной посылки	451
12.6.3. Приемник асинхронного последовательного порта	451
12.6.4. Программный прием асинхронной посылки	452
§12.7. Регистр сдвига в синхронном клавиатурном интерфейсе	453
12.7.1. Передача битов в синхронном виде.....	453
12.7.2. Приемник сигналов от клавиатуры.....	453
12.7.3. Программная передача синхронной посылки	454
12.7.4. Программный прием синхронной посылки	455
§12.8. Регистр сдвига в дуплексном синхронном интерфейсе типа SPI	455
12.8.1. Приемник и передатчик интерфейса SPI.....	455
12.8.2. Пример программы приема-передачи для SPI.....	457
 ГЛАВА 13. Счетчики.....	 458
§13.1. Двоичные счетчики.....	458
13.1.1. Асинхронный двоичный счетчик.....	458
13.1.2. Обозначение асинхронного счетчика	459

13.1.3. Синхронный двоичный счетчик	460
13.1.4. Асинхронный реверсивный счетчик	461
13.1.5. Синхронный реверсивный счетчик	461
13.1.6. УГО синхронного реверсивного счетчика	463
§13.2. Счетчики с программируемым коэффициентом пересчета...	463
13.2.1. Счетчики с двоично-взвешенным коэффициентом пересчета	463
13.2.2. Программируемый инкрементный счетчик с асинхронным сбросом	464
13.2.3. Программируемый инкрементный счетчик с предустановкой	465
13.2.4. Программируемый декрементный счетчик с предустановкой	466
13.2.5. Программируемый счетчик с цифровым компаратором	467
§13.3. Двоично-десятичные и часовые счетчики	467
13.3.1. Двоично-десятичные счетчики	467
13.3.2. Микросхема асинхронного двоично-десятичного счетчика КР155ИЕ2.....	468
13.3.3. Микросхема синхронного реверсивного двоично-десятичного счетчика КР155ИЕ6	468
13.3.4. Микросхема асинхронного часового счетчика КР155ИЕ4.....	469
13.3.5. Схема счетчика секунд или минут	469
13.3.6. Микросхема цифровых часов с параллельным интерфейсом	470
13.3.7. Микросхема цифровых часов с последовательным интерфейсом	471
§13.4. Некоторые применения счетчиков	471
13.4.1. Принцип измерения длительности импульсов и частоты	471
13.4.2. Распределитель тактов.....	472
13.4.3. Командный аппарат	473
 ГЛАВА 14. Микросхемы памяти и их использование.....	 475
§14.1. Микросхемы памяти	475
14.1.1. Общие сведения	475
14.1.2. Внутренняя структура микросхем памяти	476
14.1.3. Разновидности запоминающих ячеек ПЗУ	477
14.1.4. Типы электрически программируемых ПЗУ	477
14.1.5. Ресурс работы программируемых ПЗУ	478
14.1.6. Интерфейсы микросхем ППЗУ	480
14.1.7. Разновидности ячеек ОЗУ	481
§14.2. Некоторые применения ПЗУ	482
14.2.1. Реализация табличных функций	482
14.2.2. Реализация командоаппарата.....	482
14.2.3. Микропрограммный автомат	483
14.2.3. Цифро-аналоговый генератор	483
§14.3. Программируемые логические ИС	484
14.3.1. Общие сведения	484

14.3.2. Принцип действия программируемого комбинационного устройства.....	485
14.3.3. Составные части ПЛИС.....	487
14.3.4. Конфигурирование ПЛИС.....	488
14.3.5. Программируемые аналоговые ИС.....	489
14.3.6. Система на кристалле.....	489
§14.4. Обмен данными между процессором, ЗУ и ВУ	490
14.4.1. Средства для обмена.....	490
14.4.2. Основные механизмы обмена данными.....	491
14.4.3. Синхронный программный обмен.....	492
14.4.4. Асинхронный программный обмен.....	494
§14.5. Подключение регистров и памяти к процессору	495
14.5.1. Чтение данных из регистра.....	495
14.5.2. Чтение данных из ПЗУ.....	496
14.5.3. Запись данных во внешний регистр.....	497
14.5.4. Подключение ОЗУ для чтения и записи данных.....	497
§14.6. Подключение внешней памяти к ОВМ.....	498
14.6.1. Шины ОВМ для подключения внешней памяти.....	498
14.6.2. Подключение внешнего ПЗУ к ОВМ.....	499
14.6.3. Подключение внешнего ОЗУ к ОВМ.....	500
14.6.4. Программное использование внешнего ОЗУ.....	501
14.6.5. Внешнее ОЗУ в качестве памяти команд.....	502
§14.7. Подключение к ОВМ в режиме 8-битной адресации	503
14.7.1. Два типа адресации внешней памяти.....	503
14.7.2. Подключение микросхемы часов.....	504
14.7.3. Подключение АЦП к порту P0.....	505
14.7.4. Подключение ЖКИ с параллельным интерфейсом.....	506
ГЛАВА 15. Аналоговые ключи и коммутаторы.....	508
§15.1. Аналоговый ключ.....	508
15.1.1. Общие сведения.....	508
15.1.2. Проводящее состояние ключа.....	509
15.1.3. Непроводящее состояние ключа.....	510
15.1.4. Динамическая погрешность ключа.....	511
15.1.5. Обозначение аналогового ключа.....	512
15.1.6. Выбор микросхемы ключа.....	512
§15.2. Аналоговый коммутатор.....	513
15.2.1. Внутренняя структура и обозначение.....	513
15.2.2. Параметры.....	514
15.2.3. Пример использования аналогового коммутатора.....	514
15.2.4. Разновидности микросхем аналоговых коммутаторов.....	516
§15.3. Схема выборки-хранения.....	516
15.3.1. Схема и принцип действия.....	516

15.3.2. Погрешность недозаряда.....	517
15.3.3. Погрешность хранения	518
15.3.4. Микросхема выборки-хранения КР1100СК2.....	518
§15.4. Аналоговые ключи с изолированным управлением	519
15.4.1. Ключ с трансформаторным управлением.....	519
15.4.2. Ключ с оптическим управлением.....	520
15.4.3. Контактные ключи	520
15.4.4. «Летающий конденсатор».....	522
 ГЛАВА 16. Цифро-аналоговые преобразователи	 524
§16.1. Структура ЦАП	524
16.1.1. Общие сведения	524
16.1.2. Основные параметры ЦАП	525
16.1.3. Структура ЦАП	526
16.1.4. ЦАП на основе двоично-взвешенных резисторов.....	526
16.1.5. ЦАП на основе резисторной матрицы R-2R	528
§16.2. Микросхемы ЦАП и их интерфейсы.....	529
16.2.1. Обозначение ЦАП	529
16.2.2. Параллельный интерфейс ЦАП	529
16.2.3. Последовательный интерфейс типа SPI.....	531
16.2.4. Последовательный интерфейс типа Up/Down	532
16.2.5. Области использования ЦАП	533
§16.3. Время-импульсный ЦАП.....	534
16.3.1. Общие сведения	534
16.3.2. Формирование ШИМ сигнала.....	535
16.3.3. Аппаратное формирование ШИМ сигнала	536
16.3.4. Сглаживающий фильтр	537
 ГЛАВА 17. Пороговые устройства и генераторы импульсов.....	 539
§17.1. Аналоговые компараторы.....	539
17.1.1. Общие сведения	539
17.1.2. Основные параметры компаратора	541
17.1.3. Схемы сравнения	542
17.1.4. Двухпороговый компаратор.....	543
17.1.5. Формирование прямоугольного сигнала	544
§17.2. Пороговое устройство с гистерезисом (триггер Шмитта)	545
17.2.1. Общие сведения	545
17.2.2. Формирование прямоугольного импульса.....	545
17.2.3. Логический элемент с гистерезисом	546
17.2.4. Прецизионный триггер Шмитта	547
§17.3. Микросхема таймера КР1006ВИ1.....	548
17.3.1. Общие сведения	548

17.3.2. Структура микросхемы таймера	549
17.3.3. Обозначение таймера	550
17.3.4. Основные параметры таймера КР1006ВИ1	551
17.3.5. Применение таймера в качестве порогового устройства	552
§17.4. Генератор одиночного импульса на таймере	552
17.4.1. Схема и временные диаграммы	553
17.4.2. Определение длительности импульса	554
17.4.3. Порядок расчета одновибратора	555
§17.5. Автоколебательные генераторы импульсов на таймере ...	555
17.5.1. Генератор несимметричных импульсов	555
17.5.2. Определение частоты несимметричных импульсов	557
17.5.3. Порядок расчета генератора несимметричных импульсов	558
17.5.4. Генератор симметричных импульсов	558
17.5.5. Определение частоты симметричных импульсов	559
17.5.6. Генератор на основе логического инвертора с гистерезисом	560
17.5.7. Частотные преобразователи для датчиков	561
 ГЛАВА 18. Аналого-цифровые преобразователи	 562
§18.1. Общие сведения и параллельный АЦП	562
18.1.1. Основные параметры АЦП	562
18.1.2. Параллельный АЦП	563
18.1.3. Последовательно-параллельный АЦП	564
§18.2. АЦП на основе ЦАП и компаратора	565
18.2.1. Общие сведения	565
18.2.2. Развертывающий алгоритм	565
18.2.3. Следящий алгоритм	566
18.2.4. Алгоритм поразрядного уравнивания	567
§18.3. Особенности обслуживания микросхем АЦП	568
18.3.1. Обслуживание АЦП К572ПВЗ	568
18.3.2. АЦП с последовательным интерфейсом	569
18.3.3. Программа обслуживания АЦП с последовательным интерфейсом	571
18.3.4. Особенности встроенных АЦП	571
§18.4. Время-импульсный АЦП развертывающего типа	572
18.4.1. Основные сведения	572
18.4.2. Схема простейшего время-импульсного АЦП	573
18.4.3. Расчетные соотношения	574
18.4.4. Уменьшение погрешности	575
§18.5. АЦП двойного интегрирования	576
18.5.1. Общие сведения	576
18.5.2. Реализация АЦП двойного интегрирования	577
§18.6. Частотные и сигма-дельта АЦП	579
18.6.1. Частотные АЦП	579

18.6.2. Преобразователь напряжения в частоту (ПНЧ)	579
18.6.3. Структура сигма-дельта АЦП	580
18.6.4. Передискретизация	581
18.6.4. Замена однобитовых устройств	582

Список используемых сокращений	584

Литература	585

Введение

Всевозможные системы обработки данных состоят из большого количества электронных устройств различного назначения. Эти устройства выполнялись в свое время на отдельных транзисторах, затем на микросхемах малой и средней степени интеграции. В настоящее время практически любую систему можно реализовать в виде большой интегральной схемы (БИС). Но разрабатывать для каждой из великого множества систем свою БИС экономически нецелесообразно. Дело в том, что себестоимость производства микросхем определяет их тираж, который в свою очередь зависит от степени универсальности изделия. Поэтому в настоящее время стараются по возможности унифицировать элементную базу, используя два пути реализации универсального схемотехнического устройства.

Первый – аппаратный, когда изготавливается интегральная схема в виде набора логических и аналоговых элементов, которые коммутируются пользователем при программировании, определяя заданную схему. Заметим, что в этом случае достигается предельное для используемых элементов быстродействие.

Второй путь – программный, когда необходимая функция по обработке данных в виде программного алгоритма заносится в память однокристалльной ВМ (ОВМ), которая под управлением программы реализует необходимые операции. Быстродействие в этом случае существенно хуже, поскольку необходимые функции приходится реализовывать последовательно во времени.

Однако на практике в диапазоне относительно низких рабочих частот существует огромный круг задач, где применение ОВМ экономически и технически оправдано, благодаря вычислительному потенциалу процессорного ядра.

Отметим также, что в настоящее время происходит естественное слияние двух путей создания универсальных микросхем, аппаратного и программного, в рамках структуры «система на кристалле» (СнК), когда реализуют программируемые цифровые и аналоговые цепи вместе с процессорным ядром на одном кристалле. Поэтому изучение основ и принципа действия ОВМ совершенно необходимо современному специалисту уже на начальном этапе изучения схемотехники.

Пособие логически разбита на шесть частей, но имеет сквозную нумерацию глав, параграфов и пунктов, чтобы обеспечить перекрестные ссылки, облегчающие восприятие материала. Кроме того, мелкая дозировка материала облегчает первоначальное изучение, чтение «с середины» и поиск необходимых для работы сведений, ведь основной целью пособия было «пособить» (sic!) самостоятельной работе над курсовым или выпускным проектом.

В первой части (главы 1, 2, 3 и 4) рассмотрены ключевые и логические элементы ВМ. Особое внимание уделено практическим вопросам формирования входных сигналов и подключению нагрузки (в том числе и индуктивной, в виде импульсного трансформатора) к выходу микросхем.

Вторая часть (главы 5 и 6) предназначена для первоначального ознакомления со структурой и основными особенностями интегрального операционного усилителя (ИОУ). Изложены базовые сведения о схемотехнике входных и выходных цепей ИОУ. Приведены описания основных схем включения ИОУ, примеры типовых схем усилителей и порядок их практического расчета. Приведены условия возникновения генерации в усилительных устройствах, описаны кварцевые генераторы, приведены основные особенности их практического использования, например, при выборе длительности импульса системного сброса.

Третья часть (главы 7 и 8) посвящена основам схемотехники однокристалльной ВМ типа x51, ставшей промышленным стандартом для 8-битных процессоров. Изложены базовые сведения о схемотехнике выходных цепей и внутренних узлов ОВМ. Чтобы более наглядно иллюстрировать процессы, происходящие внутри ОВМ, принципы действия ее внутренних узлов поясняются при помощи средств языка ассемблера. Поэтому приведено описание системы команд, особенностей оформления ассемблерных программ, примеры типовых процедур, которые могут быть использованы при курсовом проектировании.

Четвертая часть (главы 9 и 10), по сути, повторяет предыдущую в части касающейся программирования, но на языке Си, который позволяет сравнительно легко переходить к программированию других семейств ОВМ. Кроме того язык высокого уровня позволяет достаточно просто пояснять принцип действия некоторых узлов.

Пятая часть (главы 11, 12, 13 и 14) содержит материал, пожалуй, наиболее совпадающий с традиционными книгами по схемотехнике. Здесь рассмотрены структура и основные особенности базовых схемотехнических узлов. Изложены основные сведения о принципах

действия и реализации, как комбинационных (дешифраторов, сумматоров, коммутаторов), так и последовательностных схем с памятью (триггеров, регистров, счетчиков). Приведены примеры и особенности практического использования подобных узлов в составе микропроцессорных систем. Особое внимание уделено пояснению структуры микросхем памяти и ПЛИС, а также принципов взаимодействия микропроцессора с его обрaмление при помощи трех шин, адреса, данных и управления.

В шестой части (главы 15, 16, 17 и 18) описываются основные устройства, сочетающие как аналоговые, так и цифровые цепи, применяемые в микропроцессорной технике для взаимодействия с аналоговым миром, окружающим процессор. К таким устройствам относятся аналоговые коммутаторы, пороговые устройства (с гистерезисом и без), генераторы прямоугольных импульсов на основе микросхемы аналогового таймера, цифро-аналоговые преобразователи и аналого-цифровые преобразователи.

Пособие не претендует на всеобъемлющий охват материала. Принципы действия устройств рассматриваются упрощенно, с позиции пользователя, а не разработчика. Основной упор сделан на возможном сочетании аппаратных и программных средств при реализации заданных функций.

Большинство устройств и программ, приведенных в пособии, были отмакетированы и отлажены в ходе лабораторных занятий, а также курсовых и выпускных работ в Смоленском филиале Московского Энергетического Института. Но, если, тем не менее, в пособии будут обнаружены ошибки, просьба сообщать о них по адресу aver22@rambler.ru.

Автор выражает искреннюю признательность за предварительную экспертизу пособия и указанные замечания В. Ф. Жиркову и В. Я. Хартову, сотрудникам кафедры «Компьютерные системы и сети» Московского Государственного Технического Университета имени Баумана.

1

КЛЮЧЕВЫЕ СХЕМЫ

Ключи являются основой логических элементов, которые используются в более сложных схемотехнических устройствах. В этом разделе приводятся первоначальные сведения о принципах действия, характеристиках и особенностях ключевых элементов.

§1.1. Базовая ключевая схема

1.1.1. Схема с идеальным ключом

Рассмотрим свойства и особенности ключевых устройств на основе простейшей схемы (рис. 1.1) с идеальным ключом K , который в зависимости от логического значения управляющего сигнала U_{BX} имеет два состояния, проводящее и непроводящее.

В проводящем состоянии внутреннее сопротивление ключа $r_{\text{квн}}$ практически равно нулю и ключ, по сути, эквивалентен «закоротке». В непроводящем состоянии значение $r_{\text{квн}}$ очень большое ($r_{\text{квн}} \rightarrow \infty$) и ключ можно рассматривать, как разрыв.

Все сигналы в большинстве случаев отсчитываются относительно общей точки (в просторечии «земли»). Если указан только один полюс, то по умолчанию непоказанный полюс подключен к земле (пунктир на рис. 1.1).

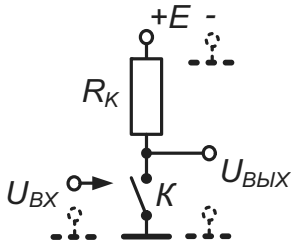


Рис. 1.1

Кроме ключа схема имеет источник питания с напряжением E и резистор R_K , благодаря которым образуется выходное напряжение. Сопротивление R_K вдобавок ограничивает ток через ключ (см. п. 1.1.3) и задаёт величину единичного значения $U_{\text{ВЫХ}}$ (см. п. 1.1.4).

1.1.2. Временная диаграмма выходного напряжения

Ключ под управлением входного сигнала периодически проводит и не проводит.

Когда ключ не проводит ($r_{\text{квн}} \rightarrow \infty$), то ток через ключ течь не будет и величина $U_{\text{ВЫХ}}$ будет определяться только напряжением питания E (см. рис. 1.2). В этом случае говорят, что на выходе действует высокий (единичный) уровень напряжения, обозначаемый U^1 .

Когда ключ проводит, его сопротивление $r_{\text{квн}}$ будет близко к нулю, а значит и падение напряжения на нем и напряжение $U_{\text{ВЫХ}}$ будут практически нулевыми. Этот низкий уровень в дальнейшем будет обозначаться как U^0 .

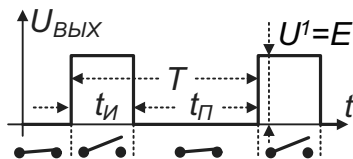


Рис. 1.2

В переводной и зарубежной литературе единичное напряжение обозначают напряжением U_H (от High – высокий), а нулевое – U_L (от Low – низкий). Но, учитывая, что вычислительные устройства работают в двоичной системе со значениями «0» и «1», более логично использовать обозначения U^0 и U^1 .

Таким образом, на выходе схемы образуются прямоугольные импульсы с амплитудой U^1 . Если эти импульсы периодические, то можно измерить их период T и определить частоту $F = 1/T$. Важными параметрами импульса, часто используемыми в расчетах, являются также длительности импульса $t_{\text{и}}$ и паузы $t_{\text{п}}$.

Кроме указанных временных параметров на практике могут использоваться коэффициент заполнения, определяемый как отношение $t_{\text{и}}/T$ и обратная ему скважность $T/t_{\text{и}}$. В коммуникационной технике также обращают внимание на широкий спектр прямоугольного сигнала и его высшие гармоники.

1.1.3. Ограничение величины внешнего резистора снизу

Ток проводящего ключа не может быть чрезмерно большим и обычно задано значение $I_{\text{МАХ}}$, которое нельзя превышать, например, исходя из требований энергопотребления устройства. Из рис. 1.1 нетрудно видеть, что для случая, когда ключ представляет «закоротку», ток ключа равен току резистора и зависит (по закону Ома) от величины внешнего резистора:

$$I_{\text{К}} = I_{\text{R}} \approx E / R_{\text{К}} \quad (1.1)$$

Отсюда, учитывая, что значение I_{MAX} превышать нельзя, можно получить неравенство, ограничивающее сопротивление резистора снизу:

$$R_K > E / I_{MAX} \quad (1.2)$$

1.1.4. Влияние сопротивления нагрузки

На выбор величины внешнего резистора R_K влияет не только рассмотренное выше токовое ограничение, но и вид нагрузки. Рассмотрим простейшую нагрузку ключевой схемы в виде активного сопротивления (рис. 1.3).

Если ключ не проводит, то ток I_R от источника питания течёт через R_K и R_H и создаёт на R_K падение напряжения. То есть величина $U'_{ВЫХ}$ становится меньше E :

$$U'_{ВЫХ} = R_H I_R \approx R_H (E / (R_K + R_H)). \quad (1.3)$$

Это напряжение, как и прежде, должно быть единичным, то есть большим $U'_{МИН}$:

$$R_H E / (R_K + R_H) > U'_{МИН} \quad (1.4)$$

На практике значения $U'_{МИН}$, R_H и E бывают заданы, поэтому последнее неравенство используют для ограничения сопротивления R_K сверху:

$$R_K < R_H (E - U'_{МИН}) / U'_{МИН} \quad (1.5)$$

Таким образом, сопротивление R_K , исходя из (1.5), не должно быть чересчур большим, чтобы обеспечить единичный сигнал на выходе. А с другой стороны согласно (1.2) не должно быть и очень маленьким, чтобы ограничить ток ключа на заданном уровне.

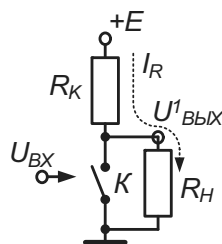


Рис. 1.3

1.1.5. Мощность, выделяемая на ключе

Замечательной особенностью идеального ключа является то, что на нем практически не выделяется мощность, хотя этот ключ может переключать значительные по величине токи и напряжения.

Как известно, мощность определяется произведением тока и напряжения. Когда ключ проводит, то падение напряжения на идеальном ключе будет нулевым, а значит, будет близка к нулю и мощность. Когда ключ не проводит и эквивалентен разрыву цепи, его ток равен нулю. То есть в обоих случаях мощность, выделяемая на ключе, близка к нулю.

Поэтому ключевой режим часто применяют при управлении мощными нагрузками, используя широтно-импульсную модуляцию (см. рис. 1.4).

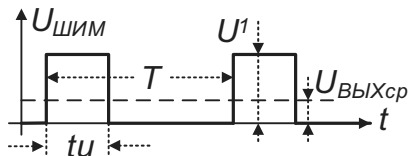


Рис. 1.4

Среднее значение импульсного напряжения (или тока)

$$U_{\text{Выхср}} = U^1 t_u / T \quad (1.6)$$

зависит от временных параметров импульса, то есть изменяя, например, величину t_u при заданном периоде T , можно управлять средним значением выходного параметра без потерь мощности на ключе.

Реальный ключ имеет не нулевые остаточные параметры, но все равно мощность, выделяемая на ключе, получается достаточно малой, поэтому КПД большинства мощных ключевых регуляторов имеет величину примерно около 90%.

1.1.6. Реальный ключ

Реальный ключ имеет конечные быстродействие и внутреннее сопротивление. Последнее означает, что в проводящем состоянии выходное напряжение станет меньше нуля (так как у реального ключа $r_{\text{Квн}} > 0$) и появится отличное от нуля напряжение U^0 (рис. 1.5), а в непроводящем — оно будет меньше расчётного $U^1_{\text{Вых}}$, определяемого по формуле (1.3), за счет остаточного тока реального ключа в непроводящем состоянии.

Кроме амплитудных параметров импульса изменится крутизна фронтов, они станут более затянутыми, что связано с инерционностью ключа, которому на переход из одного состояния в другое требуется определенное время. Поэтому реальный импульс отличается от прямоугольного и дополнительно характеризуется длительностью фронтов, положительного (производная положительна) $t_{\phi+}$ и отрицательного $t_{\phi-}$.

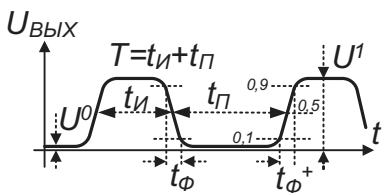


Рис. 1.5

Примечание: Иногда положительный перепад называют фронтом, а отрицательный — срезом. Но более конкретны другие обозначения: $t_{\phi+}$ и $t_{\phi-}$ — длительности перехода из «0» в «1» и из «1» в «0».

Заметим также, что сигнал не имеет резких границ, поэтому измерение длительности фронтов принято проводить на уровнях 0,1 и 0,9 от перепада сигнала, а длительность импульса и период определяют по уровню 0,5.

§1.2. Влияние емкостей на форму импульсов

1.2.1. Влияние емкости нагрузки

Специально емкость на выход ключа (рис. 1.6) включают редко (см. п. 1.2.2). Обычно она является паразитной. Например, это емкость проводов на печатной плате, паразитная входная емкость нагружающего элемента, шнура осциллографа и т. д.

Для простоты будем считать ключ идеальным и предположим, что вначале ключ проводит, то есть $U_{ВЫХ} = 0$ и емкость разряжена. Когда идеальный ключ мгновенно переходит в непроводящее состояние, напряжение на выходе равно напряжению на емкости, которое не может измениться скачком. Начнется заряд с постоянной времени $R_K C_n$ и напряжение на выходе будет по экспоненте стремиться к напряжению питания E (рис. 1.7).

Получается, что положительный фронт t_ϕ^+ при наличии емкостной нагрузки оказывается затянутым даже при мгновенном переходе ключа в непроводящее состояние, что может заметно уменьшить быстродействие схемы. Поэтому постоянную времени заряда $R_K C_n$ стараются уменьшить. Считая приближенно, что длительность фронта t_ϕ^+ экспоненциального напряжения на рис. 1.7 примерно равна двум постоянным времени ($2R_K C_n$), можно при заданном максимальном времени t_ϕ^+ ограничить сопротивление ключа:

$$R_K < t_\phi^+ / 2C_n \quad (1.7)$$

При этом величина R_K не должна быть меньше, чем вычисленная по формуле (1.2).

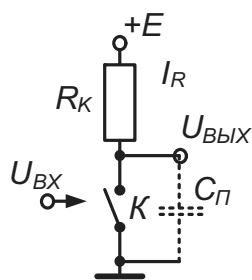


Рис. 1.6

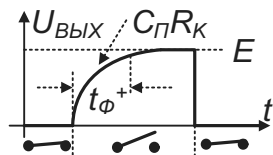


Рис. 1.7

Таким образом, паразитная емкость может изменять форму импульса, затягивая положительный фронт. Отрицательный фронт при этом почти не изменится, так как разряжаться емкость C_{Π} будет быстро через очень малое внутреннее сопротивление проводящего ключа.

1.2.2. Интегрирующая цепочка

Иногда емкость параллельно выходу включают специально, чтобы воспользоваться тем, что напряжение на ней не может скачком измениться, например, чтобы сгладить действие кратковременной импульсной помехи. Такую цепочку (рис. 1.8) называют интегрирующей (или сглаживающей) [7] и при ее использовании приходится мириться с тем, что одновременно с подавлением помехи происходит затягивание обоих фронтов выходного сигнала (см. временные диаграммы на рис. 1.9).

Другими словами интегрирующая цепочка выполняет функции фильтра нижних частот, который подавляет высокочастотные гармоники входного сигнала.

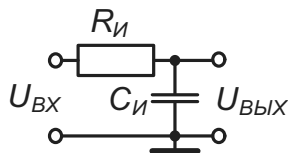


Рис. 1.8

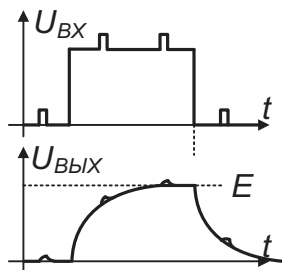


Рис. 1.9

Таким образом, при выборе постоянной времени $R_{\Pi}C_{\Pi}$ приходится идти на компромисс. С одной стороны желательно выбирать ее много большей длительности помехи, а с другой следить, чтобы фронты информационного сигнала не были чрезмерно затянуты.

1.2.3. Дифференцирующая цепочка

Изменив порядок включения резистора и конденсатора (рис. 1.10), получают другую разновидность резистивно-емкостной цепочки, которая может использоваться, например, для формирования коротких импульсов [7].

Рассмотрим работу такой цепочки, полагая, что постоянная времени $C_{\text{д}}R_{\text{д}}$ много меньше длительностей импульса и паузы, то есть конденсатор успевает быстро зарядиться и разрядиться.

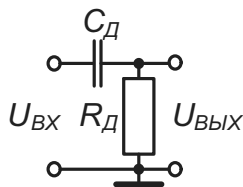


Рис. 1.10

Предположим далее, что в исходном состоянии на входе действует $U_{вх} = 0$ и будем считать, что конденсатор, благодаря малости постоянной времени, уже разрядился через сопротивление R_d , то есть напряжение на конденсаторе $U_{сд} = 0$.

Подадим далее на вход единичный сигнал U^1 (рис. 1.11а). Напряжение на емкости скачком измениться не может. Поэтому в первый момент после перепада сигнала конденсатор представляет «закоротку», и выходное напряжение будет равно входному, то есть $U_{ввых}^1 = U^1$. По мере заряда конденсатора его ток I_3 уменьшается, а напряжение на конденсаторе увеличивается (полярность указана на рис. 1.11а). Напряжение на выходе (на резисторе R_d), равное произведению $I_3 R_d$, в связи с уменьшением тока I_3 уменьшается до нуля, то есть $U_{ввых}$ достаточно быстро исчезнет, потому что заряженный конденсатор не пропускает ток и эквивалентен разрыву.

*Таким образом, при **положительном** перепаде входного сигнала ($\Delta U_{п}$ на рис. 1.11) на выходе появляется короткий **положительный** импульс.*

Пусть далее $U_{вх}$ скачком становится нулевым (рис. 1.11б). Левая (по схеме) положительно заряженная обкладка конденсатора через источник U^0 окажется подключенной к общей точке, а отрицательно заряженная (правая) обкладка будет соединена с выходом цепочки. Выходное напряжение, таким образом, становится равным напряжению на конденсаторе и будет иметь отрицательную полярность. По мере разряда конденсатора выходное напряжение будет стремиться, как и ранее, к нулю.

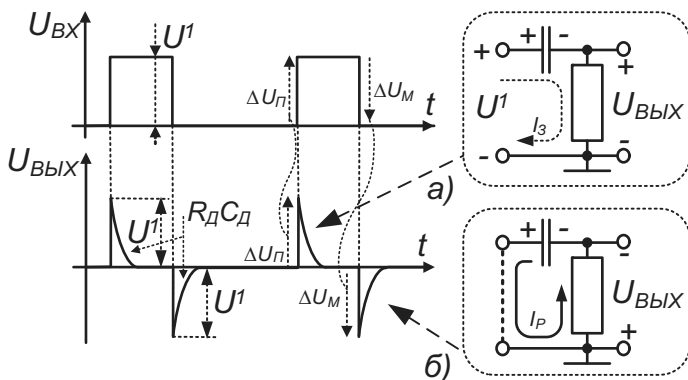


Рис. 1.11

В результате при отрицательном перепаде входного сигнала на выходе появится короткий отрицательный импульс.

Заметим, что источника энергии с отрицательным напряжением в схеме нет – функцию отрицательного источника напряжения выполняет заряженный конденсатор. Иногда эту особенность дифцепочки применяют для получения дополнительного отрицательного напряжения питания, например, для ИОУ (см. п. 6.6.5).

Отметим также, что длительность выходного импульса зависит от постоянной времени и может считаться примерно равной двум значениям $C_d R_d$. При расчете обычно бывает известно или ограничено значение R_d , тогда величину емкости конденсатора находят из соотношения:

$$C_d \approx t_{\text{и}} / 2R_d \quad (1.8).$$

Подводя итог, укажем, что в моменты перепадов входного напряжения на выходе образуются короткие импульсы, полярность и амплитуда которых соответствует полярности и амплитуде входных перепадов, то есть конденсатор в первый момент пропускает весь перепад сигнала на выход и является как бы «закороткой» для приращений. Это простейшее правило позволяет быстро рисовать выходной сигнал на основе достаточно сложного входного (рис. 1.12).

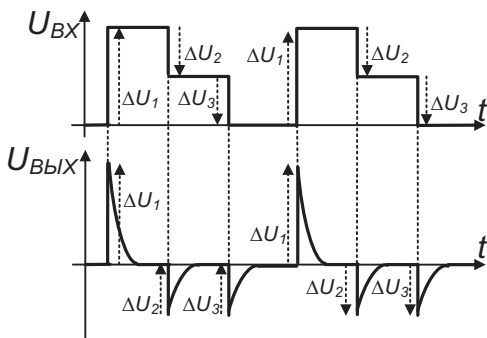


Рис. 1.12

§1.3. Свойства полупроводникового диода

1.3.1. Вольтамперная характеристика

Диод, как известно [28], имеет резко нелинейную (вентильную) характеристику (рис. 1.13а) и хорошо пропускает ток в одном направлении (прямая ветвь характеристики) и плохо в другом (обратная ветвь). Эта односторонняя проводимость отражена в условном

графическом обозначении (УГО) диода, основой которого является мнемоническое обозначение стрелки (рис. 1.13б).

Если ток идёт по «мнемострелке», то падение напряжения на диоде небольшое (около 1 В) и зависит от тока и типа диода. В обратном направлении (против «мнемострелки») диод практически не пропускает ток и значение обратного тока даже при больших напряжениях на несколько порядков меньше прямого тока.

На практике нелинейную характеристику диода использовать сложно, поэтому для упрощения расчетов нелинейный диод стараются заменить линейными элементами. Учитывая, что в цифровых (не аналоговых) схемах диод работает только в двух режимах, проводящем и непроводящем, соответственно предложены две схемы замещения диода.

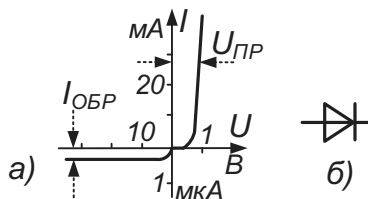


Рис. 1.13

1.3.2. Эквивалентная схема проводящего диода

В проводящем состоянии прямое падение напряжения $U_{пр}$ на диоде, как видно из его вольтамперной характеристики (рис. 1.13а) почти неизменно и практически не зависит от тока. Это позво-

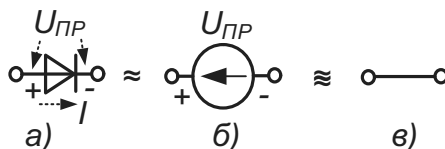


Рис. 1.14

ляет заменить нелинейный диод (рис. 1.14а) линейным элементом – источником напряжения $U_{пр} \approx 1$ В (рис. 1.14б).

Для более точных расчетов в эквивалентной схеме рис. 1.14б последовательно с источником напряжения включают резистор, отражающий дифференциальное сопротивление проводящего диода (зависимость величины $U_{пр}$ от тока) [28].

Значение $U_{пр}$ зависит от режима, типа диода, температуры и имеет достаточно большой технологический разброс. В справочниках обычно приводится $U_{пр\max}$ при максимально возможном токе. Реально напряжение $U_{пр}$ примерно наполовину меньше справочного.

То, что значение $U_{пр}$ точно неизвестно, обычно не мешает практическим оценкам, так как диод бывает включён последовательно

с источником питания и резистором, задающим ток (рис. 1.15).

В данной схеме ток от источника питания идет от плюса к минусу источника питания по «мнемострелке», то есть диод работает на прямой ветви и можно мысленно пользоваться схемой замещения рис. 1.14б. В этом случае ток диода можно определить, используя закон Ома:

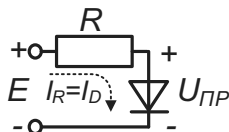


Рис. 1.15

$$I_D = I_R = U_R / R = (E - U_{пр}) / R. \quad (1.9)$$

Проведем оценку тока диода для двух значений $U_{пр}$ считая, что $E = 10 \text{ В}$, $R = 1 \text{ кОм}$. Например, при $U_{пр} = 0,7 \text{ В}$ ток будет равен $I_D = (10 - 0,7) / 1000 = 9,3 \text{ мА}$.

Пусть реальное значение $U_{пр}$ оказалось в два раза больше, то есть $U_{пр} = 1,4 \text{ В}$. В этом случае ток $I_D = (10 - 1,4) / 1000 = 8,6 \text{ мА}$. То есть при изменении $U_{пр}$ в два раза расчетный ток диода изменится менее чем на 10 %.

Из числового примера следует, что для оценки тока в подобных схемах, где выполняется неравенство $E \gg U_{пр}$ можно не искать в справочниках точное значение $U_{пр}$, а проводить расчет, например, для худшего случая.

И более того, при пояснении принципа действия ключевых устройств будем в ряде случаев считать $U_{пр} \approx 0$ и пользоваться упрощенной схемой замещения диода в виде «закоротки» (рис. 1.14в).

1.3.2. Эквивалентная схема непроводящего диода

Обратную характеристику можно аналогично аппроксимировать и представить непроводящий диод (рис. 1.16а) в виде линейного источника тока $I_{ОБР}$ (рис. 1.16б).

Значение $I_{ОБР}$ также имеет существенный технологический разброс и сильно зависит от температуры. Поэтому оценки параметров схем, в которых имеется непроводящий диод, обычно проводят с использованием эквивалентной схемы замещения на возможный худший случай, когда ток $I_{ОБР}$ максимален.

Однако при пояснениях будем пользоваться упрощенной

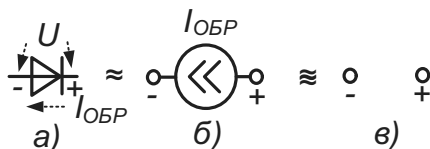


Рис. 1.16

схемой замещения, представляя непроводящий диод, эквивалентным разрыву (рис. 1.16в).

§1.4. Диодные логические элементы

1.4.1. Схема диодного ключа

Схема диодного ключа приведена на рис. 1.17 и подобно базовой ключевой схеме имеет сопротивление R_K , которое вместе с источником питания участвует в формировании выходного сигнала (см. п. 1.1.1).

Пусть напряжение U_{BX} имеет небольшой нулевой уровень U_{BX}^0 , то есть входной зажим практически соединен с общей точкой. В этом случае схема будет похожа на уже рассмотренную на рис. 1.15, и через диод от источника питания будет протекать прямой ток I_D . Поскольку диод работает в проводящем состоянии,

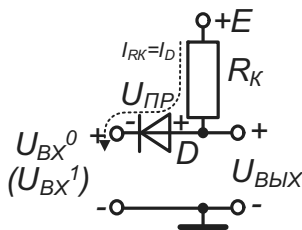


Рис. 1.17

то можно пользоваться эквивалентной схемой диода для данного случая (рис. 1.14б) и определить выходное напряжение как сумму падений напряжений на диоде $U_{пр}$ и U_{BX}^0 . Эта сумма будет иметь достаточно низкий уровень, поскольку значения $U_{пр}$ и U_{BX}^0 невелики. Поэтому выходное напряжение можно также считать имеющим нулевой уровень:

$$U_{ВЫХ} = U_{пр} + U_{BX}^0 = U_{ВЫХ}^0 \quad (1.10)$$

Пусть далее входной сигнал станет единичным (U'_{BX}), но меньшим напряжения питания E . Тогда слева и справа от диода будут одинаковые полярности напряжения и ток в цепи будет определяться разностью E и несколько меньшего по величине U'_{BX} . Поэтому диод по-прежнему будет проводить, а выходной сигнал будет равен сумме единичного входного и малого $U_{пр}$, то есть тоже будет единичным:

$$U_{ВЫХ} = U_{пр} + U'_{BX} = U'_{ВЫХ} \quad (1.11)$$

Если единичное напряжение U'_{BX} будет больше E , то диод будет находиться в непроводящем состоянии и выходное напряжение будет близко к величине E .

Таким образом, диодный ключ позволяет управлять выходным напряжением и является *повторителем* входного логического сигнала.

При расчёте диодного ключа надо найти сопротивление R_K (см. §1.1) и выбрать диод (по быстродействию, прямому току и обратному напряжению). В чистом виде рассмотренная схема применяется редко, но является основой диодных логических элементов.

1.4.2. Диодный логический элемент «И» (AND)

выполняет логическое умножение входных сигналов, то есть сигнал на выходе будет единичным, когда сигналы на всех входах будут единичными. Иными словами, если на входах будет хотя бы один ноль, то выходной сигнал тоже будет нулевым. Для примера в таблице 1 приведены выходные значения для случая двух входов.

Таблица 1

Входы		Выход
X_1	X_2	Y
0	0	0
0	1	0
1	0	0
1	1	1

Для реализации подобного двухвходового логического элемента «И» можно использовать схему на основе двух диодных ключей, работающих на общий нагрузочный резистор R_K (рис. 1.18).

Проверим работу схемы, используя различные комбинации из таблицы 1. Пусть оба входных сигнала U_1 и U_2 будут нулевыми, то есть входные зажимы, а значит и диоды подключены, по сути, к общей точке. В этом случае через них протекают прямые токи (по «мнемострелке»), диоды оказываются в проводящем состоянии, и выходной сигнал будет в соответствии с выражением (1.10) иметь низкий уровень логического сигнала.

Пусть U_2 станет единичным и диод D_2 под действием своего входного сигнала станет непроводящим (ток от источника U_2 будет направлен против его «мнемострелки»). Однако поскольку $U_1 = 0$, состояние диода D_1 останется прежним (проводящим) и выходной сигнал будет им шунтироваться (коротиться) на общую точку. То есть выходной сигнал при наличии хотя бы одного нуля на входах будет иметь низкий нулевой уровень, так как соответствующий диод будет проводить и вместе с нулем на входе фиксировать низкий уровень на выходе.

И только в том случае, когда на всех входах будут единичные сигналы, выходной сигнал станет единичным.

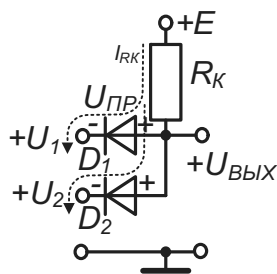


Рис. 1.18

Отметим, что уровень выходного сигнала будет определять наименьший из входных единичных сигналов, диод которого будет проводить, фиксируя $U'_{Вых}$. Другие диоды будут находиться в не-проводящем состоянии.

Таким образом, рассмотренная диодная схема реализует функцию элемента «И». Число входов такого элемента легко увеличить, подключая дополнительные диоды. Условное графическое обозначение (УГО) элемента «И» приведено на рис. 1.19 и выполнено в виде прямоугольника, внутри которого указан знак выполняемой функции «&». Входы элемента принято располагать с левой стороны прямоугольника, а выходы – с правой, что позволяет с первого взгляда идентифицировать тип сигнала и внешнего вывода логического элемента.

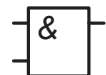


Рис. 1.19

В своё время диодные логические элементы имели достаточно широкое применение. Затем транзисторные логические схемы в интегральном исполнении вытеснили их. Но впоследствии диодные элементы «И» начали применять в БИС программируемых логических матриц (ПЛМ), в которых выполнялось множество наборов диодных логических элементов (см. п. 14.3.2). И пользоваться мог, программируя матрицу, коммутировать сигналы, подаваемые на входы по своему усмотрению.

1.4.3. Диодный логический элемент «ИЛИ» (OR)

выполняет логическое сложение. Выходной сигнал будет нулевым только в одном случае, когда на всех входах будет ноль (см. табл. 2). Сигнал будет единичным, если хотя на одном входе будет единица. Диодная реализация элемента «ИЛИ» (рис. 1.20) не имеет источника питания и соответствующего резистора R_K . Напряжение на выходе образуется за счет энергии входного сигнала.

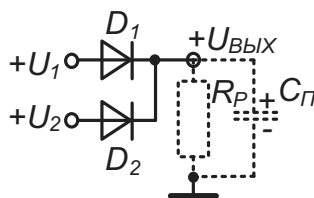


Рис. 1.20

Таблица 2

Входы		Выход
X_1	X_2	Y
0	0	0
0	1	1
1	0	1
1	1	1

Проверим вначале нулевую комбинацию таблицы 2. Когда на всех входах будет ноль, то на выходе неоткуда взяться напряжению, то есть тоже будет нулевой уровень.

Когда на любой из входов будет подан единичный сигнал, то соответствующий диод от-

кроется (ток от входного сигнала идет по «мнемострелке») и на выходе возникнет единичное напряжение, равное входному за вычетом незначительного прямого напряжения $U_{пр}$ на диоде (см. п. 1.4.1).

Примечание: Важно заметить, что остальные диоды будут находиться в непроводящем состоянии под действием обратного для них выходного напряжения.

Если на входах будут действовать несколько единичных сигналов, то проводить будет только **один** диод, с максимальным входным сигналом.

Таким образом, данная схема реализует логическую функцию «ИЛИ». Увеличить число входов можно, добавляя необходимое число диодов.

УГО элемента «ИЛИ», приведенное на рис. 1.21, отличается от ранее рассмотренного только знаком функции («1» вместо «&»).



Рис. 1.21

Резистор R_p , показанный на рис. 1.20 пунктиром, создает путь для разряда паразитной емкости $C_{п}$. Дело в том, что при единице на выходе емкость заряжается (полярность на рисунке). А когда все входные сигналы станут нулевыми, то положительное напряжение на ёмкости исчезнуть мгновенно не может и переедет все диоды в непроводящее состояние (ток от емкости идет против «мнемострелки»). Поэтому, если резистора R_d нет, то не будет и пути для быстрого разряда этой паразитной емкости, которая сможет разряжаться очень малыми обратными токами непроводящих диодов. Величину сопротивления обычно оценивают по формуле (1.7).

1.4.4. «Монтажное ИЛИ»

Простота диодной схемы «ИЛИ» может ввести в заблуждение, поскольку можно подумать, что диоды не выполняют никакой полезной функции и их вообще можно исключить, просто соединив источники сигналов для получения выходного напряжения. Однако из рис. 1.22 нетрудно видеть, что при наличии на входах разных уровней (нуля и единицы) они будут конкурировать между собой (то ли ноль, то ли единица) и возникнет ток короткого замыкания $I_{кз}$, что может привести к отказу устройства.

Диоды в этой ситуации исключают взаимовлияние источников сигнала друг на друга, поскольку может проводить только один диод (см. примечание выше).

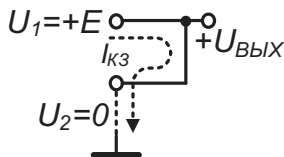


Рис. 1.22

Однако подобное «монтажное ИЛИ» иногда используется на практике при соединении (монтаже) логических элементов с выходным каскадом, допускающим конкуренцию (с открытым выходом или третьим состоянием) (см. п.п. 2.3.6 и 2.3.7).

§1.5. Диодные ограничители уровня сигнала

В некоторых случаях входные сигналы не соответствуют используемому стандарту. Например, они могут иметь отрицательную полярность или напряжение больше, чем допустимо для микросхемы. В этом случае применяют ограничители уровня на основе диодных ключей, которые могут включаться последовательно с выходом или параллельно. Соответственно принято различать эти два типа ограничителей [7]. Ниже рассмотрены наиболее распространенные из них.

1.5.1. Ограничитель отрицательного сигнала

в простейшем случае представляет собой диод, включенный последовательно между входом и выходом (рис. 1.23).

Такая схема, называемая обычно однополупериодным выпрямителем, является, по сути, частью диодной схемы «ИЛИ» (рис. 1.20) и подобно последней также плохо работает на паразитную емкость и требует при необходимости использования дополнительного резистора R_p .

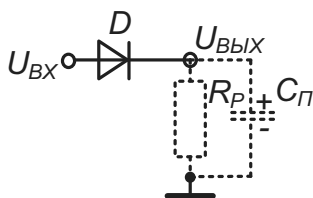


Рис. 1.23

Диод пропускает только положительное напряжение U_{BX} и не пропускает отрицательное (см. временные диаграммы на рис. 1.24). Влияние паразитной емкости на диаграмме не учтено, но отражена неидеальность проводящего диода, что приводит к уменьшению напряжения $U_{ВЫХ}$ на величину $U_{пр}$.

Рассмотренная схема применяется сравнительно редко, так как несколько уменьшает уровень положительного сигнала и имеет невысокое быстродействие.

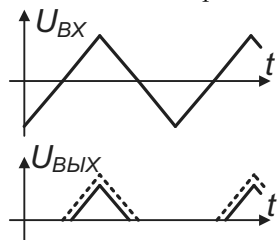


Рис. 1.24

1.5.2. Параллельный ограничитель отрицательной полярности

В этой схеме (рис. 1.25) диод D_1 включен параллельно выходному сигналу и в такой полярности, чтобы не проводить (не «мешать») при положительном сигнале и проводить при отрицательном.

Поэтому для положительной полярности входного сигнала (рис. 1.26) диод будет представляться «разрывом» (ток против «мнемострелки») и выходной сигнал будет практически равен входному (за исключением очень малого падения напряжения на резисторе $R_{огр}$ от протекания незначительного обратного тока диода).

При отрицательной полярности входного сигнала диод будет проводить, поэтому выходное напряжение, хотя и будет отрицательным, но весьма малым и равным напряжению проводящего диода $U_{пр}$, что достаточно для защиты от нестандартного обратного напряжения. Резистор $R_{огр}$ при этом ограничивает ток диода на допустимом уровне (см. п. 1.1.3).

Рассмотренная схема входит в состав входной цепи большинства интегральных логических элементов, выполненных на биполярных (см. п. 2.3.3) или полевых (см. п. 2.7.3) транзисторах.

1.5.3. Ограничитель положительного напряжения сверху

Большинство логических элементов рассчитано на работу со стандартными единичными уровнями напряжения. Чтобы обеспечить защиту от перенапряжения на входе микросхемы обычно используют диодный ограничитель параллельного типа (рис. 1.27), подобный описанному в п. 1.5.2 и имеющий, кроме ограничивающего диода D_2 , источник питания, который задает уровень ограничения. Обычно в качестве такого источника используют напряжение питания устройства или часть его.

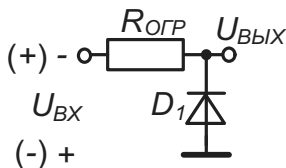


Рис. 1.25

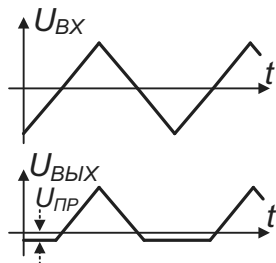


Рис. 1.26

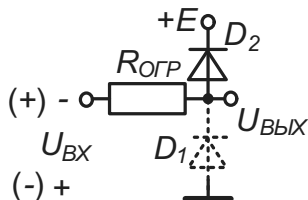


Рис. 1.27

В большинстве случаев такую схему используют для двухстороннего ограничения, поэтому в схеме рис. 1.27 для защиты от отрицательного напряжения добавлен показанный пунктиром диод D_1 , работа которого была пояснена ранее (см. п. 1.5.2).

Пусть входной сигнал имеет отрицательную полярность, при которой диод D_1 проводит, то есть на выходе будет малый потенциал $U_{пр1}$. Тогда на диоде D_2 будет действовать обратное напряжение (ток от напряжения E будет направлен против «мнемострелки») и диод D_2 будет представлять разрыв (см. рис. 1.16в). Таким образом, схема окажется такой же, как в предыдущем случае, и выходной сигнал отрицательной полярности будет ограничен снизу (рис. 1.28) на уровне $U_{пр1}$.

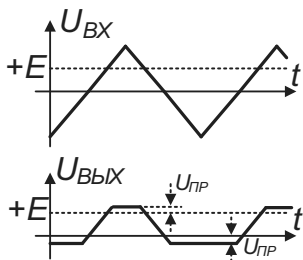


Рис. 1.28

При положительной полярности входного сигнала диод D_1 переходит в непроводящее состояние и влиять на работу схемы не будет. Основную роль при этой полярности будет играть диод D_2 , обратное напряжение на котором определяется разностью $E - U_{ВХ}$ (полярность $U_{ВХ}$ на рис. 1.27 дана в скобках).

Если $U_{ВХ} < E$, то диод D_2 будет оставаться в непроводящем состоянии, а выходной сигнал будет повторять входной. Когда же входной сигнал сравняется с величиной E и превысит ее, то диод D_2 начнет проводить и подключит выходной зажим к источнику E . Поэтому выходной сигнал перестанет увеличиваться и зафиксируется на уровне $E + U_{пр} \approx E$.

Таким образом, рассмотренная схема ограничивает сигнал сверху (а при наличии диода D_1 и снизу), исключая нежелательные уровни.

Изменяя полярности включения источника питания и диода, можно получать различные формы ограничения [7].

1.5.4. Двусторонний ограничитель на стабилитроне

Вольт-амперная характеристика стабилитрона подобна характеристике диода, но имеет на обратной ветви участок восстанавливаемого электрического пробоя с почти неизменным напряжением $U_{СТ}$ зависящим от типа прибора (рис. 1.29)

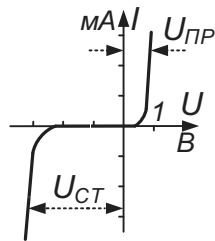


Рис. 1.29

[28]. Выбирая стабилитрон с напряжением стабилизации, равным верхнему порогу ограничения, можно реализовать двухсторонний ограничитель сигнала без источника питания (рис. 1.30).

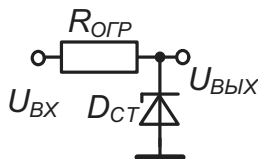


Рис. 1.30

Стабилитрон в этой схеме включен подобно диоду D_1 в п. 1.5.2 и при отрицательной полярности входного сигнала работает на прямой ветви характеристики, как и ранее, ограничивая выходной сигнал на малом уровне $U_{пр}$.

При положительной полярности стабилитрон работает на обратной ветви характеристики и при напряжении меньшем $U_{ст}$ имеет весьма малый обратный ток, не влияющий на передачу сигнала к выходным зажимам. Однако, если входной сигнал сравняется с напряжением пробоя и станет большим, стабилитрон пробьется и выходное напряжение будет ограничено на уровне $U_{ст}$.

Временные диаграммы этой схемы практически не отличаются от приведенных на рис. 1.28, только амплитуда положительного выходного импульса не будет превышать $U_{ст}$.

Стабилитрон, в отличие от диода, имеет существенно большую паразитную емкость между выводами [28], и потому стабилитронный ограничитель рекомендуется использовать на сравнительно низких рабочих частотах.

§1.6. Свойства биполярного транзистора в ключевом режиме

Особенности ключей на полевых транзисторах описаны в §2.6.

1.6.1. Режимы работы биполярного транзистора

Биполярный транзистор является управляемым полупроводниковым прибором, состоящим из двух $p-n$ переходов, но сближенных и разделенных тонким базовым слоем. Каждый из переходов подобно диодам имеет нелинейные свойства, но благодаря близости переходы могут взаимодействовать, например, ток прямосмещенного эмиттерного перехода почти полностью может перехватываться электрическим полем обратносмещенного коллекторного перехода [28].

Для определенности далее будем рассматривать транзисторы $n-p-n$ типа, которые чаще всего используется в ключевых логических элементах.

Биполярный транзистор, УГО которого приведено на рис. 1.31, в зависимости от полярности напряжений на его переходах может работать в четырех режимах [5]:

- в нормальном активном режиме;
- в инверсном активном режиме;
- в режиме насыщения;
- в режиме отсечки тока.

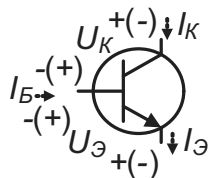


Рис. 1.31

В активном режиме (эмиттерный переход смещен в прямом направлении, а коллекторный – в обратном) направления токов показаны на рис. 1.30, транзистор является управляемым источником тока. Например, в схеме с общим эмиттером (ОЭ) выходным является ток коллектора I_K , который во много раз больше входного, базового тока I_B :

$$I_K \approx B \cdot I_B. \quad (1.12)$$

где B – коэффициент усиления транзистора в схеме с ОЭ.

Активный режим является основой усилительных схем и рассматривается в главе 5. Инверсный активный режим почти не используется на практике, но иногда возникает в ТТЛ элементах (см. п. 2.4.3).

В ключевых схемах обычно используются режимы насыщения и отсечки, которые рассмотрены далее.

В режиме *насыщения* оба перехода транзистора смещены в прямом направлении (полярность на рис. 1.31 дана в скобках), поэтому падения напряжения на переходах (как на прямосмещенных диодах) невелики, соответственно транзистор будет находиться в проводящем состоянии (см. п. 1.6.7).

В режиме *отсечки тока* (непроводящее состояние транзистора) оба перехода транзистора смещены в обратном направлении (полярность на рис. 1.31 дана без скобок), то есть

$$U_E < 0, \quad (1.13)$$

$$U_K < 0. \quad (1.14)$$

Токи переходов при этом близки к нулевым, поэтому выводы транзистора почти не имеют электрической связи друг с другом (см. далее).

В зависимости от величины обратных напряжений на переходах различают несколько разновидностей непроводящего состояния транзистора:

- режим глубокой отсечки с запирающим напряжением (см. п. 1.6.2);
- режим с оборванной базой (см. п. 1.6.3);
- режим с «заземленной» базой, (см. п. 1.6.4).

1.6.2. Токи транзистора в режиме глубокой отсечки

В этом режиме на обоих переходах должны действовать обратные напряжения (полярность приведена на рис. 1.32), величина которых должна быть не менее десятых долей V [28]. Коллекторный переход, как видно из этого рисунка, оказывается смещенным в обратном направлении за счет полярности и величины питающего напряжения E . А вот на эмиттерный переход для обеспечения условий глубокой отсечки приходится подавать дополнительное отрицательное напряжение $-E_{см}$ (см. п. 1.7.1).

При выполнении указанных условий токи транзистора оказываются минимально возможными [28]:

$$I_{\varepsilon} \approx 0, \quad (1.15)$$

$$I_K \approx I_{KO}, \quad (1.16)$$

$$I_B = I_{\varepsilon} - I_K \approx -I_{KO}, \quad (1.17)$$

где I_{KO} — обратный ток коллекторного перехода транзистора. Его величина зависит от типа и мощности транзистора (от единиц μA до десятков mA) и максимальное значение этого тока обычно указано в справочниках. Приведенные значения токов используются в эквивалентной схеме замещения непероводящего транзистора в режиме глубокой отсечки (см. п. 1.6.5).

Знак «минус» в выражении (1.17) означает, что ток I_B имеет обратное по сравнению с нормальным активным режимом направление, показанным на рис. 1.31.

Режим глубокой отсечки обеспечивает надежное непроводящее состояние транзисторного ключа и широко используется на практике, но требует дополнительного источника напряжения другой полярности, который не всегда имеется в устройстве.

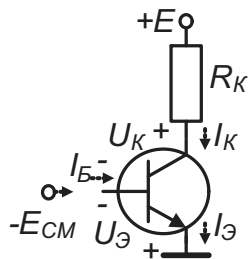


Рис. 1.32

1.6.3. Токи транзистора в режиме с оборванной базой

Стремясь исключить запирающий источник $E_{см}$, можно из приближенного выражения (1.12) сделать неправильный вывод, что при $I_B = 0$ коллекторный ток тоже будет нулевым. На самом деле такой случай является весьма нежелательным, так как происходит увеличение остаточных токов транзистора в B раз (то есть в десятки или даже в сотни раз) [28]:

$$I_{\varepsilon} = I_K \approx I_{KO} (B + 1) \quad (1.18).$$

Поэтому режим с оборванной базой *не должен использоваться* в ключевых схемах. Кроме того в режиме с оборванной базой снижается пробивное напряжение между коллектором и эмиттером.

1.6.4. Токи транзистора в режиме «заземленной» базы

Альтернативой для нежелательного режима с оборванной базой является режим «заземленной» базы (рис. 1.33), когда база через резистор соединяется с эмиттером, подключенным к общей точке («земле»). Токи транзистора в этом случае зависят от величины внешнего сопротивления и минимальны при $R_{см} = 0$ [28]:

$$I_{\varepsilon} \approx I_{KO} B_i \quad (1.19)$$

$$I_K \approx I_{KO} (B_i + 1), \quad (1.20)$$

$$I_B = I_{\varepsilon} - I_K \approx -I_{KO}, \quad (1.21)$$

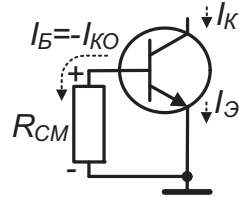


Рис. 1.33

где B_i – коэффициент усиления транзистора в *инверсном* режиме, величина которого много меньше значения B в нормальном усилительном режиме. Поэтому в большинстве случаев считают, что токи непроводящего транзистора увеличиваются незначительно (максимум в несколько раз).

Но нулевое значение $R_{см}$ не позволяет управлять транзистором, поэтому, чтобы исключить шунтирующее влияние $R_{см}$ стараются по возможности увеличить его значение. Препятствует этому падение напряжения на этом сопротивлении от протекания обратного тока непроводящего транзистора $I_B \approx -I_{KO}$, который, как видно из рис. 1.33, создает на сопротивлении падения напряжения положительной полярности, которое может приоткрыть эмиттерный переход транзис-

тора и вызвать появление прямого базового тока и, соответственно, увеличение коллекторного тока.

Чтобы избежать перехода транзистора в активный усилительный режим, используют нелинейность входной характеристики транзистора, которая, как известно, имеет напряжение отпирания. То есть прямой базовый ток появляется только при напряжении на базе большем $U_{отп}$ (рис. 1.34). Поэтому, чтобы не происходило увеличения остаточных токов транзистора, необходимо выполнение неравенства:

$$I_{КО} R_{CM} < U_{отп} \quad (1.22)$$

из которого можно получить расчетное соотношение для ограничения значения R_{CM} сверху:

$$R_{CM} < U_{отп} / I_{КО} \quad (1.23)$$

Расчет проводят обычно на худший случай, подставляя максимальное значение $I_{КО}$. Пусть, например, $I_{КОмакс} = 5$ мкА, а $U_{отп} \approx 0,3$ В, тогда $R_{CMмакс} \approx 0,3 / 5 \cdot 10^{-6} = 600$ кОм. Для большей надежности, естественно, значение R_{CM} выбирают меньше, например, в три раза.

Источника запирающего напряжения в этом случае нет, поэтому подобный режим отсечки тока достаточно часто используют на практике, например, в полупроводниковых логических элементах (см. п. 2.1.3).

1.6.5. Схемы замещения непроводящего транзистора

В соответствии с тремя возможными режимами отсечки (см. п. 1.6.1) можно предложить три схемы замещения непроводящего транзистора в виде источников остаточных токов в соответствии с уравнениями (1.15)–(1.21). Но учитывая, что режим глубокой отсечки является наиболее предпочтительным, ограничимся только одной эквивалентной схемой, именно для этого режима (рис. 1.35).

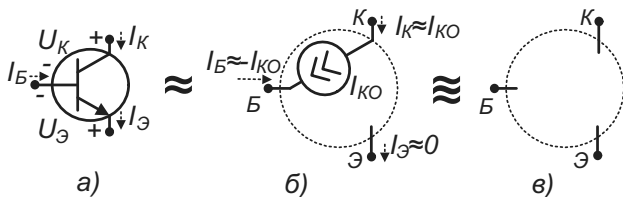


Рис. 1.35

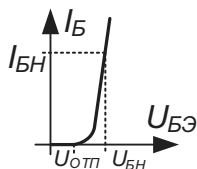


Рис. 1.34

Приведенную на рис. 1.35б схему в виде одного источника тока $I_{\text{ко}}$ будем применять при расчетах транзисторного ключа (см. §1.7). А упрощенную схему (рис. 1.35в), в которой связь между выводами транзистора вообще отсутствует, будем использовать при пояснении принципа действия транзисторных ключей.

1.6.6. Токовое условие насыщения биполярного транзистора

В режиме насыщения оба перехода транзистора должны быть смещены в прямом направлении, то есть

$$U_{\text{э}} > 0, \quad (1.24)$$

$$U_{\text{к}} > 0. \quad (1.25)$$

Проще всего выполнить первое условие (1.24), если правильно выбрать полярность входного сигнала. Второе условие выполнить сложнее, потому что полярность питающего напряжения E задана и приводит, как отмечено в п. 1.6.2, к смещению коллекторного перехода в обратном направлении (см. рис. 1.32).

Для пояснения возможности смещения коллекторного перехода в прямом направлении рассмотрим более подробно работу транзисторного ключа (рис. 1.36), чтобы оценить разность потенциалов $\varphi_{\text{б}} - \varphi_{\text{к}}$, которая определяет искомое напряжение $U_{\text{к}}$. Дело в том, что потенциал коллектора зависит не только от напряжения питания E , но и от тока $I_{\text{к}}$, создающего падение напряжения на резисторе $R_{\text{к}}$:

$$\varphi_{\text{к}} = E - I_{\text{к}} R_{\text{к}} \quad (1.26)$$

Пусть входной сигнал равен нулю. Будем считать, что при этом схема работает в режиме с «заземленной» базой (см. п. 1.6.4). Поэтому транзистор не проводит и его токами для простоты можно пренебречь, то есть потенциал коллектора $\varphi_{\text{к}}$ будет равен напряжению питания E . Отсюда можно видеть, (учитывая $\varphi_{\text{б}} \approx 0$), что напряжение на коллекторном переходе, которое определяется разностью потенциалов ($\varphi_{\text{б}} - \varphi_{\text{к}} \approx -E$) будет, как и прежде, обратным и большим по величине.

Подадим далее входной сигнал положительной полярности, чтобы выполнить первое неравенство (1.24) для режима насыщения. То

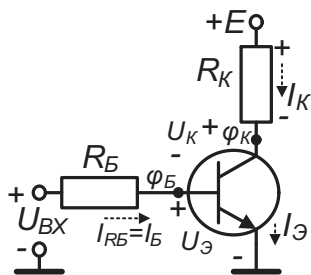


Рис. 1.36

есть эмиттерный переход сместится в прямом направлении и начнет протекать базовый ток I_B .

Но напряжение на коллекторном переходе по-прежнему будет обратным, что приводит к работе транзистора в усилительном режиме ($U_э > 0$, $U_к < 0$, см. п. 1.6.1). При этом в соответствии с (1.12) появится усиленный коллекторный ток $I_к$, что, как видно из (1.26), приведет уменьшению коллекторного потенциала и соответственно уменьшатся значение $\phi_к$ и обратное напряжение на коллекторном переходе.

Увеличивая далее входной сигнал (то есть базовый ток), можно добиться равенства потенциала коллектора и потенциала базы, что будет означать работу транзистора на границе активного режима и режима насыщения. И при определенном значении тока I_B , а значит и тока $I_к$, потенциал коллектора станет меньше, чем потенциал базы, то есть полярность напряжения на коллекторном переходе поменяет знак и станет прямой, что означает выполнение второго условия насыщения (1.25).

Заметим, что произойдет это при определенном коллекторном токе, который в режиме насыщения, когда падением напряжения на транзисторе можно пренебречь, примерно равен $I_{кн} \approx E / R_к$. Тогда, зная коллекторный ток можно найти базовый в режиме насыщения (см. п. 1.6.1), поэтому заменить редко используемое потенциальное неравенство (1.25) более распространенным токовым [28]:

$$I_{Бн} \geq I_{кн} / \beta. \quad (1.27)$$

Это *токовое условие насыщения* более удобно на практике, так как токи транзистора достаточно просто рассчитать, поскольку они зависят от внешних напряжений и сопротивлений (см. §1.7).

1.6.7. Схема замещения проводящего транзистора

В режиме насыщения через оба перехода транзистора идут прямые токи, поэтому падения напряжения на переходах невелики. Поэтому при пояснениях принципа действия ключевых устройств проводящий транзистор упрощенно считают «стянутым в точку» (рис. 1.37в). На практике же для расчетов нужны более точные параметры для замещения.

Каждый прямосмещенный переход транзистора, как и полупроводниковый диод, можно заменить его эквивалентной схемой в виде источника напряжения с величиной менее 1 В (см. п. 1.3.2).

Поэтому в эквивалентной схеме проводящего транзистора с ОЭ свойства эмиттерного перехода отражает источник напряжения $U_{БН}$ (рис. 1.37б). Кроме того свойства коллекторного перехода учтены в источнике напряжением $U_{КН}$, которое равно разности напряжений на переходах U_K и $U_Э$ и может составлять доли В.

Значения $U_{БН}$ ($\approx 0,7..1,0$ В) и $U_{КН}$ ($\approx 0,1..1,0$ В) зависят от токов транзистора и его типа. При использовании справочных данных, нужно помнить, что производители приводят максимально возможные значения, и делать поправку, учитывая реальные величины токов.

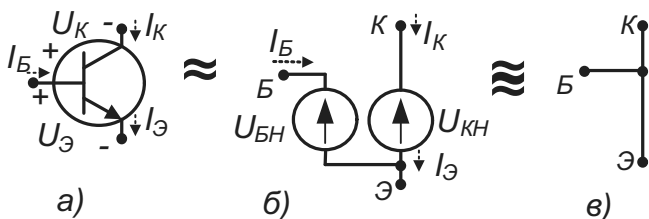


Рис. 1.37

§1.7. Ключ на биполярном транзисторе

1.7.1. Схема и работа ключа

Кроме резистора нагрузки R_K и источника питания $+E$, назначение которых рассмотрено в п. 1.1.1, для управления работой транзистора, выполняющего функции ключа, в схеме рис. 1.38а, введены дополнительные элементы в базовой цепи.

Резистор $R_{см}$ и источник отрицательного (запирающего) напряжения $E_{см}$ предназначены для реализации режима глубокой отсечки, в котором минимальны токи транзистора (см. п. 1.6.2).

Сопротивление $R_Б$ обеспечивает режим насыщения транзистора, то есть ограничивает и задаёт необходимый базовый ток (см. п. 1.6.6).

Транзисторный ключ должен работать в двух состояниях, проводящем и непроводящем, в зависимости от уровня входного сигнала.

Пусть на вход подан нулевой сигнал, то есть можно считать, что в базовой цепи будет действовать только отрицательный источник напряжения. Поэтому на базе (эмиттерном переходе) будет запирающее напряжение. На коллекторном переходе, как уже отмечалось

ранее (см. п. 1.6.2), также действует запирающее напряжение питания E , то есть транзистор будет работать в режиме глубокой отсечки тока. Это означает, что непроводящий транзистор не влияет на работу схемы и напряжение питания E будет создавать на выходе высокий единичный уровень. Таким образом, при нулевом сигнале на входе, на выходе будет единица.

Когда же на вход подано высокое единичное напряжение положительной полярности для прямого смещения эмиттерного перехода (см. п. 1.6.6), это вызовет протекание тока $I_{RБ}$ через резистор R_B , большая часть этого тока будет затекать в базу. При правильном расчете сопротивления R_B (см. п. 1.7.3) транзистор будет насыщен и «стянут в точку», поэтому при единице на входе схемы на выходе образуется нулевой сигнал.

Из этих рассуждений следует общий вывод – транзисторный ключ выполняет функцию логического инвертора, УГО которого приведено на рис. 1.38б.

Для обеспечения надежной работы транзисторного ключа необходимо, чтобы при низком уровне входного сигнала ($U_{вх} = U^0$) транзистор находился в режиме отсечки, а при высоком уровне ($U_{вх} = U^1$) – в режиме насыщения. Это будет возможно при правильном расчете значений сопротивлений схемы.

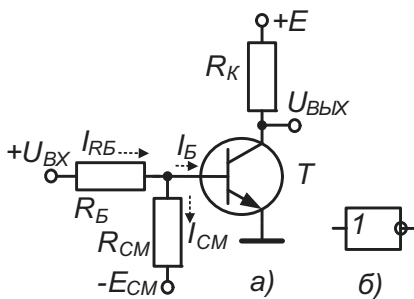


Рис. 1.38

1.7.2. Условие работоспособности ключа в режиме отсечки

В этом случае на входе действует напряжение U^0 . Транзистор должен находиться в непроводящем состоянии (в режиме глубокой отсечки). Поэтому заменим нелинейный транзистор его линейной эквивалентной схемой для указанного режима в виде источника остаточного тока $I_{к0}$ (рис. 1.35б) и получим расчетную схему, приведенную на рис. 1.39.

В режиме глубокой отсечки оба перехода транзистора должны быть смещены в обратном направлении (полярность показана на рисунке). Причем коллекторный переход смещается в обратном направлении «автоматически», благодаря стандартной полярности напряжения питания E .

Полярность напряжения на эмиттерном переходе $U_{БЭ}$ зависит, как видно из рис. 1.39, от взаимодействия трех разнополярных источников энергии, входного сигнала U^0 , напряжения смещения E_{CM} и источника обратного тока $I_{КО}$, и определяется, в конечном счете, соотношением сопротивлений в базовой цепи.

Для нахождения величины и полярности напряжения $U_{БЭ}$ будем считать, что в нашей линейной схеме источники энергии действуют независимо и поочередно, тогда согласно принципу наложения-суперпозиции [20] можно записать $U_{БЭ}$ в виде трех составляющих

$$U_{БЭ} = U_1 + U_2 + U_3, \quad (1.28)$$

где

$$U_1 = +U^0 R_{CM} / (R_B + R_{CM}), \quad (1.29)$$

$$U_2 = -E_{CM} R_B / (R_B + R_{CM}), \quad (1.30)$$

$$U_3 = +I_{КО} R_B R_{CM} / (R_B + R_{CM}). \quad (1.31)$$

Выражение (1.28) для реализации режима глубокой отсечки должно быть отрицательным, поэтому, подставляя значения составляющих из (1.29)–(1.31), можно получить условие надежного запираания транзистора:

$$U^0 R_{CM} - E_{CM} R_B + I_{КО} R_B R_{CM} < 0. \quad (1.32)$$

Обычно это неравенство используется для ограничения величины R_{CM} сверху:

$$R_{CM} < E_{CM} / (U^0 / R_B + I_{КО}). \quad (1.33)$$

Но для вычисления конкретного значения R_{CM} из (1.33) необходимо знать величину R_B , которое определяется из условия насыщения (см. далее).

Схему рис. 1.39 можно использовать также для уточнения величины единичного выходного напряжения $U_{ВЫХ}^1$, что рекомендует-ся сделать самостоятельно.

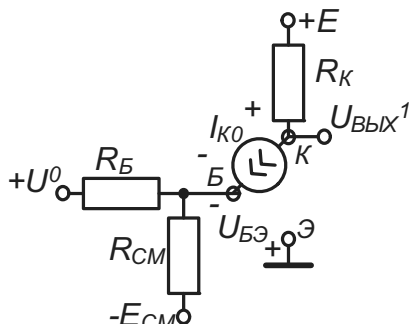


Рис. 1.39

1.7.3. Условие работоспособности ключа в режиме насыщения

Будем считать, что в этом случае на входе действует единичный сигнал U^1 , транзистор насыщен и вместо него в расчетной схеме (рис. 1.40) будем использовать эквивалентную схему замещения в виде двух источников напряжения малой величины $U_{БН}$ и $U_{КН}$.

Основным уравнением для проводящего состояния транзисторного ключа является токовое условие насыщения (1.27). Чтобы определить при каких значениях сопротивления R_B оно выполняется, следует найти токи I_K и I_B в режиме насыщения из схемы рис. 1.40.

Проще всего определить, коллекторный ток, который равен току через резистор R_K , используя закон Ома и учитывая, что $E \gg U_{КН}$:

$$I_{КН} = I_{RK} = U_{RK} / R_K = (E - U_{КН}) / R_K \approx E / R_K \quad (1.34)$$

Базовый ток является частью тока I_{RB} , поэтому его можно найти, как разность токов:

$$I_B = I_{RB} - I_{CM} \quad (1.35)$$

где

$$I_{RB} = (U^1 - U_{БН}) / R_B \approx U^1 / R_B \quad (1.36)$$

$$I_{CM} = (E_{CM} + U_{БН}) / R_{CM} \approx E_{CM} / R_{CM} \quad (1.37)$$

Поскольку на практике напряжения на переходах насыщенного транзистора много меньше напряжения внешних источников, в большинстве случаев пользуются упрощенными выражениями для токов. Тогда, подставляя их в условие насыщения (1.27), можно получить

$$U^1 / R_B - E_{CM} / R_{CM} \geq E / R_K B. \quad (1.38)$$

Это условие насыщения для конкретной схемы транзисторного ключа (рис. 1.38), выраженное через внешние напряжения и сопротивления, обычно используют для нахождения сопротивления R_B :

$$R_B \leq U^1 / (E / R_K B_{МИН} + E_{CM} / R_{CM}). \quad (1.39)$$

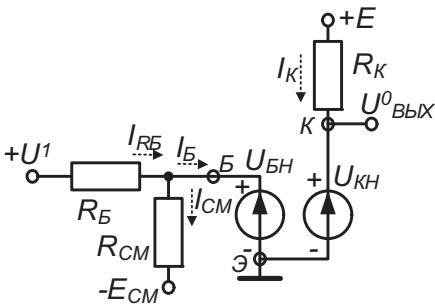


Рис. 1.40

Полученные неравенства (1.33) и (1.39) имеют два неизвестных, поэтому их решают совместно и находят значения $R_{см}$ и $R_б$, при которых будет обеспечиваться надежное запираение и насыщение транзистора в ключевой схеме.

1.7.4. Упрощенный порядок расчета ключа

Напряжения E , $E_{см}$ и параметры входного сигнала U^0 , U^1 обычно бывают заданы. Величина сопротивления нагрузки ключа R_k оценивается, исходя из требований к выходной цепи (см. п. 1.1.4). Транзистор выбирают по напряжению, току и частоте. Неравенства (1.33) и (1.39) для оценки сопротивлений $R_{см}$ и $R_б$ имеют достаточно большой диапазон допустимых значений, поэтому на практике стараются использовать их в упрощенном виде, выделяя, например, два случая.

В первом случае в неравенстве (1.33), считая $U^0 \approx 0$, пренебрегают первым слагаемым в круглых скобках, получают простое неравенство для оценки в виде $R_{см} < E_{см} / I_{ко}$ и далее при выбранном $R_{см}$ находят по формуле (1.39) величину $R_б$.

Во втором случае в неравенстве (1.33) пренебрегают вторым слагаемым в круглых скобках, считая, что $I_{ко} = 0$. После этого упрощенное соотношение, например, в виде $R_{см} \approx 0,5R_бE_{см} / U^0$, подставляют в условие насыщения (1.39) и решают его, находя $R_б$.

Расчет обычно проводят на худший случай, поэтому в выражениях используют максимальный ток $I_{ко}$ и минимальный коэффициент усиления B .

Важно отметить, что наиболее универсальными условиями работоспособности транзисторного ключа являются неравенства (1.13)–(1.14) и (1.24)–(1.25), которые применимы вне зависимости от конфигурации схемы.

§1.8. Переходный процесс при включении биполярного транзистора

1.8.1. Метод заряда

Транзистор является инерционным устройством, что обусловлено в основном происходящими внутри транзистора (и не мгновенными) процессами накопления и рассасывания носителей в области базы и в области коллектора [28].

Влияние заряда в области коллектора на переходный процесс обычно отражают наличием в схеме усредненной по диапазону напряжений коллекторной емкости. Заряд и разряд этой емкости происходит через внешний резистор, поэтому оценку доли инерционности, вносимой зарядом в области коллектора приходится проводить на заключительном этапе расчетов (см. п. 1.9.5).

Выходной коллекторный ток транзистора в усилительном режиме определяется количеством свободных носителей в области базы (то есть зарядом Q_B), что позволяет связать изменения тока I_K с законом изменения заряда Q_B :

$$I_K(t) = Q_B(t) / \tau, \quad (1.40)$$

где τ – время жизни неосновных носителей в базе, которое определяют на основе приводимой в справочниках предельной рабочей частоты $f_{пр}$ транзистора в схеме с ОБ [28]:

$$\tau = B / 2\pi f_{пр} \quad (1.41)$$

Уравнение (1.40) обычно называют уравнением связи метода заряда и важно отметить, что справедливо оно только в активном режиме работы транзистора и на границе насыщения.

Второе уравнение метода заряда отражает динамическую зависимость заряда Q_B от базового тока и записывается в виде дифференциального уравнения, приведенного ниже в упрощенном виде:

$$dQ_B / dt + Q_B / \tau = I_B(t). \quad (1.42)$$

Практическое использование метода заряда подразумевает нахождение из дифференциального уравнения (1.42) закона изменения $Q_B(t)$ при известном токе $I_B(t)$ и расчет выходного тока $I_K(t)$ при помощи уравнения связи (1.40).

Метод заряда позволяет не только достаточно наглядно пояснить процессы включения и выключения транзистора, но и оценить необходимые временные параметры.

1.8.2. Процесс включения транзистора

При подаче на вход ключа (рис. 1.41) единичного импульса эмиттерный переход транзистора сместится в прямом направлении и появится базовый ток:

$$I_B = (U^i - U_{БН}) / R_B \approx U^i / R_B = I_B^1. \quad (1.43)$$

То есть правая часть дифференциального уравнения (1.42) будет постоянной и равной I_B^1 . Поэтому решение уравнения в данном случае будет представлять экспоненту:

$$Q_B(t) = Q_B(\infty) - [Q_B(\infty) - Q_B(0)] \exp(-t/\tau), \quad (1.44)$$

где $Q_B(0)$ – начальный заряд, который в режиме отсечки был равен нулю.

$Q_B(\infty)$ – установившееся (конечное) значение заряда в базе.

Таким образом, в базе начнет накапливаться заряд по экспоненте с постоянной времени τ , стремясь достигнуть значения $Q_B(\infty)$. И в соответствии с (1.40) появится пропорциональный коллекторный ток I_K . Временные диаграммы процесса включения транзистора показаны на рис. 1.42.

В активном режиме ток I_K пропорционален Q_B , поэтому (после небольшой задержки t_3 , связанной с конечным временем диффузии дырок через область базы), ток I_K также растет по экспоненте. На границе активной области и области насыщения ток I_K перестает изменяться, поскольку в режиме насыщения он согласно уравнению (1.34) ограничен величиной сопротивления

R_K . В этот момент, когда в базе накоплен заряд $Q_{ГР}$, достаточный для насыщения транзистора, заканчивается время включения транзистора $t_{ВКЛ}$. Однако, изменение заряда Q_B еще продолжается, происходит накопление избыточного (ненужного для насыщения) заряда $Q_{ИЗБ}$, который при выключении придется удалять (см. п. 1.9.1).

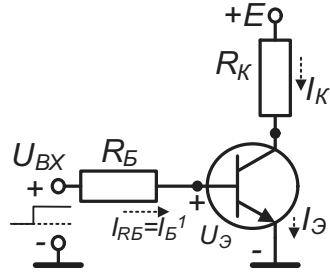


Рис. 1.41

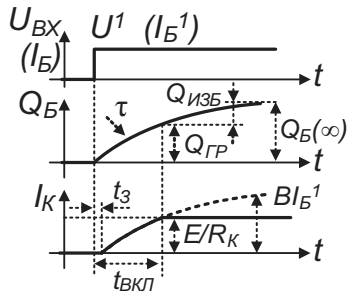


Рис. 1.42

1.8.3. Нахождение времени включения

Для оценки времени включения воспользуемся решением дифференциального уравнения (1.42) в виде (1.44), подставляя начальное условие $Q_B(0) = 0$.

Значение $Q_B(\infty)$ для случая $I_B(t) = I_B^{-1}$ найдем из следующих рассуждений. В установившемся режиме изменений заряда нет, поэтому из уравнения (1.42), считая производную $dQ_B / dt = 0$, можно получить

$$Q_B / \tau = I_B^{-1} \quad (1.45)$$

и найти затем значение

$$Q_B = I_B^{-1} \tau = Q_B(\infty). \quad (1.46)$$

С учетом изложенного закон изменения заряда в базе (1.44) можно представить в следующем виде:

$$Q_B(t) = I_B^{-1} \tau - I_B^{-1} \tau \exp(-t/\tau) \quad (1.47)$$

и использовать его для нахождения временной зависимости выходного тока $I_K(t)$ при помощи уравнения связи (1.40).

Но подобные формулы нужны в основном в моделирующих программах, на практике обычно ограничиваются оценкой времени включения. Поэтому подставим вместо текущего времени t в (1.47) время $t_{\text{вкл}}$, вместо $Q_B(t) = Q_B(t_{\text{вкл}}) = Q_{\text{ГР}} = I_{\text{КН}} \tau / B$ и получим уравнение с одним неизвестным $t_{\text{вкл}}$:

$$I_{\text{КН}} \tau / B = I_B^{-1} \tau - I_B^{-1} \tau \exp(-t/\tau). \quad (1.48)$$

Значение $Q_{\text{ГР}}$ можно найти из уравнения связи (1.40), которое на границе активного режима и режима насыщения все еще остается справедливым, а ток $I_{\text{КН}}$ на границе насыщения согласно (1.34) ограничен внешним резистором и его числовое значение легко рассчитать.

Решая уравнение (1.48), можно получить формулу для оценки времени включения:

$$t_{\text{вкл}} = \tau \ln(I_B^{-1} B / (I_B^{-1} B - I_{\text{КН}})). \quad (1.49)$$

Из формулы видно, что время включения $t_{\text{вкл}}$ зависит не только от частотных свойств транзистора, определяемых значением τ , но и нелинейно – от величины входного сигнала I_B^{-1} . Поэтому для упрощения расчетов обычно выделяют в зависимости от соотношения $I_B^{-1} B$ и $I_{\text{КН}}$ два крайних режима, *сильного* и *слабого* отпирающих сигналов.

1.8.4. Режим сильного отпирающего сигнала

характеризуется почти линейным изменением заряда на участке включения, что возможно, если на входе задан большой базовый ток I_B^{-1} , превышающий необходимый для насыщения по формуле (1.27). В этом случае можно считать $I_B^{-1} B \gg I_{\text{КН}}$ и логарифм в формуле (1.49) без особой погрешности заменить линейным членом разложения в степенной ряд:

$$t_{\text{вкл}} \approx \tau I_{\text{КН}} / I_B^{-1} B. \quad (1.50)$$

Из этого упрощенного выражения для сильного сигнала отчетливо видна обратная зависимость времени $t_{\text{вкл}}$ от величины отпи-

рающего тока I_B^1 . Причем в большинстве случаев $t_{\text{вкл}}$ представляет долю значения τ , то есть режим сильного сигнала достаточно хорошо реализует потенциальные возможности транзистора.

Однако следует иметь в виду, что чрезмерное увеличение единичного тока базы приводит к увеличению избыточный заряда $Q_{\text{изб}}$, что создает проблемы при выключении транзистора (см. п. 1.9.1).

1.8.5. Режим слабого отпирающего сигнала

получается в случае, когда базовый ток обеспечивает условие насыщения (1.27) без запаса, то есть $I_B^1 B \approx I_{\text{кн}}$, то есть в формуле (1.49) значения в разности, стоящей в знаменателе дроби под знаком логарифма, оказываются близкими. При этом $Q_{\text{гр}}$ практически совпадает с $Q_{\text{б}}(\infty)$ и процесс включения существенно затягивается. Точное определение времени $t_{\text{вкл}}$ в области медленного завершения экспоненты с учетом разброса параметров схемы затруднено, поэтому на практике пользуются приближенным соотношением:

$$t_{\text{вкл}} \approx 2\tau, \quad (1.51)$$

то есть величина $t_{\text{вкл}}$ в этом случае значительно больше, чем в режиме сильного сигнала. Поэтому режим слабого отпирающего сигнала не рекомендуется использовать на практике.

§1.9. Переходный процесс при выключении биполярного транзистора

1.9.1. Временные диаграммы

Для иллюстрации процесса выключения воспользуемся схемой ключа (рис. 1.43), в которой входной сигнал из единичного стал нулевым и запирающим (чтобы учесть действие цепочки $E_{\text{см}}$ и $R_{\text{см}}$). Смена полярности входного сигнала приведет к смене направления базового тока, который обозначим, как I_B^0 .

Отметим, что этот ток будет течь, пока в базе есть носители заряда, и транзистор не перейдет в режим отсечки тока.

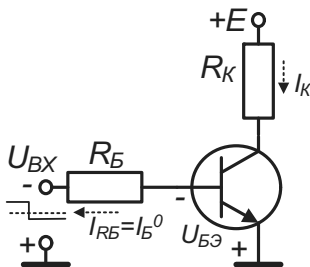


Рис. 1.43

Ток I_B^0 обратного направления будет уносить заряды из базы в соответствии с (1.42). Поэтому заряд Q_B будет уменьшаться по экспоненциальному закону (см. временные диаграммы на рис. 1.44). Но коллекторный ток транзистора, который ранее был насыщен, не будет изменяться и останется равным $I_{KH} \approx E / R_K$ пока из базы не будет удалён избыточный заряд $Q_{изб}$ (этап рассасывания t_p).

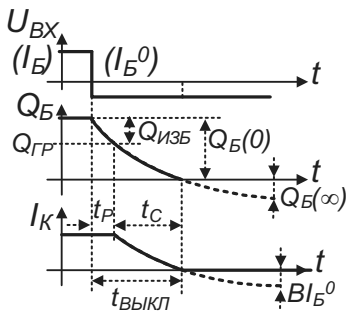


Рис. 1.44

Когда заряд станет меньше критического $Q_{гр}$, транзистор перейдет в активный режим работы и начнет действовать формула (1.40), то есть изменения коллекторного тока будут определяться изменениями заряда в базе, и ток по экспоненте начнет уменьшаться.

Процесс выключения закончится, когда заряд в базе станет равным нулю, хотя заряд в базе стремится к некоторому отрицательному значению. Но отрицательный заряд неосновных носителей в базе не имеет физического смысла, поэтому далее изменения Q_B показаны пунктиром. Ток базы после выключения тоже становится равным нулю (см. пунктир на диаграмме рис. 1.44).

Таким образом, процесс выключения транзистора характеризуется двумя этапами – этапом рассасывания избыточного заряда и этапом спада коллекторного тока, длительности которых t_p и t_c определяются быстротой удаления заряда Q_B из области базы.

1.9.2. Нахождение времени выключения

При выключении транзистора также можно пользоваться решением дифуравнения в виде (1.44), только подставляя другие граничные условия. Например, нетрудно видеть, что начальное значение $Q_B(0)$ в данном случае определяется прежним установившимся значением режима включения, которое найдено ранее в (1.46), то есть $Q_B(0) = I_B^1 \tau$. Аналогично выводу формулы (1.46), только учитывая другое значение базового тока, можно получить для установившегося режима выключения $Q_B(\infty) = -I_B^0 \tau$.

Подставляя полученные значения $Q_B(0)$ и $Q_B(\infty)$ в (1.44), запишем закон изменения заряда при выключении транзистора в следующем виде:

$$Q_B(t) = -I_B^0 \tau - [I_B^0 \tau - I_B^1 \tau] \exp(-t / \tau). \quad (1.52)$$

Далее, как и прежде, не останавливаясь на получении закона изменения коллекторного тока со временем, найдем время выключения транзистора, подставляя в (1.52) $t = t_{\text{выкл}}$ и, используя условие $Q_B(t) = Q_B(t_{\text{выкл}}) = 0$, получим уравнение:

$$0 = -I_B^0 \tau - [I_B^0 \tau - I_B^1 \tau] \exp(-t_{\text{выкл}} / \tau), \quad (1.53)$$

решая которое, можно найти длительность процесса выключения транзистора:

$$t_{\text{выкл}} = \tau \ln(I_B^0 / (I_B^0 - I_B^1)). \quad (1.54)$$

Отсюда видно, что время выключения зависит не только от частотных свойств транзистора, но и (нелинейно) от соотношения отпирающего и запирающего сигналов. В режиме выключения также принято выделять для простоты расчетов режимы сильного и слабого запирающего сигнала (см. далее).

Аналогично, используя условие $Q_B(t) = Q_B(t_p) = Q_{гр} = I_{кн} \tau / B$, можно при необходимости найти время рассасывания t_p а также (с другими граничными условиями) и время спада t_c .

1.9.3. Режим сильного запирающего сигнала

возникает, если выключающий ток I_B^0 сравним с отпирающим током I_B^1 . В этом случае на участке выключения заряд будет изменяться практически линейно, поэтому логарифм в (1.54) можно разложить в степенной ряд, оставив, не внося большой погрешности в расчет, только линейную составляющую:

$$t_{\text{выкл}} = \tau I_B^1 / I_B^0. \quad (1.55)$$

Отсюда видно, что для ускорения выключения желательно увеличивать ток I_B^0 . Кроме того, эта формула, выявляет противоречие, возникающее при выборе параметров отпирающего сигнала. С одной стороны для ускорения процесса включения в соответствии с (1.50) нужно увеличивать отпирающий ток I_B^1 . А с другой, согласно (1.55), увеличение единичного тока приводит к затягиванию процесса выключения транзистора (за счет накапливания избыточного заряда). Простейшие способы разрешения этого противоречия описаны в §1.10.

1.9.4. Режим слабого запирающего сигнала

соответствует практически нулевому запирающему току I_B^0 , когда заряд из базы не удаляется обратным базовым током, а сравнитель-

но медленно исчезает в основном за счет рекомбинации носителей в области базы [28]. Поэтому процесс выключения в этом режиме существенно затягивается и оценку времени выключения проводят по очень приближенной формуле:

$$t_{\text{выкл}} \approx 2 \tau. \quad (1.56)$$

Поскольку транзистор в режиме слабого запирающего сигнала имеет очень плохое быстродействие, этот режим стараются не применять на практике.

1.9.5. Учет заряда в области коллектора

Все полученные уравнения справедливы для транзисторов с диффузионным механизмом переноса носителей. Для высокочастотных транзисторов со встроенным электрическим полем в базе большую роль, кроме заряда в базе, играет заряд в области коллектора, который также необходимо создавать или удалять.

Не останавливаясь на подробностях, отметим, что ранее полученные формулы для расчета $t_{\text{вкл}}$ и $t_{\text{выкл}}$ можно использовать для дрейфовых высокочастотных транзисторов, если учесть наличие заряда в области коллектора путем формального увеличения постоянной времени [28]:

$$\tau^* \approx \tau + BC_K R_K = B / 2\pi f_{\text{пр}} + BC_K R_K \quad (1.57)$$

где C_K – усредненное значение коллекторной емкости транзистора. То есть в расчетных формулах для высокочастотных транзисторов вместо значения τ подставляется увеличенное значение τ^* .

Для высокочастотных транзисторов первое слагаемое оказывается очень малым, а второе – зависящим от сопротивления внешнего резистора R_K . Поэтому в этом случае не следует брать значение R_K чрезмерно большим (не более 5..10 кОм), чтобы не проиграть в быстродействии.

§1.10. Повышение быстродействия ключа на биполярном транзисторе

1.10.1. Оптимальная форма базового тока

Чтобы быстрее включить транзистор, нужно задать большой отпирающий ток I_B^1 (см. п. 1.8.4). Но такой ток, превышающий необходимое для насыщения транзистора значение по формуле (1.27), создаёт

избыточный заряд, удаление которого приводит к увеличению времени выключения. Поэтому, чтобы разрешить это противоречие при выборе базового тока, было предложено изменять величину тока на разных этапах переходного процесса в транзисторе.

Так при включении, базовый ток можно сделать максимально возможным, чтобы форсировать включение транзистора. Но после перехода транзистора в режим насыщения, чтобы не происходило накопления избыточного заряда, значение тока I_B нужно уменьшить до величины примерно равной $I_{KH}B$, чтобы удерживать транзистор в режиме насыщения (рис. 1.45).

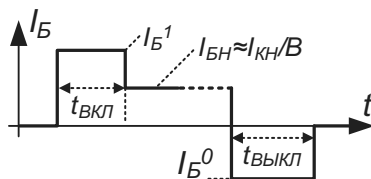


Рис. 1.45

Аналогично поступают и при выключении, снижая ток I_B до нуля после удаления заряда. Впрочем, это происходит само собой при полном удалении заряда из базы и обратном смещении эмиттерного перехода.

Существует два основных способа реализации такой формы базового тока, которую обычно называют оптимальной.

1.10.2. Ключ с ускоряющим конденсатором

Прообразом данного ключа послужила дифференцирующая RC -цепочка, выходной сигнал которой (рис. 1.11) отдаленно напоминает оптимальную форму базового тока. Схема ключа, приведенная на рис. 1.46, отличается от традиционной (см. п. 1.7.1) только наличием ускоряющего конденсатора C_y , который в моменты перепадов входного сигнала обеспечивает временное увеличение отпирающего и запирающего токов базы.

Пусть входное напряжение равно нулю. Поскольку запирающее напряжение на базе невелико, то для простоты можно считать, что на обкладках конденсатора C_y напряжение тоже равно нулю.

Подает на вход скачок единичного сигнала. Напряжение на емкости мгновенно измениться не может, и в первый момент емкость представ-

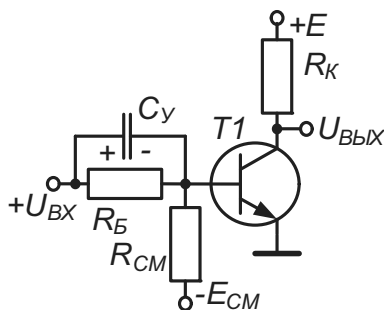


Рис. 1.46

ляет «закоротку» и шунтирует $R_{б'}$. Поэтому в базу идет большой ёмкостной ток, который ограничивается не показанными на схеме и весьма малыми выходным сопротивлением источника сигнала $r_{и}$ и нелинейным входным сопротивлением транзистора $r_{бэ}$:

$$I_{б} \approx U^1 / (r_{и} + r_{бэ}) \gg U^1 / R_{б'}. \quad (1.58)$$

Это большое значение отпирающего базового тока обеспечит *форсированное включение* транзистора, то есть малое значение $t_{вкл}$.

По мере заряда конденсатора C_y ёмкостной ток будет уменьшаться до нуля и ток базы будет определяться (без учета тока смещения) в основном величиной $R_{б'}$, то есть по формуле (1.43). Поскольку это значение базового тока можно выбрать минимальным (только для обеспечения режима насыщения), то большого избыточного заряда в базе при этом не будет.

Отметим также, что конденсатор окажется заряженным до значения единичного входного сигнала, то есть $U_{cy} \approx U^1$. Полярность указана на рис. 1.46.

Пусть далее входное напряжение снова становится нулевым, при этом левая (положительная) обкладка конденсатора окажется подключенной к общей точке, а правая (отрицательная) – к базе. Таким образом между базой и эмиттером транзистора будет действовать источник запирающего напряжения с начальной амплитудой $U_{cy} = U^1$, и потечет большой *обратный* базовый ток, величина которого будет подобно (1.56) ограничиваться только малыми значениями $r_{и}$ и $r_{бэ}$. Благодаря этому будет обеспечиваться быстрое удаление заряда из базы и малое время $t_{выкл}$.

Поводя итог, можно отметить, что наличие ускоряющего конденсатора C_y резко ускоряет процессы включения и выключения транзистора за счет того, что форма базового тока (рис. 1.47) в некоторой степени подобна оптимальной.

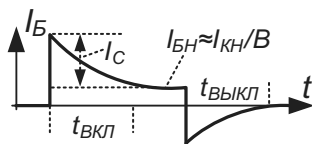


Рис. 1.47

1.10.3. Выбор емкости ускоряющего конденсатора

При выборе величины емкости C_y в принципе можно воспользоваться дифуравнением (1.42) метода заряда с учетом непостоянства базового тока (рис. 1.47). Но для упрощения расчетов обращают внимание на схожесть процессов накопления-рассасывания зарядов в базе транзистора и обкладках конденсатора. Это позволяет про-

известнн выбор величины C_y с использованием правила: *изменение заряда на конденсаторе примерно равно изменению заряда в базе.*

$$\Delta Q_c \approx \Delta Q_b. \quad (1.59)$$

Как отмечалось ранее (см. п. 1.8.3) при включении необходимо накопить граничный заряд

$$Q_{гр} = I_{кн} \tau / B, \quad (1.60)$$

а при выключении нужно удалить несколько больший (на величину $Q_{изб}$) заряд. Поэтому для расчета возьмем значение $Q_b(\infty)$ из (1.46), считая

$$\Delta Q_b = Q_b(\infty) = I_b^{-1} \tau. \quad (1.61)$$

Для оценки изменения заряда на конденсаторе ΔQ_c воспользуемся известным из электротехники определением емкости, как отношения приращения заряда к приращению напряжения:

$$C_y = \Delta Q_c / \Delta U_c, \quad (1.62)$$

поэтому

$$\Delta Q_c = C_y \Delta U_c. \quad (1.63)$$

Вспомним далее (см. п. 1.10.2), что ускоряющий конденсатор заряжается от нуля до U' , то есть $\Delta U_c \approx U'$, что позволяет определить требуемое изменение заряда конденсатора:

$$\Delta Q_c \approx C_y U'. \quad (1.64)$$

Подставляя рассчитанные изменения зарядов ΔQ_c и ΔQ_b в (1.59) и решая получившееся уравнение, можно получить:

$$C_y \approx I_b^{-1} \tau / U'. \quad (1.65)$$

Далее, подставляя из (1.43) приближенное значение I_b^{-1} , получаем окончательную расчетную формулу:

$$C_y \approx U' \tau / R_b U' = \tau / R_b = B / R_b 2\pi f_{пр} \quad (1.66)$$

Из формулы видно, что величина C_y оказывается зависящей от свойств транзистора (предельной частоты $f_{пр}$ и коэффициента усиления B), то есть при максимальном быстродействии для каждого транзистора желательно подбирать свое оптимальное значение C_y . Это исключает применение схемы с ускоряющим конденсатором при изготовлении логических элементов по твердотельной технологии в виде микросхем. Однако в ключах на дискретных транзисторах, работающих на средних и низких частотах, этот недостаток не является существенным.

1.10.4. Экспериментальная оценка правильности выбора C_y

Проверить оптимальность выбора емкости конденсатора можно при помощи осциллографа, наблюдая форму напряжения $U_{БЭ}$ при запираании транзистора. Возможны три основных варианта временных диаграмм $U_{БЭ}$ (рис. 1.48).

В идеальном случае, когда весь заряд емкости будет потрачен на удаление заряда из базы, форма $U_{БЭ}$ будет прямоугольной, и можно считать, что емкость выбрана правильно, то есть $C_y \approx C_{опт}$.

Если отрицательный фронт затянут (заряда емкости не хватило и заряд в базе медленно удаляется за счет обратного направления базового тока), то в этом случае $C_y < C_{опт}$.

Если на базе наблюдается отрицательный выброс напряжения, это означает, что заряд из базы удалился полностью, но на конденсаторе еще остался заряд и $C_y > C_{опт}$.

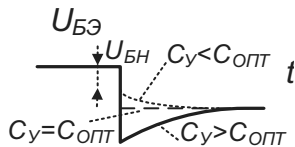


Рис. 1.48

1.10.5. Ключ с нелинейной обратной связью (ОС)

реализует другой способ создания оптимальной формы базового тока за счет введения отрицательной ОС при помощи диода $D_{ш}$ между коллектором и базой транзистора (рис. 1.49). Цепь смещения на схеме для простоты не показана.

Диод, являясь пороговым устройством, контролирует напряжение на коллекторном переходе транзистора и при прямом смещении перехода (после того, как транзистор включился), открывается и уменьшает базовый ток, за счет отвода тока $I_{ос}$.

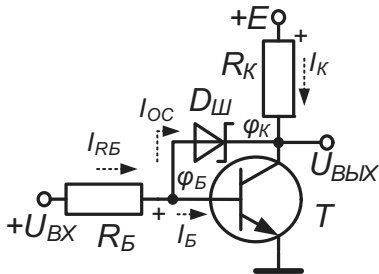


Рис. 1.49

Приятно отметить, что впервые этот принцип предложил советский инженер-физик Б. Н. Кононов в 1955 году [11].

Указанный диод должен быть быстродействующим и иметь близкое к нулю прямое падение напряжения. Такими свойствами в настоящее время обладает диод Шоттки.

Диод Шоттки образуется при определенных условиях на границе между металлом и полупроводником [28]. В нем практически нет процессов накопления зарядов, поэтому диод Шоттки имеет большее быстродействие, чем обычный полупроводниковый диод. И, кроме того, прямое падение напряжения на нем заметно меньше, чем на p-n переходе.

При изготовлении подобных схем по твердотельной технологии было выяснено, что диод Шоттки можно получить «бесплатно», как переход между металлическим выводом базы и полупроводниковым коллектором.

Подобные транзисторы с диодами Шоттки в обратной цепи называют транзисторами Шоттки. Их условное обозначение приведено на рис. 1.50.

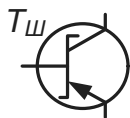


Рис. 1.50

Опишем работу ключа, считая, что в исходном состоянии транзистор был выключен, то есть на выходе действует большое напряжение $\varphi_k \approx E$, которое является обратным для коллекторного перехода (см. п. 1.6.6) и диода D_w .

Подадим на вход отпирающий единичный сигнал, который вызовет протекание тока $I_{RE} \approx U' / R_E$. Как видно из рис. 1.49, этот ток разветвляется на два направления, что позволяет найти основной для транзистора базовый ток, как разность:

$$I_B = I_{RE} - I_{OC}. \quad (1.67)$$

Потенциал коллектора φ_k в первый момент после скачка входного напряжения из-за инерционности транзистора не изменится, то есть на диоде останется обратное напряжение и ток диода I_{OC} будет практически нулевым. Поэтому, как следует из (1.67) при $I_{OC} = 0$ весь ток I_{RE} пойдет в базу, обеспечивая форсированное накопление заряда в базе. Далее происходит процесс включения, появляется коллекторный ток, и напряжение на коллекторе будет уменьшаться согласно (1.26).

Как только потенциал коллектора φ_k сравняется с потенциалом базы φ_B , это будет означать выход транзистора на границу активного режима и режима насыщения, то есть окончание процесса включения. Далее потенциал φ_k станет меньше потенциала φ_B , то есть напряжение на коллекторном переходе и диоде станет прямым, и в этот момент начинает играть свою роль диод.

Поскольку прямая характеристика диода Шоттки близка к идеальной, он начнет проводить при напряжении меньшем напряжения отпирания $U_{отп}$ коллекторного перехода (см. рис. 1.33), поэтому проводящее состояние диода исключает прямое смещение коллекторного перехода.

С другой стороны появление тока диода I_{OC} , как видно из (1.67), приводит к уменьшению базового тока. Однако он будет достаточным, чтобы поддерживать транзистор на границе насыщения.

Таким образом, форма тока I_B в этом ключе будет похожа на оптимальную. Разброс параметров транзистора в этой схеме влиять не будет, так как диод в обратной связи сам определяет момент насыщения транзистора и уменьшает ток I_B до необходимого уровня.

1.10.6. Преимущества и недостатки ключа с нелинейной ОС

За счет исключения работы транзистора в режиме насыщения в схеме с нелинейной ОС отсутствует накопление в базе избыточного заряда $Q_{изб}$, поэтому нет этапа рассасывания при любых значениях тока I_{RB} через резистор R_B . Это позволяет резко увеличить отпирающий ток I_B' без последствий для процесса выключения транзистора и, следовательно, существенно повысить быстродействие.

С другой стороны ключ с нелинейной ОС имеет два недостатка, *невысокие помехозащищенность и температурную стабильность.*

Первый недостаток связан с тем, что транзистор работает на границе активного режима и поэтому может наряду с полезным сигналом усиливать и помехи. Второй недостаток проявляется в существенном изменении параметров выходного нулевого сигнала с температурой. Обычно эти особенности ключа с нелинейной ОС учитывают при разработке БИС, минимизируя их технологически и схемными методами.

Ненасыщенный ключ на основе транзистора Шоттки широко используется в быстродействующих микросхемах на биполярных транзисторах.

2 ИНТЕГРАЛЬНЫЕ ЛОГИЧЕСКИЕ ЭЛЕМЕНТЫ

В этой главе приводятся начальные сведения о ключевых и логических элементах, используемых при построении более сложных вычислительных устройств. Рассмотрены их принципы действия, базовые схемы, основные параметры и характеристики.

§2.1. Разновидности логических интегральных элементов на биполярных транзисторах

2.1.1. Резистивно-транзисторная логика (РТЛ)

Первый полностью полупроводниковый (твердотельный) логический элемент был выполнен на основе транзисторного ключа с резистивной связью (см. п. 1.7.1). Основой этого РТЛ элемента (для простоты на рис. 2.1а показано только два входа) является резистивная схема сложения токов, которая создает выходной базовый ток, когда хотя бы один из входных сигналов единичен (**или** первый, **или** второй, **или** оба).

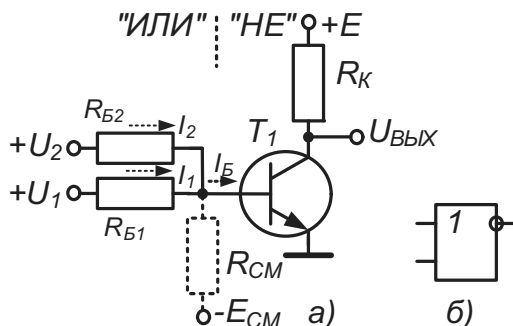


Рис. 2.1

Если все входные сигналы будут нулевыми, то транзистор будет находиться в режиме глубокой отсечки (см. п. 1.6.2) за счёт действия отрицательного E_{CM} (показано пунктиром), и выходное напряжение будет единичным.

Когда один из входных сигналов станет единичным, потечет ток через соответствующий резистор R_b , появится базовый ток, транзистор насытится, и выходной сигнал станет нулевым. Поэтому нетрудно видеть, что данная схема реализует функцию логического элемента «ИЛИ-НЕ», УГО которого приведено на рис. 2.1б.

Однако, несмотря на простоту, подобные РТЛ элементы из-за низкого быстродействия применялись недолго, потому что транзистор при наличии нескольких единиц на входе чрезмерно перенасыщался. При суммарном токе через резисторы R_b в базе накапливался избыточный заряд, в несколько раз превышающий граничный заряд, необходимый для насыщения, что резко увеличивало время выключения (см. п. 1.9.1).

Ускоряющие конденсаторы в этой схеме использовать нельзя, так как во время перепада сигнала конденсаторы будут эквивалентны «закоротке» (см. п. 1.2.3). Из-за этого нулевой сигнал одного из входных источников может влиять на единичный другого. Отметим также, что проблему мог бы решить ключ с нелинейной ОС (см. п. 1.10.5), но он еще не был изобретен.

2.1.2. Диодно-транзисторная логика (ДТЛ)

Чтобы устранить перенасыщение транзистора, свойственное предыдущей схеме, была предложена комбинация диодной логической схемы «И» и ключевого инвертора (рис. 2.2).

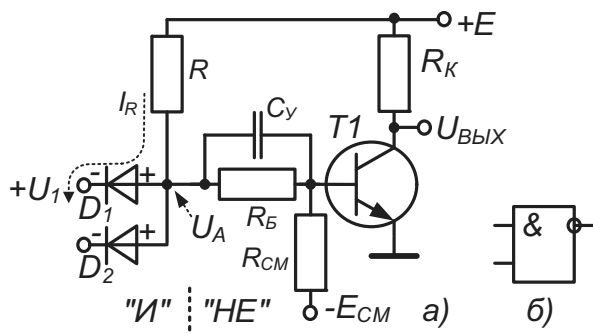


Рис. 2.2

Такое построение позволяет избавиться от избыточного перенасыщения транзистора, поскольку единичный выходной сигнал схемы «И» (напряжение U_A) не зависит от числа активных входов. Он будет единичным только тогда, когда на всех входах будут единицы. Если хотя бы на одном входе будет ноль, то соответствующий диод будет проводить (ток I_R на схеме) и фиксировать напряжение U_A на примерно нулевом уровне (см. п. 1.4.2).

В этой схеме можно уже без проблем включить ускоряющий конденсатор, который изолирован от входных сигналов диодами схемы «И», и получить, таким образом, значительно большее быстродействие, чем в РТЛ.

Но в наследство от РТЛ все-таки остался небольшой недостаток – наличие дополнительного источника смещения. Чтобы исключить этот не всегда имеющийся источник была предложена модифицированная схема.

2.1.3. ДТЛ элемент с одним источником питания

В этой схеме (рис. 2.3) используется только один источник питания E , а запирающего напряжения смещения нет. Поэтому в непроводящем состоянии транзистор работает в режиме, близком к режиму заземлённой базы (см. п. 1.6.4), который имеет несколько большие остаточные токи, что, тем не менее, допустимо для непроводящего ключа.

Однако входное напряжение ключа U_A имеет конечный нулевой уровень (см. формулу (1.10) в п. 1.4.1), что, как показано на рис. 2.4, может вызвать появление некоторого базового тока в непроводящем состоянии. Для того, чтобы транзистор был надёжно заперт, последо-

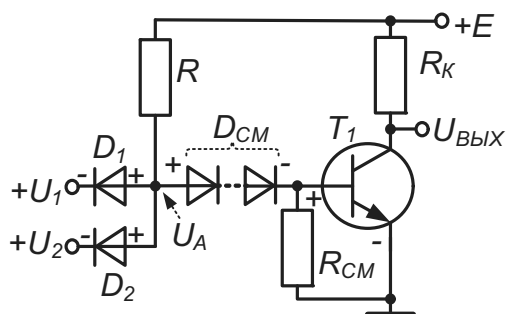


Рис. 2.3

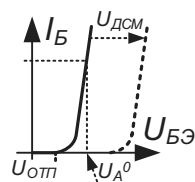


Рис. 2.4

вательно с базой транзистора включают несколько диодов D_{CM} , имеющих свое напряжение $U_{отп}$. В результате, как показано на рис. 2.4 пунктиром, порог отпирания входной цепи ключа увеличится, что обеспечивает, наряду с правильным выбором сопротивления R_{CM} (см. п. 1.6.4), надежное запираение транзистора.

Кроме того в схеме отсутствует резистор R_B , задающий базовый ток в режиме насыщения. Его роль выполняет резистор нагрузки R диодной схемы «И». Когда напряжение U_A единичное, то входные диоды D_1 и D_2 не проводят и поэтому базовый ток определяется сопротивлением R :

$$I_B = (E - U_{ДСМ} - U_{БЭ}) / R \quad (2.1)$$

то есть для правильного выбора величины R должно быть выполнено условие насыщения (1.27).

2.1.4. Многоэмиттерный транзистор

Подобные ДТЛ элементы в свое время достаточно широко использовались на практике. Причем для более плотной упаковки компонентов на кристалле размеры диодов уменьшали и сближали их друг с другом. В результате p - n переходы входных диодов и диода смещения стали взаимодействовать и влиять друг на друга, как в транзисторе.

Поэтому входную часть ДТЛ элемента (рис. 2.5а) начали изображать в виде многоэмиттерного транзистора T_M (рис. 2.5б). Причем его эмиттеры играют роль диодов логической схемы «И», а коллектор - диода смещения.

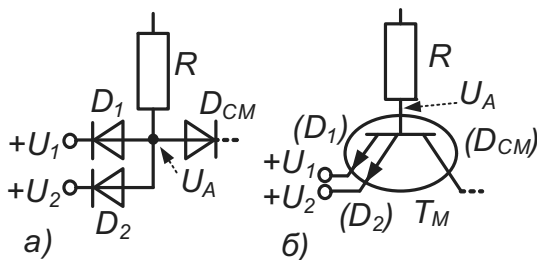


Рис. 2.5

Таким образом, такой элемент ДТЛ по мере совершенствования технологии стал рассматриваться как элемент транзисторно-транзисторной логики (ТТЛ), основанной на использовании многоэмиттерного транзистора.

§2.2. Структура ТТЛ элемента

2.2.1. Иллюстративная схема ТТЛ элемента

Основой схемы (рис. 2.6) является многоэмиттерный транзистор T_M , который включает в себя, по сути, входные диоды и диод смещения из предыдущей ДТЛ схемы, то есть на эмиттерных переходах реализуется, как и ранее, диодная логическая схема «И». Роль инвертора выполняет ключевой транзистор T_1 . Количество входных сигналов может быть увеличено за счет добавления эмиттерных переходов в транзисторе T_M .

Если хотя бы один из входных сигналов нулевой, то соответствующий эмиттерный переход T_M будет смещен в прямом направлении и ток I_R логической схемы «И» будет протекать, минуя базовую цепь ключевого транзистора T_1 . Поэтому транзистор T_1 находится в непроводящем состоянии и напряжение на выходе будет единичным.

В том случае, когда на всех входах будут единицы, эмиттерные переходы, как прежде диоды в ДТЛ элементе, окажутся в непроводящем состоянии, и ток I_R переключится в базовую цепь транзистора T_1 (пунктир I_R на рис. 2.6), обеспечивая насыщение транзистора и нулевой уровень выходного сигнала.

Недостатком однотранзисторной выходной цепи ТТЛ элемента является невысокое быстродействие при работе на выходную паразитную емкость (см. п. 1.2.1). Длительность положительного фронта выходного сигнала оказывается зависящей от ограничивающего коллекторный ток резистора R_K , сопротивление которого нельзя выбрать менее чем, указано в неравенстве (1.2).

2.2.2. Двухтранзисторный выходной каскад

Для повышения быстродействия выходной каскад инвертора выполняют двухтранзисторным (рис. 2.7). В этой схеме транзисторы включаются по очереди – когда один проводит, другой не проводит. Однако данный каскад выполнен на биполярных транзисторах одного типа проводимости и поэтому для противофазного управления

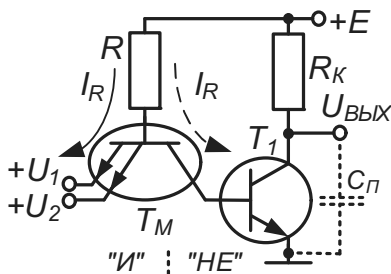


Рис. 2.6

транзисторами приходится дополнительно использовать схему управления СУ (см. п. 2.3.2), показанную на рисунке пунктиром.

Отметим, что при выполнении двухтранзисторного каскада на разнотипных полевых транзисторах (см. п. 2.6.5) можно обойтись без схемы управления.

В схеме, приведенной на рис. 2.7, при образовании единицы на выходе заряд паразитной емкости током I_3 происходит в момент насыщения транзистора T_2 (транзистор T_1 при этом не проводит). Сопротивление насыщенного биполярного транзистора не превышает единиц Ом, поэтому длительность положительного фронта выходного сигнала будет весьма малой.

Разряд паразитной емкости током I_p при формировании нулевого выходного сигнала также будет происходить форсировано через малое сопротивление насыщенного транзистора T_1 (T_2 не проводит).

Таким образом, двухтранзисторный каскад отличается высоким быстродействием, однако, как водится, не обошлось и без недостатков.

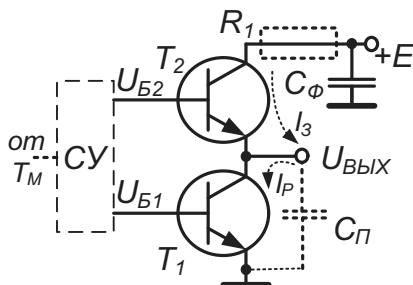


Рис. 2.7

2.2.3. «Сквозной» ток двухтранзисторного выходного каскада

Транзисторы T_2 и T_1 являются инерционными устройствами. Это приводит к тому, что оба транзистора могут одновременно (на очень малое время) оказаться в проводящем состоянии. Например, транзистор T_1 уже насытился, а транзистор T_2 ещё не успел перейти в режим отсечки тока. В этом случае источник питания будет нагружен очень малыми внутренними сопротивлениями двух проводящих транзисторов, то есть возникнет *экстремальный* ток потребления $I_{\text{ПОТР}}$ от источника питания (рис. 2.8).

Величина этого кратковременно-го импульса «сквозного» тока, будет

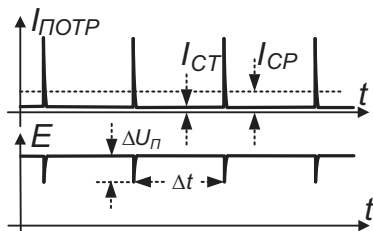


Рис. 2.8

очень большой, значительно превышающей статический ток потребления I_{CT} . При этом не только увеличивается средний потребляемый ток I_{CP} , но в эти моменты могут возникнуть провалы напряжения питания ΔU_{Γ} , так как источник не рассчитан на столь значительный ток.

На осциллографе такие очень короткие импульсы (иногда называемые «иголками») трудно обнаружить, однако они могут вызвать ложное срабатывание других схем и привести к ошибкам в работе устройств.

Заметим, что подобное кратковременное состояние в принципе может возникнуть и в двухтранзисторном комплементарном ключе (см. п. 2.6.5), но внутренние сопротивления полевых транзисторов значительно больше и к существенным последствиям (кроме некоторого увеличения потребляемого тока) это не приводит.

Полностью устранить сквозной ток технологически не удастся. Поэтому для уменьшения его величины в схему выходного каскада (рис. 2.7) вводится ограничивающее сопротивление R_1 , которое выбирается из компромиссных соображений. С одной стороны, чтобы уменьшить сквозной ток до приемлемой величины. А с другой – таким, чтобы не затягивать сильно длительность положительного фронта при заряде паразитной емкости. Величина этого резистора выбирается производителем для стандартных ТТЛ схем на уровне 100 Ом, а для быстродействующих серий менее 50 Ом.

Кстати, резистор R_1 дополнительно работает в качестве своеобразного предохранителя от перегрузки выходного каскада при коротком замыкании выхода на общую точку.

Таким образом, сквозные токи в схеме все равно остаются, но меньшие по величине. Остаются и провалы в напряжении питания. Чтобы их уменьшить рекомендуется подключать к источнику питания фильтрующий конденсатор C_{Φ} (рис. 2.7) с малой паразитной индуктивностью. Причём, подключать нужно непосредственно к выводам микросхемы, чтобы исключить индуктивность соединительных проводов. Напряжение на ёмкости скачком измениться не может, поэтому кратковременный сквозной ток будет обеспечиваться емкостью, и напряжение питания будет изменяться незначительно.

При оценке величины C_{Φ} можно пользоваться известной формулой, подобной (1.62):

$$C_{\Phi} = \Delta Q_C / \Delta U_C \approx I_{CP} \Delta t / \Delta U_{\Gamma} \quad (2.2)$$

задаваясь необходимыми значениями I_{CP} , Δt и ΔU_{Γ}

2.2.4. Программное управление двухтранзисторным каскадом

При управлении мощными источниками импульсов, например, в системах электропитания или в электроприводе, иногда используется двухтранзисторная схема на основе мощных транзисторов, в которой также могут возникнуть сквозные токи. Но уровень этих многоамперных токов будет очень большим, и транзисторы могут сгореть. Учитывая, что управление подобными ключами зачастую осуществляется программно, сквозные токи устраняют алгоритмически – сначала выключают один транзистор, делают паузу на время выключения, а потом только включают другой (рис. 2.9).

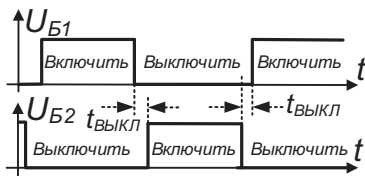


Рис. 2.9

§2.3. Базовая схема ТТЛ элемента

Имеется множество разновидностей (серий) интегральных схем ТТЛ типа, которые отличаются быстродействием, потребляемой мощностью, функциональной полнотой. Но все они имеют общую структуру, которая описана ниже.

2.3.1. Функциональная схема

ТТЛ элемент состоит из логического элемента «И» на многоэмиттерном транзисторе и логического инвертора (рис. 2.10). Учитывая, что для обеспечения высокого быстродействия инвертор выполняют на двух транзисторах, дополнительно используется схема управления СУ (см. п. 2.2.2) с двумя противофазными сигналами, которую иногда называют «фазоразделителем», схема которого описана далее.

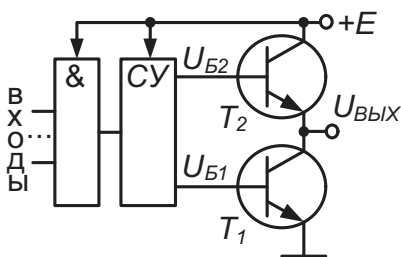


Рис. 2.10

2.3.2. «Фазоразделитель»

представляет собой транзисторный каскад с резисторами в эмиттерной и коллекторной цепях (рис. 2.11). Для этой схемы сигнал U_{B1}

на эмиттере транзистора T_3 равен падению напряжения от протекания эмиттерного тока:

$$U_{B1} \approx I_3 R_3 \approx (B+1)I_B R_3 \quad (2.3)$$

и совпадает по фазе с входным током I_B .

В то время как сигнал U_{B2} на коллекторе определяется разностью:

$$U_{B2} = E - I_K R_K \approx E - BI_B R_3, \quad (2.4)$$

в которой знак «минус» свидетельствует о том, что напряжение U_{B2} находится в противофазе с входным током I_B , поскольку при увеличении этого тока напряжение уменьшается.

Таким образом каскад на транзисторе T_3 имеет два противофазных выхода, обеспечивающих поочередную работу транзисторов T_1 и T_2 .

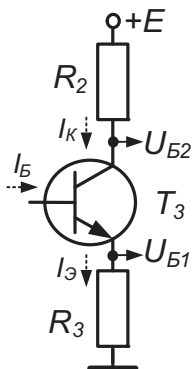


Рис. 2.11

2.3.3. Базовая схема ТТЛ элемента

Согласно функциональной схеме в принципиальной (рис. 2.12) можно выделить три составные части – логическую схему «И» (на многоэмиттерном транзисторе T_M), «фазоразделитель» (на транзисторе T_3) и выходной каскад (на транзисторах T_1 и T_2).

Значения сопротивлений твердотельных резисторов, приведенные для сведения на схеме в скобках, имеют большой технологический разброс и температурный дрейф.

Кроме перечисленных основных элементов, обеспечивающих реализацию функции «И-НЕ», в схеме имеются и ряд вспомогательных

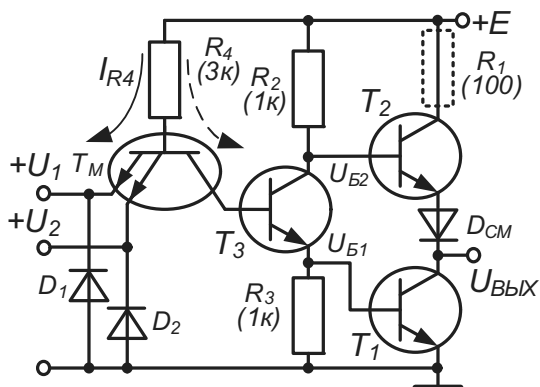


Рис. 2.12

элементов. Резистор R_1 , как отмечалось, уменьшает уровень «сквозного» тока. Диоды D_1 и D_2 предохраняют вход элемента от большого отрицательного напряжения и работают в качестве ограничителей (см. п. 1.5.2).

Эти диоды иногда называют демпфирующими (сглаживающими колебания). Термин связан с тем, что входной сигнал может приобрести колебательный характер из-за паразитных емкости и индуктивности линии связи. Указанные диоды, открываясь в момент появления отрицательной полуволны, ограничивают колебания, как бы сглаживают их.

Диод D_{CM} обеспечивает более надежное запираание транзистора T_2 , увеличивая порог отпираания (см. п. 2.1.3) и компенсируя тем самым неидеальность транзисторов T_1 и T_3 в режиме насыщения.

Производители микросхем обычно вносят в базовую схему дополнительные элементы для улучшения некоторых характеристик. Но подобные тонкости здесь не рассматриваются, поскольку пользователя в первую очередь интересует входная и выходная цепи, их характеристики и особенности передачи информации между этими цепями.

2.3.4. Работа ТТЛ элемента

Схема «И-НЕ» может иметь два основных состояния выходного сигнала, единичное, когда хотя бы один из входных сигналов нулевой, и нулевое, когда все входные сигналы единичные.

Пусть для примера напряжение $U_1 = 0$, поэтому соответствующий эмиттерный переход транзистора T_M будет смещен в прямом направлении и ток I_{R_4} будет протекать, как указано на рис. 2.12. Коллекторная цепь транзистора T_M при этом будет обесточена, что означает отсутствие базового тока транзистора T_3 и нахождение последнего в непроводящем состоянии.

В этом случае база T_1 также будет лишена управляющего сигнала и подключена к земле через R_3 , то есть транзистор T_1 также не будет проводить. Поэтому ток через сопротивление R_2 будет идти полностью в базу транзистора T_2 , насыщая последний и обеспечивая единичное выходное напряжение, близкое к напряжению питания (за вычетом падения напряжения в выходной цепочке):

$$U_{Вых} = E - U_{R_1} - U_{K\beta 2} - U_{DCM} \approx Ek - 1,5 B = U'. \quad (2.5)$$

Во втором состоянии ТТЛ элемента, когда все входные сигналы единичные, они смещают эмиттерные переходы T_M в обратном на-

правлении, и ток резистора R_4 , показанный на рис. 2.12 пунктиром, переключается поэтому в коллекторную цепь транзистора T_M и в базу T_3 . Транзистор T_3 насыщается и направляет ток через R_2 в базу T_1 , насыщая его.

Транзистор T_2 будет находиться в непроводящем режиме «заземленной» базы (см. п. 1.6.4), так как его база через проводящие транзисторы T_3 и T_1 оказывается подключенной к его эмиттеру. Диод D_{CM} при этом компенсирует остаточные напряжения насыщенных транзисторов T_3 и T_1 за счет увеличения напряжения отпирания эмиттерного перехода (см. п. 2.1.3). Таким образом выходное напряжение при всех входных единичных сигналах определяется насыщенным транзистором T_1 :

$$U_{Вых} = U_{кн1} \approx 0,05..0,4 \text{ В} = U^0 \quad (2.6)$$

Выражения (2.5) и (2.6) использовались в свое время при определении границ для стандартных уровней нуля и единицы (см. п. 2.4.1).

2.3.5. Работа ТТЛ элементов на общую информационную линию

При сборе данных выходные сигналы от нескольких источников информации должны подаваться на общую линию связи, то есть выходы элементов приходится соединять друг с другом. Но при этом возникает проблема – конкуренция источников сигнала. Поясним ее, используя рис. 2.13, на котором изображены объединенные выходные каскады двух ТТЛ элементов.

Пусть один из элементов имеет единичный, а другой нулевой выходной сигналы. Из рисунка видно, что при таком подключении к общей линии, выходной сигнал будет не определен. Кроме того в результате конфликта может образоваться цепь из двух проводящих ключей в разных элементах (на рисунке K_1 и K_2), и источник питания будет работать в недопустимом режиме короткого замыкания.

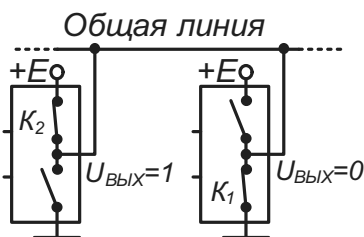


Рис. 2.13

Для исключения конкуренции источников данных используют элементы со специальными выходными каскадами – с открытым коллектором и третьим состоянием.

2.3.6. ТТЛ элемент с открытым коллектором

имеет одностранзисторный выходной каскад, то есть в базовой схеме рис. 2.12 отсутствует транзистор T_2 и элементы его обрамления (R_1 и $D_{см}$). И в выходном каскаде остается только транзистор T_1 , коллектор которого является выходом и *открыт* для внешних подключений. Однако для образования единичного сигнала на выходе обычно необходим внешний нагрузочный резистор R_K (взамен транзистора T_2).

Подобный выход можно соединять с аналогичными выходами других элементов, не опасаясь последствий конкуренции сигналов. Для примера рассмотрим использование каскада с открытым коллектором при подключении к общей линии данных (рис. 2.14а), при котором образуется так называемое «монтажное ИЛИ» (см. п. 1.4.4).

Из рисунка видно, что, благодаря отсутствию в выходном каскаде второго (верхнего) ключа, конфликт нуля и единицы не приводит к короткому замыканию источника питания.

Не показанная на рисунке внешняя управляющая схема, входящая в состав такого неявного коммутатора (см. п. 11.8.5), разрешает поочередную работу только одного логического элемента, используя принцип «монтажного ИЛИ».

УГО элемента с открытым коллектором отличается от обычного тем, что выход помечается ромбом с чертой внизу (рис. 2.14б)

Выходной каскад с открытым коллектором, как правило, допускает повышенное предельное напряжение и поэтому внешний резистор (или иную нагрузку) можно подключать к отдельному источнику с большим напряжением 15..30 В. В этом смысле подобный выход считается универсальным, поскольку единичный выходной уровень будет определяться выбранным напряжением дополнительного источника питания.

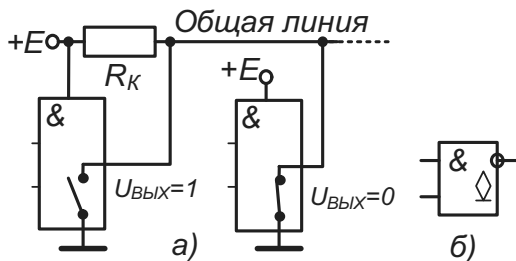


Рис. 2.14

Недостатком элемента с открытым коллектором, как и всякого однотранзисторного каскада, является плохое быстродействие при работе на паразитную емкость, зависящее от величины нагрузочного резистора (см. п. 1.2.1). Поэтому такой элемент применяется в низкоскоростных устройствах, например, малых сетях типа I^2C [25].

2.3.7. ТТЛ элемент с третьим состоянием

Чтобы обеспечить бесконкурентное подключение быстродействующего двухтранзисторного выходного каскада к общей информационной линии, используется принцип третьего состояния. Для иллюстрации его структуры представим два ключа каскада в виде перекидного, имеющего три положения (рис. 2.15а), единичное, нулевое и промежуточное (третье), когда оба ключа не проводят.

УГО элемента с третьим состоянием имеет выход, помеченный ромбом с чертой посередине (рис. 2.15б), и дополнительный входной сигнал, который является управляющим и обозначается аббревиатурой **OE (Output Enable** – выход разрешен).

Пассивный уровень (обычно единичный) указанного управляющего сигнала переводит оба ключа выходного каскада в непроводящее (третье) состояние. Другие особенности управляющих цепей в ТТЛ элементе выходят за рамки настоящего пособия и для простоты не рассматриваются.

Элементы, находящиеся в третьем состоянии, как нетрудно видеть, не могут влиять на сигнал общей линии, который будет определяться только одним элементом, работа которого будет разрешена активным уровнем управляющего сигнала, благодаря чему конфликт источников информации будет исключен.

Каскады с третьим состоянием входят в состав более сложных функциональных узлов, называемых неявными коммутаторами (см. п. 11.8.6), обеспечивая, например регистрам или микросхемам памяти быстрый и бесконфликтный доступ к общим микропроцессорным магистралям (см. п. 14.5.1).

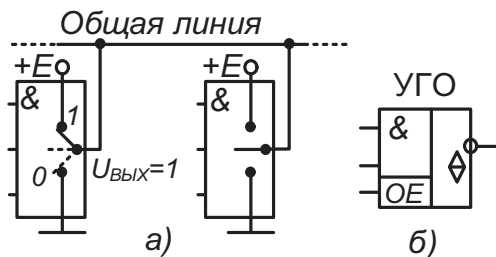


Рис. 2.15

§2.4. Параметры и характеристики ТТЛ элемента

2.4.1. Основные параметры

Изначально для ТТЛ элементов было определено стандартное напряжение питания $E = +5 \text{ В}$ и границы двоичного информационного сигнала:

$$U^0 \leq 0,4 \text{ В}, \quad (2.7)$$

$$U^1 \geq 2,4 \text{ В}. \quad (2.8)$$

Отметим, что в настоящее время для экономии энергии были определены и более низкие уровни напряжения питания, например 3,3 В при сохранении значений нуля и единицы.

Важным параметром для вычислительных устройств, в особенности питающихся от батарей и аккумуляторов, является потребляемая мощность, которая зависит от разновидности элемента. Стандартный ТТЛ элемент в статике (на низкой частоте) потребляет около 10 мВт, элемент микромощных серий – в несколько раз меньше. При увеличении частоты сигнала до предельной потребляемая мощность возрастает приблизительно в два раза.

Увеличение мощности при повышении частоты связано с зарядом и разрядом паразитных ёмкостей внутри схемы или снаружи. При заряде паразитной ёмкости потребляется ток от источника питания, при её разряде энергия превращается в тепловую (рассеивается на внутренних элементах). Как правило, потребляемая мощность связана с предельной рабочей частотой. Чем больше предельная частота, тем больше потребляемая мощность.

Другие параметры обычно находят из трех различных типов характеристик ТТЛ элемента:

- *входной характеристики* (зависимости входного тока от входного напряжения), условно обозначаемой как $I_{\text{вх}} = F(U_{\text{вх}})$,
- *передаточной* (зависимости выходного напряжения от входного) – $U_{\text{вых}} = F(U_{\text{вх}})$,
- *выходной* (зависимости выходного напряжения от выходного тока) – $U_{\text{вых}} = F(I_{\text{вых}})$.

2.4.2. Входная характеристика

снимается для одного из входов с помощью вольтметра и амперметра (рис. 2.16а), имеет резко нелинейный вид (рис. 2.16б) и на ней можно выделить несколько участков.

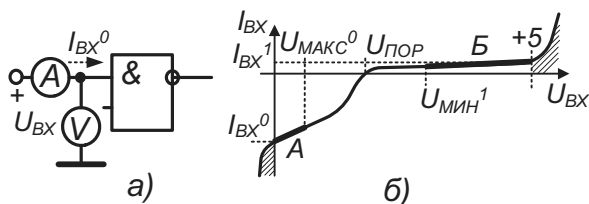


Рис. 2.16

Во-первых, *нерабочие* участки (на рисунке они заштрихованы), когда входное напряжение отрицательное или превышает напряжение питания 5 В . При отрицательном напряжении открывается входной ограничивающий диод (D_1 или D_2 на рис. 2.12), защищая вход от попадания большого отрицательного напряжения (см. п. 1.5.2). При напряжении большем 5 В на эмиттерном переходе транзистора T_M действует обратное напряжение и он может пробиться, что вызывает резкое увеличение входного тока и тепловой (невосстанавливаемый) пробой. Этот режим на практике должен быть исключен, так как может привести к отказу ТТЛ элемента.

В *рабочей области* входного сигнала (от 0 до $+5\text{ В}$) можно выделить два режима: нулевой (участок «А») и единичный (участок «Б»). Этим режимам соответствуют входные токи I_{BX}^0 и I_{BX}^1 , значения которых часто используются на практике

2.4.3. Определение входных токов ТТЛ элемента

Пусть на входе действует нулевой сигнал U_{BX}^0 . Из рис. 2.17, на котором приведена только входная цепь ТТЛ элемента, видно, что ток через резистор R_4 идет во входную цепь, то есть ток I_{BX}^0 вытекает из входного зажима и поэтому считается отрицательным (см. рис. 2.16б).

Найдем величину входного тока, учитывая, что он равен току через резистор R_4 :

$$I_{BX}^0 = I_{R4} = (E - U_{БЭМ} - U_{BX}^0) / R_4. \quad (2.9)$$

Используя значение R_4 , приведенное на рисунке в скобках, можно оценить ток $I_{BX}^0 \approx (5 - 0,6 - 0,4) / 3000 \approx 1,3\text{ мА}$. Производители обычно указывают максимальные значения, например [14], для стандартной схемы серии К155 ток $I_{BX}^0 = -1,6\text{ мА}$. Знак минус иногда опускают, но помнить об отрицательном направлении нужно.

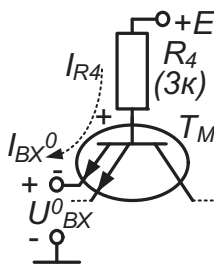


Рис. 2.17

Конкретные величины токов зависят от разновидности ТТЛ элемента, например, для микромощных серий I_{BX}^0 меньше примерно в два раза, а для быстродействующих наоборот больше.

В том случае, когда на входе действует единичный сигнал, эмиттерный переход транзистора T_M будет смещен в обратном направлении, и ожидаемое значение входного тока должно быть достаточно малым (микроамперным). Однако в это время коллекторный переход T_M проводит, и транзистор оказывается работающим в *инверсном активном режиме*, то есть обратносмещенный эмиттерный переход «перехватывает» часть прямого тока коллекторного перехода. В результате реальная величина I_{BX}^1 на практике оказывается большей и составляет десятки мкА. Например, для стандартной 155 серии считается [14], что $I_{BX}^1 \leq 40$ мкА.

2.4.4. Определение порогового напряжения

Важным параметром ТТЛ элемента, определяющим вид входной и передаточной характеристики, а также помехозащищенность элемента, является напряжение, при котором входной ток (см. рис. 2.16б) меняет свое направление. Это напряжение называют пороговым и определяют из уравнения для входной цепи, приведенной на рис. 2.18, считая, что в момент смены направления тока напряжение $U_{ЭБМ}$ на эмиттерном переходе равно нулю:

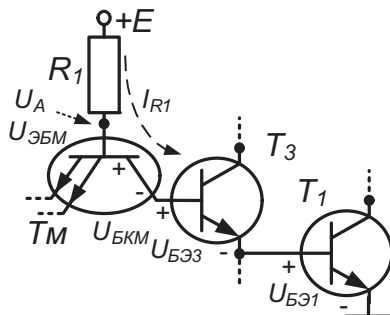


Рис. 2.18

$$U_{BX} = U_{ПОР} = U_{ЭБМ} + U_A \approx U_A. \quad (2.10)$$

Записывая далее напряжение U_A через напряжения на переходах транзисторов входной цепи, можно найти пороговое напряжение:

$$U_A = U_{БКМ} + U_{БЭ3} + U_{БЭ1}. \quad (2.11)$$

И в заключение, учитывая, что ток через переход появляется, когда напряжение станет больше напряжения отпирания $U_{отп}$ (см. рис. 1.34), можно примерно оценить пороговое напряжение:

$$U_{ПОР} = U_A \approx 3U_{отп} \approx 1,5 \text{ В}. \quad (2.12)$$

Отметим, что пороговое напряжение сильно меняется при изменении температуры, поскольку каждое из его составляющих

имеет температурный коэффициент напряжения (ТКН) примерно равный $-2...3$ мВ/градус [28], то есть суммарный ТКН в три раза больше.

2.4.5. Передаточная характеристика

снимается с использованием одного из входов, когда на остальных будут единицы. Эта зависимость напряжения $U_{ВЫХ}$ от $U_{ВХ}$ (рис. 2.19) является инвертирующей (при нуле на входе – на выходе единица и наоборот).

Заметим, что ТТЛ элементы с неинвертирующей характеристикой обычно реализуются на основе двух инверторов и имеют поэтому худшее быстродействие.

Рабочими участками для цифровых схем характеристики остаются, как и прежде, нулевой и единичный, помеченные буквами «А» и «Б». На остальных участках передаточной характеристики элемент находится короткое время в моменты переключения.

Вид передаточной характеристики определяют три параметра – $U_{ВЫХ}^1$, $U_{ВЫХ}^0$ и $U_{ПОР}$. Каждый из этих параметров имеет технологический разброс, поэтому производитель описывает передаточную характеристику для худшего случая, используя на рабочих участках граничные значения $U_{МАКС}^0$ и $U_{МИН}^1$ (см. п. 2.4.1), при которых гарантируется работоспособность логической схемы (пунктирные характеристики на рисунке).

На работе цифровой схемы этот разброс не сказывается, но в нестандартном режиме, когда ТТЛ элемент используют на крутом участке передаточной характеристики в пороговых или времязадающих устройствах (см. п.п. 17.2.3 и 17.5.6), большой температурный дрейф $U_{ПОР}$ может нарушить работу устройства.

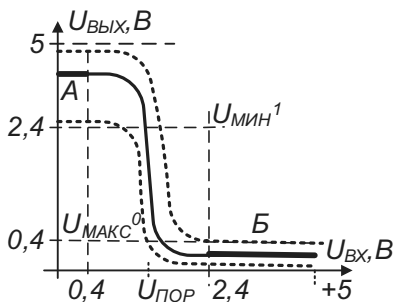


Рис. 2.19

2.4.6. Особенности выходных (нагрузочных) характеристик

Выходное напряжение имеет два уровня, поэтому и характеристик две, причем снимаются они при разных направлениях выходного тока, чтобы учесть двуполярность нагружающей входной характе-

ристики (см. п. 2.4.2): при единице – положительную, при нуле – отрицательную.

Единичная характеристика снимается по схеме рис. 2.20 при единичном выходном напряжении (ноль на входе). Потенциометр R_{Π} , включенный как переменный резистор, имитирует нагрузку, в качестве которой в большинстве случаев выступает вход другого логического элемента. Ток нагрузки $I_{ВЫХ}^1$ при этом вытекает из выходного зажима и его направление совпадает с направлением тока $I_{ВХ}^1$ для предполагаемого нагружающего входа.

Нулевая характеристика должна сниматься для нулевого выходного напряжения (на входе единица) и при обратной полярности выходного тока, что обеспечивается подключением резистора R_{Π} уже не к общей точке, как ранее, а к источнику питания (рис. 2.21). Ток $I_{ВЫХ}^0$ при этом втекает в выходной зажим и совпадает по направлению с током $I_{ВХ}^0$ нагружающего входа.

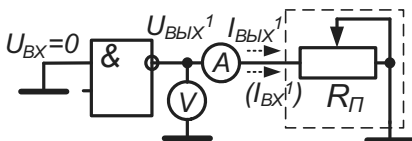


Рис. 2.20

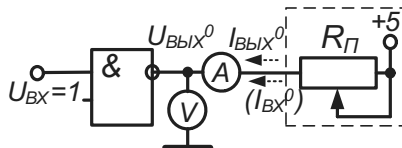


Рис. 2.21

Примечание: последовательно с потенциометром R_{Π} должен быть включен резистор около 100 Ом, не показанный на рисунке и предназначенный для исключения режима короткого замыкания выхода.

2.4.7. Единичная нагрузочная характеристика

снимается, как отмечалось, при $U_{ВЫХ}^1$, что обеспечивается проводящим состоянием транзистора T_2 (см. рис. 2.12). Для наглядности представим этот транзистор и окружающие его элементы на рис. 2.22, из которого нетрудно видеть, что ток $I_{ВЫХ}^1$, протекая по последовательной цепочке, создает падения напряжения U_{R1} , U_{KH2} , U_{DCM} . Причем напряжения на проводящих транзисторе (U_{KH2}) и диоде (U_{DCM}) будут слабо меняться с изменением тока. Поэтому основное влияние на изменение выходного напряжения будет оказывать напряжение U_{R1} , то есть при увеличении тока нагрузки величина $U_{ВЫХ}^1$ будет уменьшаться:

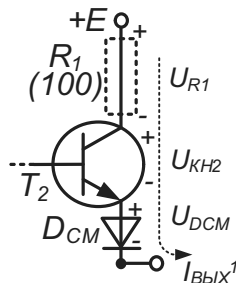


Рис. 2.22

$$U_{Вых}^1 = E - I_{Вых}^1 R_1 - U_{КН2} - U_{ДСТ} \quad (2.13)$$

Но в любом случае единичное выходное напряжение не должно быть менее чем $U_{МИН}^1 = 2,4 \text{ В}$ (см. п. 2.4.1), поэтому из нагрузочной характеристики, приведенной на рис. 2.23, можно определить максимально возможное значение выходного тока $I_{МАКС}^1$.

В некоторых случаях в справочниках вместо значения $I_{МАКС}^1$ приводится коэффициент разветвления K_P ограничивающий количество входов подобных элементов, которое можно подключить к выходу. Для стандартных ТТЛ схем $K_P = 10$, что позволяет сделать очень приближенную оценку $I_{МАКС}^1 = K_P I_{ВХ}^1 \approx 1...5 \text{ мА}$.

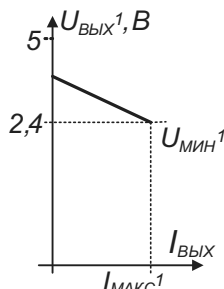


Рис. 2.23

2.4.8. Нулевая нагрузочная характеристика

снимается при другом направлении выходного тока (рис. 2.21) и зависит в соответствии с формулой (2.6) от напряжения насыщения $U_{КН1}$ проводящего транзистора T_1 . При увеличении (по модулю) тока $I_{Вых}$ увеличивается падение напряжения на внутреннем сопротивлении насыщенного транзистора и соответственно увеличивается значение $U_{Вых}^0$ (рис. 2.24), но оно не должно превышать величины $U_{МАКС}^0 = 0,4 \text{ В}$. Из этой характеристики обычно определяют максимальный выходной ток $I_{Вых}^0$ при низком (нулевом) уровне сигнала на выходе.

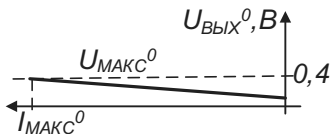


Рис. 2.24

Интересно отметить, что для ТТЛ элемента значение $I_{Вых}^0$ оказывается на порядок большим тока $I_{МАКС}^1$. Для примера можно примерно оценить величину $I_{МАКС}^0$ через коэффициент разветвления:

$$I_{Вых}^0 = K_P I_{ВХ}^0 \approx 16..20 \text{ мА}. \quad (2.14)$$

Эта неодинаковость токов объясняется необходимостью работы ТТЛ выхода на резко несимметричную входную характеристику следующего логического элемента.

2.4.9. Временные параметры

Инерционность логических элементов принято оценивать двумя параметрами, длительностью фронтов и задержкой формирования выходного сигнала.

Длительность фронтов обычно измеряют на уровнях 0,1 и 0,9 (см. п. 1.1.6). При прохождении сигнала по цепочке логических элементов длительность фронта почти не изменяется, поскольку каждый из элементов обладает формирующими свойствами, то есть гарантируется время фронта не более паспортного.

Однако в цепочке может *накапливаться* более опасная для работы устройств *задержка* выходного сигнала относительно входного.

Для оценки задержки, вносимой одним элементом, можно пользоваться осциллограммами, приведенными на рис. 2.25. Обычно различают два типа задержек (по уровню 0,5), t_3^{01} и t_3^{10} . Но в цепочке элементов количество и тех и других примерно одинаково и поэтому в расчетах используют среднюю для элемента задержку:

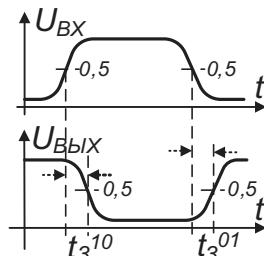


Рис. 2.25

$$t_{зср} = (t_3^{01} + t_3^{10}) / 2. \quad (2.15)$$

Средняя задержка элемента определяет максимальную возможную частоту, на которой может работать логический элемент.

Например, элемент серии К155 имеет $t_{зср}$ около 15 нс (зависит от емкостной нагрузки), а предельная рабочая частота – 10 МГц.

Однако осциллографы не всегда имеют нужное наносекундное разрешение. В этом случае используют способность цепочки логических элементов накапливать задержку. Для этого в кольцо (рис. 2.26) соединяют *нечётное* количество $K_{и}$ логических инверторов. В такой кольцевой схеме (см. п. 6.7.2) возникает генерация, причём период генерации T_r связан со средним временем задержки $t_{зср}$ соотношением:

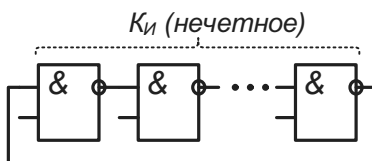


Рис. 2.26

$$T_r = 2K_{и} t_{зср} \quad (2.16)$$

позволяющим оценить значение времени задержки. Причем, чем больше элементов в так называемом *кольцевом генераторе*, тем больше период и легче его измерить.

§2.5. Нормализация входного сигнала для ТТЛ элемента

Существенную нелинейность входной характеристики ТТЛ элемента необходимо учитывать на практике, особенно в тех случаях, когда информационный сигнал образуется вне ТТЛ устройства.

2.5.1. Двухключевая схема

для имитации ТТЛ сигнала является наиболее предпочтительной схемой, поскольку моделирует структуру выходного двухтранзисторного каскада (рис. 2.27).

Однако на практике не всегда удастся ее реализовать. Например, при построении контактной клавиатуры клавиша имеет только один ключ, поскольку применение второго противофазного ключа приводит к существенному усложнению механической конструкции. Поэтому на практике применяют различные варианты одноключевых схем, формирующих ТТЛ сигнал.

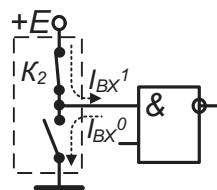


Рис. 2.27

2.5.2. Схема с одним верхним ключом

на первый взгляд (рис. 2.28) обеспечивает нужный логический сигнал на ТТЛ входе. Но это справедливо только для проводящего состояния ключа K_2 , когда имеем «честную» единицу на входе. Однако рассуждение, что непроводящий ключ не создает единичный сигнал и это можно считать наличием нуля, является *ложным*, поскольку источник нулевого ТТЛ сигнала обязательно должен пропускать нагружающий входной ток I_{BX}^0 . В противном случае эмиттерный переход многоэмиттерного транзистора логического элемента (см. п. 2.2.1) не будет проводить, будто на входе действует единичный уровень.

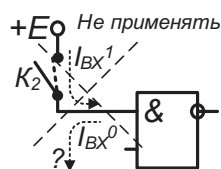


Рис. 2.28

Поэтому никуда не подключенный ТТЛ вход эквивалентен единичному состоянию на этом входе. Вообще говоря, оставлять вход неподключённым для ТТЛ элемента нежелательно. Ножка микросхемы, как миниатюрная антенна, воспринимает при этом электромагнитное излучение, что может уменьшить помехозащищённость устройства.

Таким образом схема с одним верхним ключом K_2 оказывается *не-работоспособной*, поскольку при коммутации ключа состояние ТТЛ схемы не изменяется. И причиной этого является отсутствие ключа K_1 , создающего путь для протекания тока I_{BX}^0 . Однако эту функцию может выполнять и более простое устройство – внешний резистор.

2.5.3 Схема с верхним ключом и резистором

Сопротивление R_{K1} в этой схеме (рис. 2.29) заменяет ключ K_1 и создаёт путь для протекания тока I_{BX}^0 , когда ключ K_2 не проводит. Однако такая замена не совсем эквивалентна.

Дело в том, что сопротивление проводящего ключа K_1 практически нулевое, тогда как сопротивление R_{K1} должно быть достаточно большим, чтобы ограничить ток потребляемый от источника питания, когда ключ K_2 проводит (см. п. 1.1.3).

Поэтому при протекании тока I_{BX}^0 на входе микросхемы возникает падение напряжения, которое должно быть воспринято элементом, как нулевое, то есть быть меньше порогового уровня $U_{ПОР}$ (см. п. 2.4.5). Из этого условия получается расчетное соотношение для выбора величины R_{K1} :

$$R_{K1} < U_{ПОР} / I_{BX}^0. \quad (2.17)$$

Пусть, например, $U_{ПОР} \approx 1,5 \text{ В}$, $I_{BX}^0 \approx 1,5 \text{ мА}$, тогда максимальное значение $R_{K1} \approx 1,5 / 1,5 \cdot 10^{-3} \approx 1 \text{ кОм}$. На практике величину R_{K1} выбирают меньшей примерно в 2-3 раза для увеличения помехозащищённости, потому что, если входное напряжение будет вблизи порогового, то небольшая помеха на входе может вызвать переключение элемента.

Ключ K_2 в рассмотренной схеме оторван от общей точки и им, если он транзисторный, затруднительно управлять, поэтому предпочтительней использовать нижний ключ K_1 , а ключ K_2 заменить резистором.

2.5.4. Схема с нижним ключом и резистором

подобна по смыслу предыдущей схеме (рис. 2.30), только резистор R_{K2} создает путь (вместо K_2) для протекания гораздо меньшего тока I_{BX}^1 . В этом случае ключ K_1 формирует нулевой сигнал и обеспечивает протекание тока

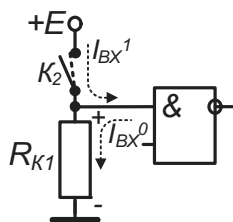


Рис. 2.29

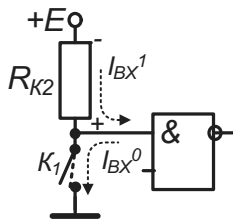


Рис. 2.30

I_{BX}^0 . Функции верхнего ключа K_2 и путь для протекания тока I_{BX}^1 обеспечиваются резистором R_{K2} .

При выборе величины R_{K2} , как и ранее, нужно иметь в виду, что при протекании тока создается падение напряжения, изменяющее величину входного сигнала. Только этот сигнал должен восприниматься элементом как единичный, то есть $E - I_{BX}^1 R_{K2} \geq U_{МИН}^1$. Отсюда получается расчетное соотношение:

$$R_{K2} < (E - U_{МИН}^1) / I_{BX}^1 \quad (2.18)$$

Пусть, например, $E = 5 \text{ В}$, $U_{МИН}^1 \approx 2,4 \text{ В}$, $I_{BX}^1 \approx 0,1 \text{ мА}$, тогда максимальное значение $R_{K2} \approx (5 - 2,4) / 0,1 \cdot 10^{-3} \approx 26 \text{ кОм}$. Таким образом, сопротивление R_{K2} оказалось значительно большим по величине, чем R_{K1} , и поэтому при замкнутом положении ключа от источника питания потребляется меньший ток.

Чаще всего в качестве нижнего ключа K_1 используют транзисторный ключ, в расчете которого появляются определенные особенности.

2.5.5. Преобразователь двуполярного напряжения в ТТЛ уровень

В некоторых случаях на обработку поступает двуполярный сигнал с амплитудой 10..20 В, то есть с амплитудой и полярностью, не допустимыми для ТТЛ элемента.

Например, для связи с внешним устройством иногда используется последовательный порт стандарта RS-232 [6]. В нем биты передаются последовательно друг за другом по одной линии, причем единица передается уровнем минус 15 В, а ноль – уровнем плюс 15 В. То есть на входе приемного устройства действует достаточно большое двуполярное напряжение, которое нужно преобразовать в ТТЛ уровень.

На рис. 2.31 приведена подобная схема, выполненная на основе предыдущей схемы с нижним ключом и резистором R_{K2} , рассчитанным по формуле (2.18). По конфигурации эта схема представляет транзисторный ключ с резистивной связью (см. п. 1.7.1).

Только цепь смещения в нем отсутствует, поскольку во входном сигнале имеется достаточное

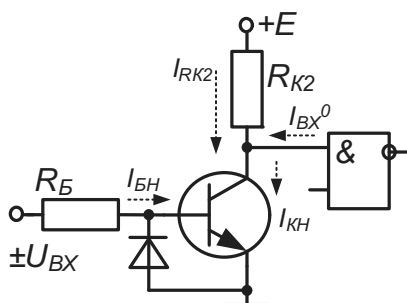


Рис. 2.31

отрицательное напряжение, которое обеспечивает непроводящее состояние транзистора. Более того, для защиты эмиттерного перехода от большого обратного напряжения приходится включать ограничивающий диод D_p , который при обратном сигнале фиксирует потенциал базы на небольшом отрицательном уровне, равном напряжению $U_{пр}$ на проводящем диоде (см. п. 1.5.2).

Когда на входе действует напряжение положительной полярности, эмиттерный переход транзистора смещается в прямом направлении и начинает протекать базовый ток $I_{БН}$, величина которого будет определяться сопротивлением R_B . При правильном расчете последнего транзистор должен быть насыщен и иметь напряжение $U_{КН} < 0.4 В$.

Однако в отличие от расчета типовой схемы (см. п. 1.7.3) в данном случае при расчете коллекторного тока необходимо учесть дополнительный ток $I_{ВХ}^0$ в коллекторной цепи (см. рис. 2.30), создаваемый входом нагружающей микросхемы:

$$I_{КН} = I_{РК2} + I_{ВХ}^0 = E / R_{К2} + I_{ВХ}^0 \quad (2.19).$$

Пусть для примера $R_{К2} = 20 кОм$ (см. п. 2.5.4), $I_{ВХ}^0 \approx 1,5 мА$ (см. п. 2.4.3), тогда коллекторный ток $I_{КН} = 5 / 20000 + 1,5 \cdot 10^{-3} = 1,75 мА$, то есть ток $I_{КН}$ будет почти полностью определяться входным током микросхемы. Это увеличенное значение следует учитывать при расчете требуемого значения базового тока из условия насыщения ($I_{БН} \geq I_{КН} / V_{мин}$) и выборе базового сопротивления ($R_B \leq U^1 / I_{БН}$).

Подобный преобразователь двуполярного напряжения можно выполнить и на основе транзисторного оптрона (см. п. 4.1.1).

2.5.6. Стабилитронный ограничитель двуполярного сигнала

Предыдущая схема инвертирует входной сигнал. Когда это не желательно, для нормирования сигнала можно использовать диодный (см. п. 1.5.3) или стабилитронный ограничитель (см. п. 1.5.4).

Рассмотрим ограничитель на основе стабилитрона (рис. 2.32), пробивное напряжение которого выбирается в середине диапазона единичного ТТЛ сигнала (2,4...5,0 В) с учетом возможного технологического разброса. Например, можно использовать стабилитрон КС139, который имеет напряжение пробоя около 3,9 В.

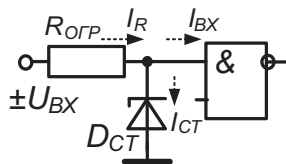


Рис. 2.32

При положительной полярности сигнала стабилитрон работает на обратной ветви характеристики, а при отрицательной – на прямой. Учитывая, что работа ограничителя описана ранее в п. 1.5.4, ограничимся особенностями расчета ограничивающего резистора $R_{огр}$. Дело в том, что этот резистор должен пропускать два тока, стабилитрона и микросхемы:

$$I_R = I_{CT} + I_{BX} \quad (2.20)$$

где ток I_{CT} выбирают обычно более 1.2 мА, а ток I_{BX} микросхемы в зависимости от входного сигнала имеет два значения, $I_{BX}^0 \approx 1,5$ мА и $I_{BX}^1 \approx 0,1$ мА. Отсюда видно, что током I_{BX}^1 без большой погрешности можно пренебречь. Поэтому при расчете учитывают только случай нулевого сигнала на входе микросхемы (входное напряжение отрицательно) и оценку сопротивления проводят по закону Ома:

$$R_{огр} \approx (U_{BX} - U_{пр}) / (I_{CT} + I_{BX}^0) \approx U_{BX} / (I_{CT} + I_{BX}^0). \quad (2.21)$$

Следует заметить, что подобную схему используют обычно на достаточно низких частотах, поскольку стабилитрон обладает заметной паразитной емкостью. Для высокочастотных сигналов рекомендуется применять более быстродействующий диодный ограничитель сверху (см. п. 1.5.3), учитывая, что снизу сигнал будет обрезать внутренний защитный диод ТТЛ микросхемы (см. п. 2.3.3).

Интересно отметить, что подобные диодные ограничители входного сигнала сверху и снизу встроены внутрь логических элементов на полевых транзисторах (см. п. 2.7.3).

§2.6. Ключи на полевых транзисторах

Полевые транзисторы, как известно, бывают с р-п переходом и с изолированным затвором. В логических ключевых схемах чаще всего используют полевые транзисторы с изолированным затвором, которые для краткости традиционно именуют МОП транзисторами.

2.6.1. Общие сведения о МОП транзисторах

В зависимости от конструктивного выполнения различают МОП транзисторы с *индуцированным* каналом, который возникает при напряжении на затворе больше порогового, и со *встроенным* каналом, который конструктивно реализуют внутри транзистора [28]. Последние в основном используют в усилительных схемах, тогда

как более простые в изготовлении транзисторы с индуцированным каналом являются основой ключевых и логических схем.

Отметим также, что в зависимости от полярности питающего напряжения в схемах используют n -канальные или p -канальные МОП транзисторы, УГО которых приведено на рис. 2.33. В большинстве случаев используются n -канальные транзисторы, совместимые по полярности питающего напряжения с ТТЛ схемами, хотя технологически p -канальные устройства легче в изготовлении.

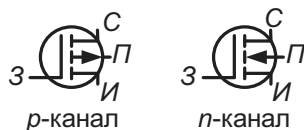


Рис. 2.33

МОП транзистор в дополнении к трем электродам, в наименовании которых используются “водяные” аналогии (управляющий затвор, выходные электроды сток и исток), имеет четвертый вывод — подложку, на которой реализуются в едином технологическом потоке все транзисторы БИС.

Из УГО видны некоторые особенности МОП транзисторов:

- затвор не имеет электрической связи с остальными электродами, чем подчеркивается его изолированность и отсутствие тока затвора,
- связь между стоком и истоком показана пунктирно, что должно напоминать об отсутствии канала при нулевом напряжении на затворе,
- стрелка у вывода подложки указывает направление *недопустимого* прямого тока через стоковый и истоковый p - n переходы транзистора.

В дальнейшем также будем учитывать, что в отличие от биполярного транзистора, который управляется током и является управляемым источником тока, полевой транзистор управляется напряжением и является управляемым резистором.

2.6.2. Ключ на полевом транзисторе

имеет минимум элементов (рис. 2.34) и схож с ранее рассмотренной базовой схемой рис. 1.1, имеющей только ключевой элемент и нагрузочный резистор. В качестве ключа для положительного напряжения питания используется n -канальный МОП транзистор с индуцированным каналом, то есть при нуле на

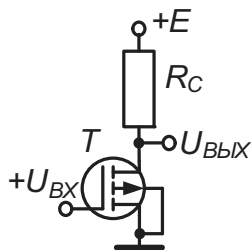


Рис. 2.34

затворе подобный транзистор не проводит. На выходе в этом случае будет единичный уровень.

Ток между стоком и истоком появляется, если входной сигнал превышает некоторый пороговый уровень (напряжение образования канала). Чем больше это превышение, тем меньше (до определенного предела) сопротивление канала $r_{си}$ между стоком и истоком, которое должно обеспечивать низкий уровень выходного сигнала, близкий к нулевому, то есть $r_{си} \ll R_C$. Поэтому для определения нулевого уровня приходится привлекать выходные характеристики транзистора и эквивалентную схему ключа в этом режиме.

2.6.3. Схемы замещения полевого транзистора

Непроводящий транзистор (при $U_{вх} = U^0$) обычно представляют в виде источника очень малого (наноамперного) тока утечки $I_{ут}$ (рис. 2.35а), тогда как проводящий транзистор (при $U_{вх} = U'$) заменяют его внутренним сопротивлением канала $r_{си}$ (рис. 2.35б), зависящим от величины $U_{вх}$.

С помощью эквивалентных схем можно найти уровни выходного сигнала U^0 и U' :

$$U_{Вых}^1 = E - I_{ут} R_C \quad (2.22)$$

$$U_{Вых}^0 = E r_{си} / (r_{си} + R_C). \quad (2.23)$$

Значение тока $I_{ут}$ обычно приводится в справочниках, что позволяет быстро оценить $U_{Вых}^1$. Однако внутреннее сопротивление транзистора $r_{си}$ сильно зависит от режима, поэтому для оценки напряжения $U_{Вых}^0$ приходится привлекать выходные характеристики транзистора, приведенные на рис. 2.36.

Чтобы дополнительно учесть наличие источника питания и резистора в соответствии со схемой рис. 2.35б, воспользуемся уравнением, подобным (1.26), записав выходной сигнал в виде разности:

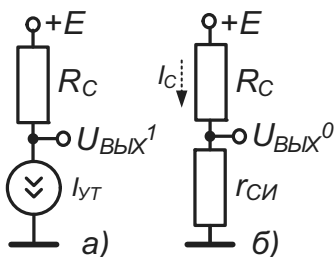


Рис. 2.35

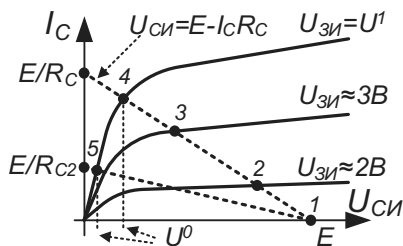


Рис. 2.36

$$U_{си} = E - I_c R_c \quad (2.24)$$

Это уравнение отображается на графике прямой линией из точки «1» в точку «4».

Таким образом, подавая входной сигнал, мы тем самым выбираем одну из выходных характеристик семейства, которая будет определять значение тока стока I_c . Тогда в соответствии с выражением (2.24) напряжение $U_{си}$ будет находиться в точке пересечения выходной характеристики, дающей значение тока I_c , с указанной прямой, определяющей выходной сигнал.

Например, если входной сигнал будет меньше порогового напряжения образования канала, то ток через транзистор будет отсутствовать, то есть состояние ключа будет соответствовать точке «1» ($U_{си} = E$). При увеличении входного сигнала сверх порогового будет происходить перемещение рабочего режима в точки «2» и «3», а выходное напряжение будет уменьшаться.

Но максимум входного сигнала ограничен единичным значением U' . Поэтому точка «4», соответствующая этому напряжению, будет характеризовать минимальный уровень выходного напряжения, который должен быть меньше стандартного нулевого U^0 .

Поскольку входной сигнал увеличивать сверх значения U' нельзя, то, как видно из (2.23) можно уменьшить напряжение $U_{вых}^0$ за счет увеличения сопротивления нагрузки, что показано на рис. 2.36 прямой линией из точки «1» в точку «5» для $R_{c2} > R_c$. В этом случае выходной сигнал будет соответствовать меньшему напряжению U^0 в точке «5».

С учетом изложенного, величина R_c на практике получается весьма большой (десятки и сотни кОм). Очевидно, что быстродействие такой схемы будет невелико (см. п. 1.2.1), поскольку паразитная емкость нагрузки будет медленно заряжаться через высокоомный резистор R_c .

Другим существенным недостатком является то, что такое сопротивление занимает на кристалле площадь около 10 транзисторов. Поэтому для уменьшения физических размеров ключа была предложена схема, где вместо полупроводникового резистора R_c используется имитатор резистора – дополнительный транзистор в качестве нагрузки. Проводящий канал транзистора управляется напряжением и может иметь очень малое сечение, то есть нетрудно получить требуемое большое сопротивление. А размеры транзистора гораздо меньше размеров высокоомного полупроводникового резистора.

2.6.4. Ключ на полевом транзисторе с транзисторной нагрузкой

В этой схеме (рис. 2.37) используются два транзистора, причём ключевым транзистором является только транзистор T_1 , а транзистор T_0 выполняет роль электронной нагрузки. Он имеет специальную конструкцию, то есть его геометрия выбирается таким образом, чтобы его ток был приблизительно равен току через резистор R_c в предыдущей схеме.

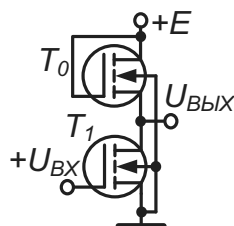


Рис. 2.37

Поэтому такой ключ допускает более плотную упаковку на кристалле, что особенно важно при изготовлении БИС. Также удобно на практике то, что транзисторы T_0 и T_1 имеют одинаковый тип канала и выполняются на одной подложке в едином технологическом процессе. До сих пор на основе подобных ключей выпускаются дешевые калькуляторные БИС.

Однако быстродействие ключа с транзисторной нагрузкой по-прежнему осталось невысоким.

2.6.5. Ключ на разнотипных МОП транзисторах

В таком, технологически более сложном ключе (рис. 2.38) комплементарные (взаимодополняющие) транзисторы разного типа проводимости выполнены на разных подложках. Причем оба транзистора T_1 и T_2 являются ключевыми, но работающими в противофазе. Когда один транзистор проводит, другой – не проводит. Это обеспечивается благодаря разному типу каналов транзисторов, хотя управляются они от единого входного сигнала. Обратите внимание на то, что сигналы на затворах отсчитываются относительно разных истоков.

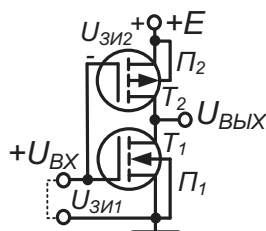


Рис. 2.38

Пусть входное напряжение близко к нулю (пунктирная линия на схеме). Напряжение на затворе $U_{з1}$ будет меньше порогового и транзистор T_1 проводить не будет.

Затвор транзистора T_2 в этом случае будет подключен к общей точке (полярность приведена на рисунке). Поэтому между затвором и истоком транзистора T_2 будет действовать напряжение питания, большее порогового, обеспечивая проводящее состояние транзисто-

ра. Выходное напряжение при этом будет иметь высокий единичный уровень.

Пусть входное напряжение будет единичным и для простоты равным напряжению питания E . Очевидно, что при этом транзистор T_1 будет проводить. В это же время потенциалы затвора и истока транзистора T_2 будут почти одинаковыми, то есть $U_{зи2} \approx 0$. Вследствие этого транзистор T_2 находится в непроводящем состоянии, а выходное напряжение $U_{Вых}$ благодаря проводящему транзистору T_1 будет иметь низкий нулевой уровень.

Главными достоинствами ключа на комплементарных транзисторах являются высокое быстродействие и очень малый ток, потребляемый от источника питания в статическом состоянии.

Высокое быстродействие обеспечивается быстрым зарядом и разрядом паразитной ёмкости на выходе (см. п. 1.2.1) через малые внутренние сопротивления проводящих транзисторов T_2 и T_1 . В схеме ключей с активным сопротивлением или электронной нагрузкой (см. п. 2.6.3 и 2.6.4) заряд происходит значительно медленнее через высокоомное сопротивление R_c или транзистор T_0 с незначительным эквивалентным током.

Чрезвычайно малый потребляемый ток объясняется двумя основными причинами. Во-первых, нагружается комплементарный ключ практически нулевым током изолированных затворов следующего каскада. А, во-вторых, противофазной работой транзисторов, когда один из транзисторов схемы не проводит, а значит и ток от источника питания отсутствует и в нулевом, и в единичном состоянии.

Однако, отметим, что дополнительный ток может потребляться на заряд паразитных емкостей (см. далее).

2.6.6. Быстродействие ключей на полевых транзисторах

определяется процессами перезаряда внутренних межэлектродных паразитных емкостей транзистора [28]. Чем меньше размеры транзистора, тем меньше паразитные емкости и проще их быстро перезарядить. Поэтому переход при изготовлении БИС на технологический процесс с большим разрешением фотошаблонов позволяет реализовать миниатюрные ключевые схемы и, как следствие, существенно повышаются рабочие частоты.

Кроме того, нужно помнить, что скорость обработки сигналов неразрывно связана с потребляемой энергией. При переключении

от источника питания всегда потребляется ток на заряд паразитных емкостей. Энергия заряда паразитной емкости не возвращается в источник питания, так как в другом такте емкость разряжается через ключевой транзистор T_r . Поэтому энергия заряда превращается в тепловую, рассеивается на транзисторе, приводя к нагреву кристалла.

Чем выше частота переключения, тем больше средний ток на перезаряд емкостей, тем больше потребляемая мощность и сильнее греется устройство.

§2.7. Логика на полевых транзисторах

2.7.1. Общие сведения

В зависимости от типа канала используемых полевых транзисторов различают три основные технологии реализации логических элементов – *p*МОП, *n*МОП и *к*МОП.

- *p*МОП элементы имеют отрицательное напряжение питания и применяются до сих пор в автономных устройствах типа калькуляторов.
- *n*МОП технология, характеризующаяся положительным напряжением питания, совместимым по полярности с ТТЛ элементами, использовалась в первых микропроцессорных БИС.
- *к*МОП логические элементы, наиболее массовые в настоящее время, имеют также положительное напряжение питания.

Благодаря высокому входному сопротивлению МОП транзистора облегчается управление ключами и появляется возможность использования релейного принципа. Например, функцию «ИЛИ» (рис. 2.39а) можно реализовать при помощи параллельного соединения ключей (ток течет, когда проводят **или** ключ K_1 , **или** K_2), а функцию «И» (рис. 2.39б) – последовательного (ток течет, когда проводят **и** ключ K_1 , **и** K_2).

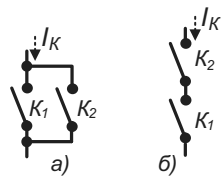


Рис. 2.39

Поэтому логические элементы на полевых транзисторах, имеющие очень простые и практически *безрезисторные* схемы, занимают на кристалле в несколько раз меньшую площадь, чем ТТЛ элемент, что позволяет реализовать достаточно сложные системы в виде БИС.

2.7.2. МОПТЛ элементы на однородных транзисторах

Для примера на рис. 2.40а приведена схема, реализующая функцию «И-НЕ», в которой транзистор T_0 выполняет роль электронной нагрузки, а ключевые транзисторы T_1 и T_2 соединены последовательно.

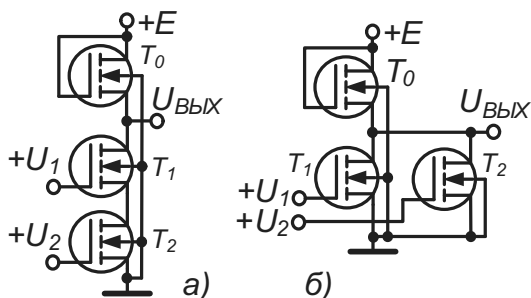


Рис. 2.40

Если на одном из входов элемента будет «0», то соответствующий ключевой транзистор не будет проводить, поэтому последовательная цепочка будет разорвана и выходной сигнал будет единичным.

Если на всех входах будут единичные сигналы, то все транзисторы последовательной цепочки будут проводить и выходной сигнал будет равен нулю. Таким образом, данный элемент реализует функцию «И-НЕ».

Соединив ключевые транзисторы T_1 и T_2 параллельно, можно получить элемент «ИЛИ-НЕ» (рис. 2.40б).

Таким образом, соединяя транзисторы последовательно или параллельно легко получить и более сложные конфигурации типа «И-ИЛИ-НЕ», которые занимают на кристалле несравненно меньше места, чем ТТЛ устройства.

Логические элементы на однородных транзисторах характеризуются сравнительно большим временем задержки (доли мкс), так как заряд паразитной ёмкости происходит малым током транзистора T_0 и, кроме того, при большом числе элементов потребляемая мощность становится заметной.

На основе nМОП технологии в свое время производились микропроцессорные и интерфейсные БИС. В настоящее время чаще всего используют более сложную кМОП технологию с двумя подложками.

2.7.3. Базовая схема кМОПТЛ элемента «И-НЕ»

для случая двух входов выполнена на основе двух пар разнотипных транзисторов (рис. 2.41). Причем для реализации функции «И» нижние транзисторы комплементарных пар (T_1 и T_3) соединены, как и ранее (см. п. 2.3.2), последовательно друг с другом. Верхние транзисторы каждой пары (T_2 и T_4), управляемые в противофазе, согласно логическому принципу «дуальности» [23] должны быть включены параллельно друг другу. Увеличивая число пар можно увеличить число входов.

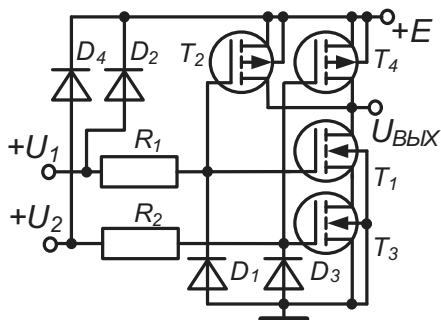


Рис. 2.41

Кроме ключевых транзисторов схема имеет на входах встроенные диодные ограничители, причем диоды D_1 и D_3 ограничивают сигнал снизу на небольшом отрицательном уровне (см. п. 1.5.2), а диоды D_2 и D_4 – сверху на уровне, близком к напряжению питания (см. п. 1.5.3). Небольшие внутренние резисторы R_1 и R_2 (около 1 кОм) для ограничения тока имеются только у нижних диодов. Верхние диоды резисторов не имеют, чтобы уменьшить площадь элемента. Поэтому в критических случаях рекомендуется предусматривать для этой цели внешние резисторы.

Кроме того, эти диоды, в известной степени, защищают полевые транзисторы от статического электричества, обеспечивая стекание отрицательного статического заряда на землю, а положительного – на источник питания.

Рассмотрим работу схемы. Пусть на одном из входов будет нулевой сигнал (например, $U_1 = 0$), тогда соответствующий нижний транзистор комплементарной пары (T_1) будет находиться в непроводящем состоянии, обеспечивая разрыв последовательной цепочки. В это время верхний транзистор (T_2) будет проводить, формируя единичный выходной сигнал.

Если все входные сигналы будут единичными, то все нечетные транзисторы последовательной цепочки будут проводить, а все четные, включенные параллельно, будут находиться в непроводящем состоянии. Выходной сигнал при этом будет нулевым, то есть данная схема реализует функцию «И-НЕ».

Достоинства такой базовой структуры такие же, как у комплементарного ключа – малая потребляемая мощность в статике и высокое быстродействие.

Изменяя порядок включения комплементарных ключей, нетрудно самостоятельно реализовать схему «ИЛИ-НЕ», в которой нижние транзисторы комплементарных пар, реализующие функцию «ИЛИ», должны быть включены параллельно, а верхние – последовательно.

2.7.4. Характеристики кМОПТЛ элемента

такие же, как у ТТЛ элемента – входная, передаточная и две выходных.

Входная характеристика на рабочем участке является, по сути, характеристикой изолятора (рис. 2.42).

Обычно ее не рисуют, а приводят в справочнике ток утечки затвора (доли и единицы nA). Но на практике необходимо помнить о наличии ограничивающих диодов, которые делают входную характеристику на нерабочих участках резко нелинейной и похожей в определенном смысле на характеристику ТТЛ элемента (см. п. 2.4.2).

Увеличение входного тока при отрицательном напряжении объясняется отпиранием соответствующего нижнего диода (например, D_1), а при превышении напряжения питания – отпиранием верхнего диода (например, D_2). Таким образом, входная характеристика иллюстрирует двустороннее ограничение входного сигнала.

Пороговое напряжение, один из основных параметров элемента, из входной характеристики определить невозможно. Его находят из передаточной характеристики и зависит оно от напряжения питания:

$$U_{пор} = E / 2. \quad (2.25)$$

Например, если $E = 5 В$, то $U_{пор} = 2,5 В$, что больше $U_{мин}^1 = 2,4 В$ для ТТЛ схем. Отсюда следует несколько выводов.

Первый – вид передаточной характеристики существенно зависит от напряжения питания (рис. 2.43).

Второй – логические уровни кМОПТЛ элемента также определяются напря-



Рис. 2.42

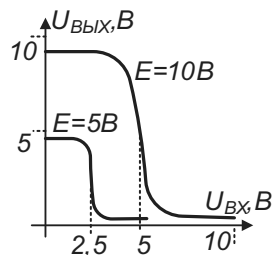


Рис. 2.43

жением питания и сильно отличаются от стандартных ТТЛ уровней, особенно единичный.

Третий – напрямую ТТЛ и КМОПТЛ элементы соединять нельзя. Поэтому для согласования МОПТЛ элементов с ТТЛ элементами используют специальные преобразователи уровней, которые есть в составе большинства серий (например, К561ПУ1 и ПУ2).

Отметим, что в настоящее время разработаны специальные серии КМОПТЛ, например, К1554 (аналог 74АС), с порогом равным 1,5 В для замены ТТЛ микросхем.

Выходные характеристики (их две – при нуле и при единице) по виду такие же, как у ТТЛ элемента (рис. 2.44), но значения максимальных выходных токов (единицы, а иногда десятки мА) зависят от типа микросхемы.

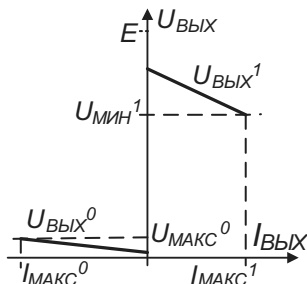


Рис. 2.44

Примечание: реальные устройства могут иметь дополнительно к базовой схеме рис. 2.41 буферные комплементарные ключи, обеспечивающие увеличенный ток нагрузки, в тех элементах, которые непосредственно подключены к выходному зажиму микросхемы.

Быстродействие КМОПТЛ зависит в основном от технологии, то есть разрешающей способности фотошаблона. Время задержки типовых серий примерно такое же, как у ТТЛ элементов (единицы нс). Рабочие частоты – десятки или даже сотни МГц.

Энергопотребление КМОПТЛ в статике (на низкой частоте) очень малое (несколько мкВт). На предельных рабочих частотах потребляемая мощность может возрасти до десятков мВт, что сравнимо с потреблением ТТЛ элемента.

3

ПОДКЛЮЧЕНИЕ НАГРУЗКИ К ВЫХОДУ МИКРОСХЕМ

Внешние объекты, подключаемые в виде нагрузки к выходам микросхем, могут иметь различные характеристики, которые необходимо принимать во внимание. В дальнейшем будем считать, что нагрузка представляет активное сопротивление, величина которого, как и напряжение, известны.

§3.1. Подключение нагрузки без преобразования уровня сигнала

3.1.1. Нагрузка в виде входов логических элементов

При стандартном использовании элементов к выходу обычно подключаются входы других *однотипных* логических элементов (рис. 3.1). Количество подключаемых входов ограничено так называемым коэффициентом разветвления (см. п. 2.4.7).

Если вместо этого коэффициента в справочнике приведены максимальные значения выходного тока $I_{\text{МАКС}}^1$ и $I_{\text{МАКС}}^0$, то необходимо, чтобы сумма входных токов (при нуле или единице) была меньше соответствующего максимального тока:

$$\sum I_{\text{ВХ}} < I_{\text{МАКС}} \quad (3.1)$$

Другим критерием, ограничивающим количество подключенных входов, является вносимая ими емкостная нагрузка, которая снижает быстродействие элемента. В справочниках обычно приводится максимальная емкость нагрузки $C_{\text{МАКС}}$, при которой гарантируется паспортное значение рабочей частоты элемента. Поэтому необходимо, чтобы и емкостная нагрузка не была чрезмерной:

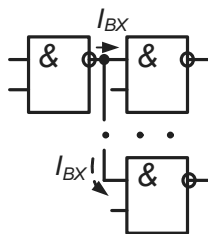


Рис. 3.1

$$\Sigma C_{BX} < C_{\text{МАКС}} \quad (3.2)$$

При соединении *разнотипных* логических элементов в первую очередь обращают внимание на уровни $U_{\text{МАКС}}^0$, $U_{\text{МИН}}^1$ и $U_{\text{ПОР}}$. Если эти уровни обеспечивают правильную работу нагружающих логических элементов, то есть находятся на рабочих участках передаточной характеристики (см. п. 2.4.5), то по-прежнему проверяют выполнение токового (3.1) и емкостного (3.2) условий. Иначе необходимо дополнять схему элементом, согласующим логические уровни, используя одну из ключевых схем для преобразования величины выходного сигнала (см. §3.2).

3.1.2. Нагрузка в виде светодиода

На практике используют две схемы включения светодиода для индикации логического состояния.

В *первой* из них (рис. 3.2) светодиод светится при наличии единичного сигнала $U_{\text{МС}}$ на выходе элемента. Поэтому при выборе сопротивления $R_{\text{ОГР}}$ нужно, чтобы оно обеспечивало протекание номинального тока $I_{\text{СД}}$:

$$R_{\text{ОГР}} \approx (U_{\text{МИН}}^1 - U_{\text{СД}}) / I_{\text{СД}} \quad (3.3)$$

Заметим, что значение $I_{\text{СД}}$ не должно превышать максимальное значение выходного тока микросхемы.

Здесь следует обратить внимание на то, что прямое напряжение светодиода $U_{\text{СД}}$ имеет повышенное по сравнению с обычным диодом значение, сильно зависит от тока и характеризуется существенным технологическим разбросом. Поэтому на практике величина $U_{\text{СД}}$ ($\approx 1,5 \dots 2,5 \text{ В}$) оказывается сравнимым со значением $U_{\text{МИН}}^1$ ($\approx 2,4 \text{ В}$), что приводит к большой неопределенности при выборе $R_{\text{ОГР}}$.

В этом смысле более предпочтительной является схема (рис. 3.3), в которой светодиод светится при наличии нулевого сигнала на выходе, то есть когда $U_{\text{МС}} = U^0$. При этом ток в цепи будет определяться в основном напряжением питания, большим, чем неизвестное напряжение $U_{\text{СД}}$. Поэтому уравнение для выбора сопротивления $R_{\text{ОГР}}$ становится более пригодным для практической оценки:

$$R_{\text{ОГР}} \approx (E - U_{\text{МАКС}}^0 - U_{\text{СД}}) / I_{\text{СД}} \quad (3.4)$$

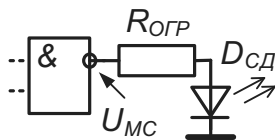


Рис. 3.2

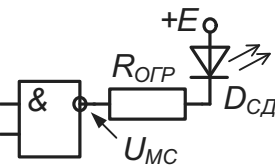


Рис. 3.3

Кроме того отметим, что для ТТЛ элементов максимальный выходной ток при нуле $I_{\text{МАКС}}^0$ на порядок больше значения $I_{\text{МАКС}}^1$, что позволяет в такой схеме при необходимости увеличить яркость индикатора. Для примера предположим, что $E = 5 \text{ В}$, $U_{\text{МАКС}}^0 = 0,4 \text{ В}$, $U_{\text{СД}} \approx 2 \text{ В}$, $I_{\text{МАКС}}^0 = 16 \text{ мА}$, тогда $R_{\text{ОГР}} \geq (5 - 0,4 - 2) / 16 \cdot 10^{-3} \approx 160 \text{ Ом}$. Реально величину $R_{\text{ОГР}}$ выбирают больше, например, примерно около $0,5 \text{ кОм}$, что для большинства светодиодов обеспечивает достаточную яркость.

Отметим также, что в этой схеме можно использовать элемент с открытым коллектором (см. п. 2.3.6), который позволяет использовать повышенное напряжение питания индицирующей цепочки, что бывает полезно при последовательном включении нескольких светодиодов.

Кстати элемент с открытым электродом можно использовать для индикации единичного состояния, включая диод параллельно выходу (рис. 3.4). В этом случае светодиод при нулевом напряжении будет закорочен, а при единичном, когда внутренний ключ элемента не проводит, наоборот будет светиться. Ограничивающее сопротивление (при $I_{\text{СД}} < I_{\text{МАКС}}^0$) рассчитывают по формуле, похожей на (3.4):

$$R_{\text{ОГР}} \approx (E - U_{\text{МАКС}}^0) / I_{\text{СД}} \quad (3.5)$$

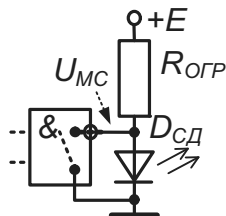


Рис. 3.4

3.1.3. Параллельная работа элементов

Соединять выходы между собой не рекомендуется (см. п. 2.3.5) из-за возможного конфликта сигналов (один элемент в единице, а другой в нуле). Но для увеличения максимального выходного тока можно при определенных условиях соединять входы и выходы элементов параллельно. Конкуренции логических сигналов на выходе при этом не будет.

Такое соединение возможно для элементов одной микросхемы (рис. 3.5), в которых выходные уровни U^1 и U^0 практически одинаковые. Максимальный выходной ток при этом будет больше в четыре раза.

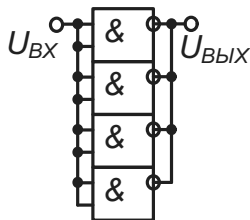


Рис. 3.5

Если использовать элементы из разных микросхем, то могут возникнуть неконтролируемое и нелинейное перераспределение энергии между источниками напряжения (с малым выходным со-

противлением) из-за того, что один из уровней напряжения больше. Например, выходной каскад элемента с меньшим уровнем сигнала может из источника энергии превратиться в ее потребителя.

§3.2. Однокаскадные ключи для преобразования уровня выходного сигнала

В некоторых случаях передача данных происходит с преобразованием электрического уровня информационного сигнала, для чего используются разнообразные ключевые схемы, чаще всего на основе биполярных транзисторов, которые отличаются дешевизной и устойчивостью к воздействию внешней среды.

3.2.1. Использование элемента с открытым электродом

позволяет обойтись (см. п. 2.3.6) для преобразования уровня только одним внешним резистором (рис. 3.6), который подключен к другому источнику питания E_1 , напряжение которого будет определять единичный уровень выходного напряжения U_{MC} . Поэтому считают, что элемент с открытым электродом имеет в определенном смысле универсальный выход, поскольку разработчик может выбрать напряжение E_1 больше или меньше стандартного $E = +5$ В.

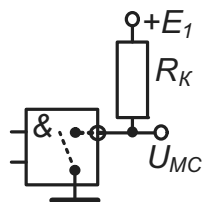


Рис. 3.6

При выборе сопротивления внешнего резистора R_K нужно принимать во внимание единичный ток нагружающий схему (см. п. 2.5.4) и влияние величины R_K на быстрдействие ключа (см. п. 1.2.1).

Элемент с открытым коллектором или стоком является сравнительно низковольтным, то есть имеет предельное напряжение не более 15–30 В. Поэтому при работе с большими напряжениями питания используют внешние ключевые устройства.

3.2.2. Внешний ключ на *п-р-п* транзисторе

для увеличения уровня сигнала (рис. 3.7) можно реализовать на основе ранее рассмотренного ключа на биполярном транзисторе (см. п. 1.7.1). Причем напряжение E_1 может быть любым по вели-

чине, лишь бы транзистор имел соответствующее допустимое напряжение.

При расчете ключа (см. п. 1.7.4) необходимо иметь в виду ограниченный выходной ток микросхемы. То есть базовый ток транзистора не может быть превышать $I_{\text{МАКС}}^1$, что в свою очередь в соответствии с токовым условием насыщения (1.27) ограничивает допустимый ток в коллекторной цепи и сопротивление R_K .

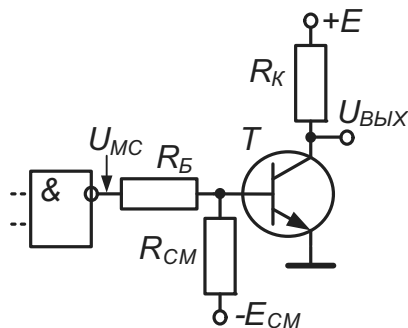


Рис. 3.7

Определенным недостатком схемы является наличие двух источников питания.

3.2.3. Внешний ключ без источника смещения

использует (рис. 3.8) вместо режима глубокой отсечки тока режим «заземленной» базы, который создается при помощи нелинейной характеристики дополнительного диода смещения D_{CM} . Подобный способ был применен при построении ДТЛ элемента (см. п. 2.1.3). Поэтому в данной схеме сумма напряжений отпирающего диода и эмиттерного перехода транзистора оказывается большей значения $U_{\text{МАКС}}^0$ (см. рис. 2.4), что обеспечивает непроводящее состояние транзистора.

Наличие диода во входной цепи наряду с полезным действием приводит и к некоторым нежелательным последствиям. Дело в том, что при оценке сопротивления R_B , которое осуществляется по формуле:

$$R_B = (U_{\text{МИН}}^1 - U_{\text{ДСМ}} - U_{\text{БН}}) / I_{\text{БН}} \quad (3.6),$$

существует трудность, похожая на рассмотренную в п. 3.1.2 и связанная с близостью значения $U_{\text{МИН}}^1$ ($\approx 2,4 \text{ В}$) и разбросом прямых напряжений на переходах $U_{\text{ДСМ}} + U_{\text{БН}}$ ($\approx 1,2 \dots 2 \text{ В}$). В связи с этим расчет требует осторожности и практического уточнения значения R_B .

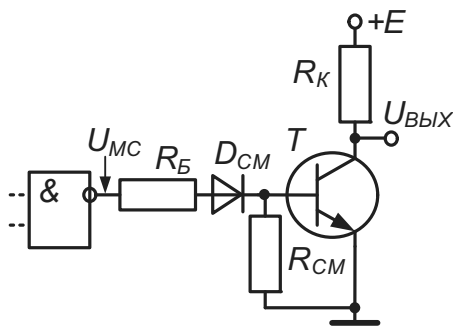


Рис. 3.8

Однако на практике учитывают, что при нулевом значении U_{MC} микросхема практически не нагружается и, как видно из нулевой выходной характеристики (см. п. 2.4.8), при нулевом токе $I_{ВЫХ}$ выходное напряжение $U_{ВЫХ}^0 (\approx 0,1 \text{ В})$ оказывается меньше напряжения отпирания $U_{ОТП} (\approx 0,3 \text{ В})$ транзистора. Поэтому в большинстве подобных ключевых схем диод и резистор смещения не нужны.

3.2.4. Внешний ключ на р-п-р транзисторе

имеет нагрузку, соединенную с общей точкой (рис. 3.9), что оказывается полезным в некоторых случаях. Однако следует отметить, что напряжение E_1 в этой схеме не должно быть меньше значения $U_{МИН}^1$, но больше $U_{МАКС}^0$, хотя бы на 1,5 В.

Проверим работу схемы для разных уровней сигнала. Пусть для простоты $U_{MC} = 0$, тогда нижний вывод резистора R_B будет подключен к общей точке и в базовой цепи транзистора появится показанный на рисунке ток $I_{БН}$. При правильном расчете

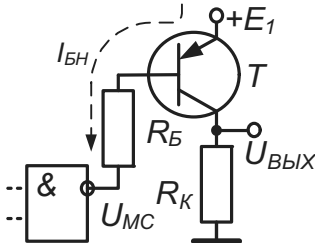


Рис. 3.9

$$R_B \leq (E_1 - U_{БН} - U_{МАКС}^0) / I_{БН} \quad (3.7),$$

где $I_{БН} \geq I_{КН} / V_{МИН} \approx E_1 / R_K \cdot V_{МИН}$, транзистор насыщен и выходное напряжение будет иметь высокий единичный уровень, близкий к E_1 .

Для единичного выходного сигнала микросхемы предположим, что $U_{MC} = U^1 \approx E_1$. Тогда база и эмиттер транзистора будут иметь примерно одинаковый потенциал, что означает нахождение транзистора в непроводящем состоянии и отсутствие напряжения на выходе.

Явный недостаток этой схемы, заключающийся в малом диапазоне выбора напряжения E_1 , обычно устраняется введением дополнительного ключевого каскада (см. п. 3.3.5).

§3.3. Двухкаскадные ключевые усилители

В некоторых случаях токи нагрузки составляют величину в несколько ампер и более. И поэтому базовый ток одиночного ключа получается тоже большим и превышающим предельный выходной ток микросхем, что требует введения дополнительного усилительного каскада.

3.3.1. Ключ на составном транзисторе

Простейшим способом уменьшения нагрузки на микросхему является использование составного транзистора (рис. 3.10). При подобном соединении транзисторов (двух и более), которое в переводной литературе называют схемой Дарлингтона, происходит увеличение общего коэффициента усиления за счет умножения коэффициентов усиления отдельных транзисторов. Выбирая количество транзисторов можно снизить нагрузку на выход логического элемента до необходимого уровня:

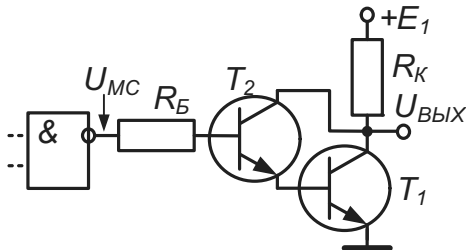


Рис. 3.10

$$I_{Б2} = I_{К1} / B_1 \cdot B_2 \leq I_{МАКС}^1. \quad (3.8)$$

Отметим также, что в данном случае, как и в схеме рис. 3.8, нет источника смещения, и роль диода $D_{см}$ выполняет эмиттерный переход транзистора T_2 .

Но ключ на составном транзисторе имеет и существенные недостатки.

Во-первых, расчет сопротивления R_B затруднен, как и в ключе без источника смещения (см. п. 3.2.3), в связи с неопределенностью и малой величиной разности $U_{МИН}^1$ и напряжений на эмиттерных переходах составного транзистора.

Во-вторых, выходной транзистор T_1 в проводящем состоянии не работает в режиме насыщения и имеет поэтому повышенное напряжение, что приводит к дополнительным потерям мощности на ключе. Поясним это, считая, что транзистор T_2 идеальный и в проводящем состоянии он обеспечивает соединение коллектора и базы транзистора T_1 , что означает нахождение последнего на границе режима насыщения. Реально напряжение на транзисторе T_2 не равно нулю, поэтому напряжение на проводящем транзисторе T_1 определяется суммой:

$$U_{КЭ1} = U_{КН2} + U_{БН1} \quad (3.9)$$

и оказывается больше примерно на 1 В, чем в режиме насыщения, что при больших токах нагрузки приводит к существенным потерям энергии, разогреву транзистора и необходимости увеличивать площадь теплоотводящего радиатора.

И, в *третьих*, в непроводящем состоянии транзистор T_1 работает в режиме с оборванной базой, поскольку транзистор T_2 не проводит и база T_1 оказывается никуда не подключенной.

3.3.2. Устранение недостатков составного транзистора

Первый недостаток, связанный с расчетом сопротивления R_B (см. п. 3.2.3), является «родовым» и в ключевых схемах с другой конфигурацией (см. далее) проявляется в меньшей степени.

Второй недостаток устраняется тем, что коллектор транзистора T_2 подключается (рис. 3.11) к дополнительному сопротивлению R_{K2} :

$$R_{K2} \approx E / I_{BH1} \leq E \cdot B_1 / I_{KH1} \quad (3.10),$$

которое задает базовый ток I_{BH1} , необходимый для насыщения выходного транзистора T_1 .

Режим с оборванной базой исключается введением сопротивления R_{CM} , которое создает путь для остаточных токов транзисторов (см. п. 1.6.4).

Отметим, что рассмотренные изменения схемы ключа не применимы, если транзисторы составного выполнены в едином корпусе.

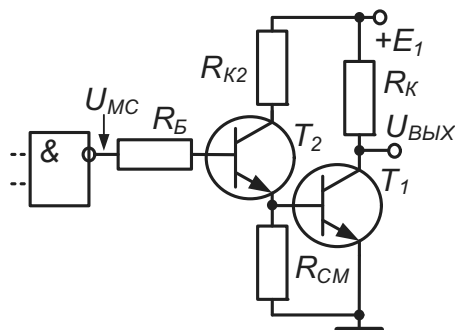


Рис. 3.11

3.3.3. Двухкаскадный ключ

реализует другой способ соединения транзисторов для увеличения коэффициента усиления, когда база выходного транзистора соединена не с эмиттером предыдущего, как в составном, а с коллектором (рис. 3.12).

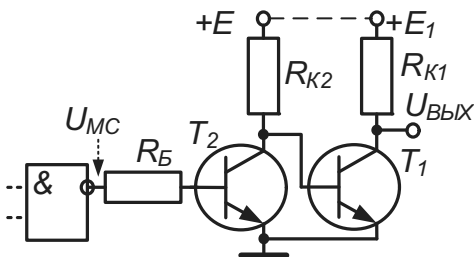


Рис. 3.12

В этой схеме насыщение выходного транзистора обеспечивается благодаря правильному выбору сопротивления R_{K2} и отсутствует режим с оборванной базой, поскольку, когда транзистор T_2 проводит, то база T_1 соединяется с его эмиттером, обеспечивая режим с «заземленной» базой.

Таким образом, схема лишена двух основных недостатков составного транзистора и вдобавок транзистор T_2 можно выбрать низковольтным, потому что напряжение на его коллекторе определяется прямым напряжением $U_{БН1} \approx 1 \text{ В}$.

Нужно заметить, что сигнал от микросхемы дважды инвертируется, то есть когда $U_{MC} = 0$, транзистор T_2 не проводит, а транзистор T_1 проводит, обеспечивая $U_{ВЫХ} = 0$. Если $U_{MC} = 1$, то состояния транзисторов меняются на противоположные и $U_{ВЫХ} = 1$.

Чтобы исключить трудности с расчетом сопротивления R_B (см. п. 4.2.3) в этой схеме, можно применить элемент с открытым электродом и исключить транзистор T_2 и его элементы (рис. 3.13). Но в этом случае базовый ток $I_{БН1} \approx E / R_{K2}$ должен быть меньше выходного тока $I_{МАКС}^0$.

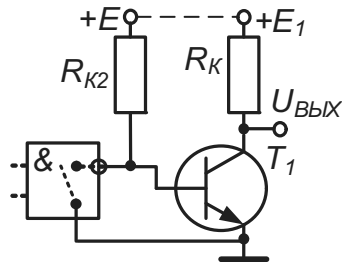


Рис. 3.13

3.3.4. Защита мощных выходных транзисторов

Иногда при эксплуатации низкоомная нагрузка может быть случайно закорочена, что приводит к появлению очень большого тока выходного транзистора, и он выходит из строя.

Поэтому мощный транзистор нуждается в защите от перегрузки. Рассмотрим простейшую схему ограничителя тока, приведенную на рис. 3.14, на котором выделен только выходной транзистор T_1 и транзистор защиты $T_{ЗЩ}$.

Резистор $R_{ДТ}$ выполняет функции датчика тока, потому что напряжение на нем

$$U_R \approx I_{Э1} \cdot R_{ДТ} \quad (3.11)$$

несет информацию о токе нагрузки, так как $I_{Э1} \approx I_{К1}$. Принцип действия защиты основан на нелинейности входной характеристики

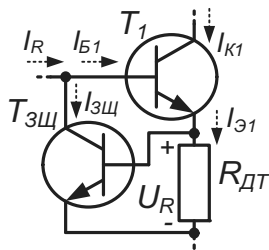


Рис. 3.14

транзистора. В рабочем режиме напряжение U_R должно быть меньше напряжения отпирания $U_{отп}$ (см. п. 1.6.4) транзистора $T_{зщ}$, что обеспечивается соответствующим выбором сопротивления:

$$R_{дт} \leq U_{отп} / I_{ном}, \quad (3.12)$$

где $I_{ном}$ – номинальный ток нагрузки.

Поэтому в рабочем режиме транзистор $T_{зщ}$ будет находиться в непроводящем состоянии, токи его будут отсутствовать (то есть $I_{зщ} = 0$), и схема защиты не будет мешать работе транзистора T_1 , поскольку

$$I_{Б1} = I_R - I_{зщ} \quad (3.13)$$

При перегрузке ток $I_{Э1}$ увеличивается, напряжение на резисторе защиты становится больше $U_{отп}$ и появляется базовый и, главное, усиленный коллекторный ток транзистора защиты $I_{зщ}$.

Из формулы (3.13) видно, что при этом базовый ток выходного транзистора будет уменьшаться и схема перейдет в режим стабилизации тока нагрузки, защищая транзистор от превышения допустимого тока.

Следует отметить, что подобная защита только ограничивает ток, тогда как более сложные схемы обеспечивают полное выключение выходного транзистора до устранения перегрузки. В настоящее время некоторые мощные ключи имеют встроенную схему защиты.

Но в любом случае необходимо оценивать мощность, которая выделяется на транзисторе в проводящем и непроводящем состоянии, умножая остаточные ток и напряжение. При работе на высоких частотах к статическим потерям добавляются динамические [1]. Поэтому мощные ключи, как правило, устанавливают на теплоотводящем радиаторе.

3.3.5. Ключ с двуполярным выходным напряжением

может использоваться (рис. 3.15), например, в передатчике последовательного порта микропроцессорной системы, выполняя преобразование сигнала ТТЛ уровня в двуполярный уровень связанного стандарта RS232 [6]. Схема имеет выходной $p-n-p$ транзистор T_1 подобно ключу, рассмотренному ранее в п. 3.2.4. Но

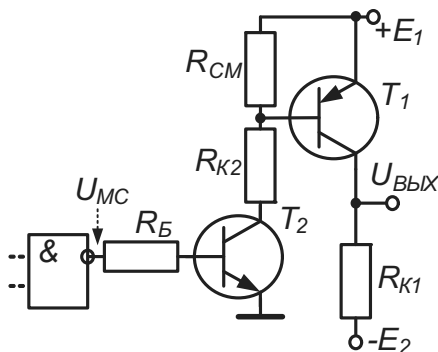


Рис. 3.15

уровни напряжения питания E_1 и E_2 здесь не связаны с управляющим сигналом и могут быть выбраны в зависимости от надобности и предельного напряжения имеющихся транзисторов.

Первый ключевой каскад выполнен на транзисторе T_2 по обычной схеме (см. п. 3.2.3) без источника смещения.

Пусть $U_{MC} = U_1$, тогда транзистор T_2 проводит и большая часть его коллекторного тока потечет в базу транзистора T_1 . При правильном расчете R_{K2} последний будет насыщен и подключит выходной зажим к положительному источнику, то есть $U_{ВЫХ} = +E_1$.

Когда $U_{MC} = 0$, транзистор T_2 не проводит, поэтому базовый ток транзистора T_1 близок к нулю (сопротивление R_{CM} исключает режим оборванной базы). Это означает, что транзистор T_1 эквивалентен разрыву и выходное напряжение в этом случае будет определяться (через R_{K1}) отрицательным источником, то есть $U_{ВЫХ} = -E_2$.

В этой схеме также можно заменить транзистор T_2 и его резистор R_B элементом с открытым коллектором, как на рис. 3.13. Но напряжение E_1 не должно быть при этом больше предельного для элемента с открытым электродом, а базовый ток $I_{B1} \approx E / R_{K2}$ должен быть меньше выходного тока $I_{МАКС}^0$.

3.3.6. Микросхемы с наборами мощных ключей

упрощают разработку и позволяют упростить внешнее обрамление процессоров, поскольку в одном корпусе бывает выполнено несколько ключей вместе со схемами управления.

Примером подобного ключевого модуля является мостовой драйвер, структура которого приведена рис. 3.16 и состоит из 4 мощных ключей, образующих два каскада, внешне похожих на выходную цепь ТТЛ элемента (см. п. 2.2.2). Каждый из каскадов имеет внутреннюю

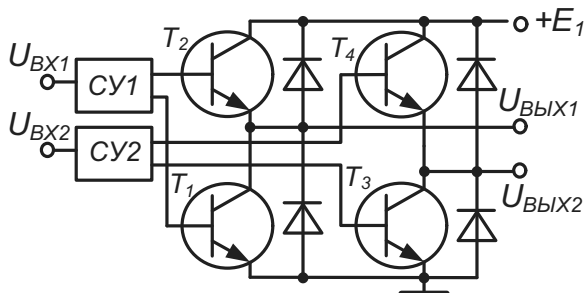


Рис. 3.16

схему управления транзисторами, которая не только обеспечивает надежное противофазное включение и выключение транзисторов, но и устраняет возникающие при переключении «сквозные» токи (см. п. 2.2.3).

Кроме того такие микросхемы имеют, как правило, встроенную схему защиты от перегрузок и диоды, включенные параллельно транзисторам и пропускающие обратный ток, который может возникнуть при индуктивной нагрузке.

Управляющие сигналы могут несколько отличаться от стандартного ТТЛ уровня, например, повышенными значениями $U_{мин}^1 \approx 3,5 \text{ В}$ и $U_{макс}^0 \approx 0,8 \text{ В}$.

Мостовые схемы позволяют организовать двуполярное перемагничивание импульсных трансформаторов (см. п. 4.5.1) или обмоток электродвигателей при работе от одного источника питания.

§3.4. Ключи на мощных полевых транзисторах

3.4.1. Ключ на полевом транзисторе с низковольтным управлением

имеет минимум внешних элементов (рис. 3.17), поскольку затвор может соединяться с выходом микросхемы напрямую. Но определенные особенности полевого транзистора приходится учитывать при выборе его типа.

Дело в том, что полевой транзистор не имеет формального условия перехода в проводящее состояние, как биполярный транзистор. И нулевой уровень выходного напряжения на стоке транзистора зависит при заданной нагрузке R_c от внутреннего сопротивления $r_{си}$ (см. п. 2.6.3), то есть в конечном счете от единичного значения напряжения на затворе $U_{мс}$, которое для худшего случая следует считать равным $U_{мин}^1$.

Поэтому при выборе типа полевого транзистора сначала из выражения (2.23) находят требуемое внутреннее сопротивление:

$$r_{си} \leq R_c U_{Вых}^0 / (E - U_{Вых}^0) \quad (3.14)$$

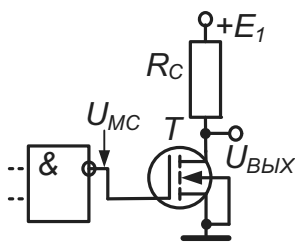


Рис. 3.17

и затем подбирают полевой транзистор, имеющий при напряжении на затворе $U_{\text{мин}}^1$, требуемую величину $r_{\text{си}}$.

Если транзистор с нужными параметрами подобрать не удастся, то можно после микросхемы поставить преобразователь уровня сигнала, то есть перейти к двухкаскадной схеме (см. далее).

3.4.2. Ключ на полевом транзисторе с преобразованием управляющего уровня

МОП транзистор имеет в большинстве случаев повышенное управляющее напряжение и нуждается поэтому (по причинам указанным в п. 3.4.1) в дополнительном преобразователе уровня сигнала, как и мощный биполярный транзистор.

В качестве такого преобразователя можно применить, например, один из ранее рассмотренных однокаскадных ключей. А при возможности обычно ограничиваются использованием элемента с открытым электродом (рис. 3.18), выбирая напряжение питания E , чтобы обеспечить требуемое внутреннее сопротивление $r_{\text{си}}$ (см. п. 3.4.1).

Вместе с тем отметим, что в настоящее время разработаны мощные полевые транзисторы с низковольтным управляющим напряжением около 2..4 В, что позволяет исключить промежуточный усилительный каскад.

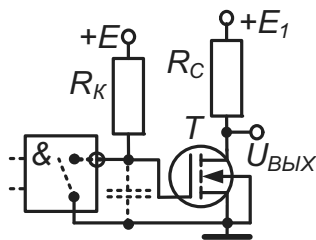


Рис. 3.18

Не вдаваясь в подробности, которые выходят за рамки пособия, необходимо отметить, что многие мощные полевые транзисторы имеют выполненный на этой же подложке диод между стоком и истоком, пропускающий ток в обратном направлении, что бывает полезно при индуктивной нагрузке. Поэтому полупроводниковая структура становится достаточно сложной, что приводит к образованию паразитного тиристора (см. п. 4.2.1), который в нормальном режиме не мешает работе, но может активироваться за счет проникновения управляющего сигнала через межэлектродные емкости. Вследствие этого большие скорости переключения не рекомендуются при управлении затворами мощных полевых транзисторов. И для исключения эффекта так называемой «защелки» увеличивают постоянную времени входной цепи, например, за счет дополнительного конденсатора (на рисунке показан пунктиром). В ряде случаев для управления мощными полевыми транзисторами используют специализированные микросхемы (драйверы), учитывающие этот эффект.

Кроме диода на кристалле с мощным полевым транзистором могут быть размещены встроенные датчики температуры и косвенного измерения тока нагрузки. Подобные транзисторы иногда даже называют (преувеличенно) «интеллектуальными» ключами.

Отметим также, мощные полевые транзисторы (в переводной литературе иногда называемые *MOSFET – Metal Oxide Semiconductor Field Effect Transistor*) превосходят биполярные транзисторы по быстродействию, но являются сравнительно низковольтными. Для высоковольтных применений были разработаны комбинированные полупроводниковые устройства, сочетающие свойства биполярного и полевого транзистора.

3.4.3. Биполярный транзистор с изолированным затвором,

называемый сокращенно БТИЗ или *IGBT (Insulated Gate Bipolar Transistor)*, представляет сочетание полевого транзистора на входе и биполярного на выходе. Одна из возможных структур такого своеобразного составного транзистора в упрощенном виде приведена на рис. 3.19а, а УГО на рис. 3.19б.

Высокое входное сопротивление полевого транзистора T_n позволяет получить малую мощность в цепи управления, а биполярный транзистор T_b позволяет обеспечить малое падение напряжения в проводящем состоянии и достаточно высокое предельное напряжение в непереводящем.

Поэтому в настоящее время подобные комбинированные ключи находят все большее применение в сверхмощных устройствах и наряду с мощными полевыми транзисторами считаются наиболее перспективными для новых разработок.

Уровни управляющих напряжений для БТИЗ значительно превышают ТТЛ сигнал и поэтому в комплекте с такими гибридными ключами обычно используют достаточно сложные схемы формирования входного сигнала (драйверы).

Отметим также, что внутри подобных полупроводниковых модулей имеются паразитные тиристорные структуры и поэтому могут возникнуть связанные с ними нежелательные процессы при переключении. В связи с этим необходимо строго выполнять все требования технической документации.

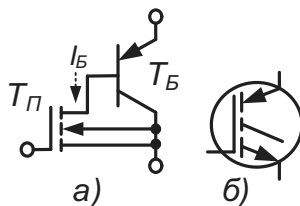


Рис. 3.19

4

КЛЮЧЕВЫЕ УСТРОЙСТВА С ГАЛЬВАНИЧЕСКИМ РАЗДЕЛЕНИЕМ ВХОДА И ВЫХОДА

В ряде практических случаев, например, при изолировании прибора от опасного напряжения промышленной сети, необходимо передача цифровой информации без электрического соединения входа и выхода, для чего используется оптическая или магнитная связь.

§4.1. Ключевые устройства с оптической связью

4.1.1. Транзисторный оптрон,

состоящий из светодиода и фототранзистора, широко используется в цифровых, а иногда и в аналоговых цепях. Причем информационный сигнал от фототранзистора, управляемого светом, можно снимать, как с коллектора (рис. 4.1), так и с эмиттера (рис. 4.2). Управление светодиодом входной цепи было пояснено ранее (см. п. 3.1.2). В данном случае для примера используется элемент с открытым электродом и резистор $R_{огр}$.

Обратить внимание на то, что источники питания входной и выходной цепей разные, не имеют между собой связи и отсчитываются относительно разных общих точек, помеченных, как

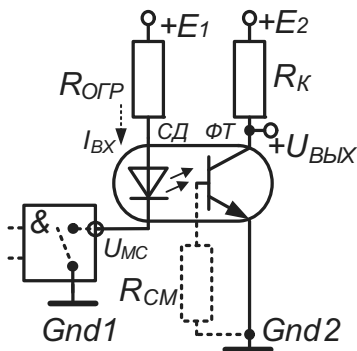


Рис. 4.1

Gnd1 и Gnd2. Таким образом, электрическая связь между входом и выходом отсутствует, а информация передается световым потоком.

Пусть входной ток оптрона $I_{ВХ}$ равен нулю ($U_{МС} \approx E$), то есть светодиод не излучает. Поэтому фототранзистор не освещен, находится в непроводящем состоянии, и схема рис. 4.1 будет иметь высокий уровень выходного напряжения, а другая схема (рис. 4.2) – низкий.

Отметим, что затемненный фототранзистор работает в режиме с обрванной базой, который характеризуется большим остаточным током (см. п. 1.6.3). Если оптрон имеет вспомогательный вывод базы, то для лучшего запирания следует подключить резистор $R_{СМ}$ (см. п. 1.6.4), показанный пунктиром.

При задании единичного входного тока ($U_{МС} \approx 0$) светодиод излучает свет, который вызывает генерацию носителей в базе фототранзистора и появление базового тока. В остальном транзистор работает как обычно, то есть усиливает базовый ток в примерно в β раз и, в конечном счете, выходной коллекторный ток транзистора определяется входным током оптрона. В цифровых схемах при единичном токе через светодиод (см. далее) транзистор должен быть насыщен. Поэтому на выходе схемы, приведенной на рис. 4.1, будет нулевой сигнал, а в другой схеме (рис. 4.2) – единичный.

Таким образом, можно считать, что одна из схем инвертирует входной логический сигнал, а другая повторяет. Выбор внешних сопротивлений был описан ранее. Например, сопротивление R_K рассчитывается по формуле (2.18), а R_E по формуле (2.17). Оценка входного тока и сопротивления $R_{ОПР}$ описывается далее.

Общей особенностью схем 1 и 2 является возможность выбора уровня единичного сигнала за счет соответствующего выбора напряжения питания, то есть выходы обеих схем в определенной степени универсальные. Рассмотренную схему можно использовать также для преобразования двуполярного напряжения в ТТЛ сигнал (см. п. 2.5.5).

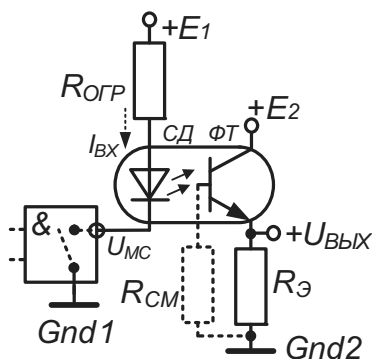


Рис. 4.2

4.1.2. Коэффициент передачи тока оптрона

является основным параметром транзисторного оптрона и определяется как отношение

$$K_I = I_{\text{ВЫХ}} / I_{\text{ВХ}} \quad (4.1)$$

В справочниках обычно приводится в процентах и имеет типовое значение 50..100%. Если в выходной цепи оптрона используется составной транзистор, то коэффициент может достигать 500..800%.

Величина K_I имеет большой технологический разброс, сильно зависит от температуры и от режима, поэтому в ряде случаев в технической документации приводится семейство передаточных характеристик (рис. 4.3).

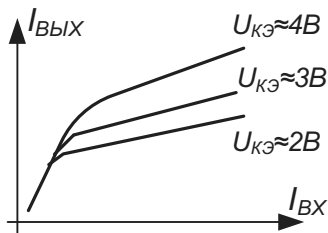


Рис. 4.3

Линейные участки характеристики оптрона могут использоваться в аналоговых цепях. Но следует иметь в виду, что стабильность подобных устройств будет невысокой.

В цифровых цепях, которые ориентируются на худший случай, нестабильность коэффициента передачи тока на работе практически не сказывается. Например, для рассмотренных схем определяют значение тока $I_{\text{МАХ}}$ в выходной цепи ($\approx E_2 / R_K$), делят на минимальный коэффициент передачи тока выбранного оптрона и получают нужное значение $I_{\text{ВХ}}$, необходимое для расчета $R_{\text{ОГР}}$ (см. п. 3.1.2).

4.1.3. Параметры транзисторного оптрона

Кроме обычных параметров, характеризующих светодиод и фототранзистор (напряжения и токи в проводящем и непроводящем состоянии, предельно допустимые значения и пр.), оптрон имеет *специфические* параметры, описывающие сопротивление изоляции, проходную емкость, допустимое напряжение между входом и выходом, которые обязательно нужно учитывать на практике и оговаривать в техническом задании.

Например, в некоторых случаях между входом и выходом может действовать сетевое напряжение 220 В, имеющее амплитудное значение около 300 В. Поэтому для надежности это допустимое напряжение оптрона нужно выбирать более 500 В.

Еще одним важным параметром оптрона является его *быстродействие*, которое оценивают при подаче скачка входного тока.

Контролируя форму выходного импульса (рис. 4.4), можно определить времена включения и выключения, которые имеют величину от единиц до десятков *мкс*, зависящую от типа оптрона и выходного тока.

Обычно процесс выключения более затянут, потому что при включении фотогенерация носителей в базе происходит форсировано, за счет энергии света. А удаление заряда из базы происходит в пассивном режиме (слабого запирающего сигнала), в основном за счет незначительной рекомбинации неосновных носителей в базе (см. п. 1.9.4). Несколько повышает быстродействие наличие резистора R_{CM} , который создает дополнительный путь для удаления заряда. Однако коэффициент передачи тока при этом уменьшается.

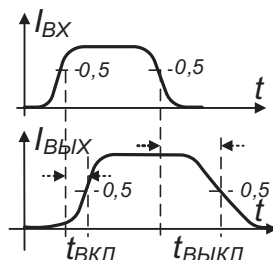


Рис. 4.4

При использовании справочных данных о времени переключения необходимо обратить внимание на схему, при которой измерялись временные параметры, так как они сильно зависят от режима.

В заключение отметим, что диодный оптрон имеет лучшее быстродействие, чем транзисторный, так как при выключении заряд из *p-n* перехода достаточно быстро удаляется внутренним электрическим полем. Но малый выходной сигнал фотодиода требует усиления, поэтому кроме светодиода и фотодиода в микросхему диодного оптрона иногда включают дополнительный усилитель.

4.1.4. Оптронный преобразователь тока в ТТЛ сигнал

рассмотрим на примере микросхемы К293ЛП1, которая выполнена на основе комбинации диодного оптрона и ТТЛ схемы в одном корпусе (рис. 4.5).

ТТЛ каскад представляет собой двухключевую выходную цепь на транзисторах T_1 и T_2 (см. п. 2.2.2), «фазоразделитель» на транзисторе T_3 (см. п. 2.3.2) и усилитель для фотодиода на транзисторе T_4 .

Фотодиод ФД работает на обратной ветви своей характеристики (рис. 4.6) в так называемом вентильном режиме, как источник фототока I_Φ .

При нулевом входном токе фотодиод будет затемнен и иметь весьма малый обратный ток (точка «1» на рисунке). Поэтому напряжение

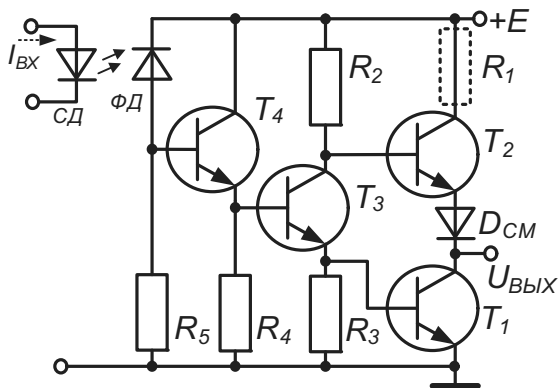


Рис. 4.5

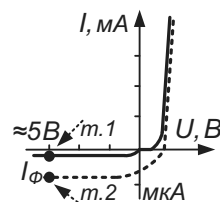


Рис. 4.6

на базе транзистора T_4 будет практически нулевым. В свою очередь непроводящее состояние транзистора T_4 лишает управляющего тока базовые цепи транзисторов T_3 и T_1 , которые также не проводят. И будет насыщен только транзистор T_2 (см. п. 2.3.4), обеспечивая единичное значение выходного сигнала, близкое к напряжению питания.

Когда задается единичное (паспортное) значение входного тока, светодиод освещает фотодиод, характеристика последнего смещается в точку «2». Появляется фототок I_Φ , который почти полностью уходит в базу транзистора T_4 , вызывая возникновение его усиленного эмиттерного тока, который переводит транзисторы T_3 и T_1 в режим насыщения. В этом случае транзистор T_2 не проводит (см. п. 2.3.4) и выходное напряжение будет нулевым.

Рассмотренная схема имеет лучшее быстродействие, чем транзисторный оптрон, поскольку имеет время переключения в доли мкс.

Отметим также, что данная схема в отличие от транзисторного оптрона, напряжение питания которого может изменяться в широких пределах, подключается к стандартному источнику (обычно 5 В) и потребляет большую мощность.

По-прежнему важным параметром для оптронной микросхемы является допустимое напряжение между входом и выходом.

Кроме рассмотренных биполярных транзисторных устройств с оптической связью на практике широко применяют так называемые «оптореле» на основе полевых транзисторов, которые предназначены для коммутации сигналов различного уровня и мощности, как в силовых цепях, так и в измерительных (см. п. 15.4.2).

§4.2. Оптически управляемые тиристорные ключи

Чтобы развязать информационный сигнал вычислительного устройства от электрически опасной промышленной сети и обеспечить нужную мощность в цепи управления технологическим оборудованием, применяют высоковольтные сильноточные ключи с гальваническим разделением. Наиболее простыми из них являются оптически управляемые тиристоры.

4.2.1. Основные свойства тиристора

Форма резко нелинейной вольтамперной характеристики тиристора (рис. 4.7), имеющей участок с отрицательным сопротивлением, определяется напряжением включения $U_{\text{вкл}}$ [28], при котором тиристор лавинообразно переходит из непроводящего состояния в проводящее.

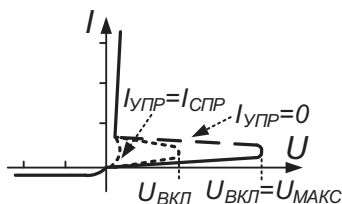


Рис. 4.7

Обычно считают, что при нулевом токе управления $I_{\text{упр}}$ тиристор не проводит и способен выдержать большое напряжение, не превосходящее $U_{\text{макс}}$.

Если ток $I_{\text{упр}}$ отличен от нуля, то происходит уменьшение напряжения $U_{\text{вкл}}$ (показано пунктиром), а при достаточно большом токе $I_{\text{упр}} = I_{\text{спр}}$, который называют током спрямления, тиристор почти сразу переходит в проводящее состояние.

Отсюда следует, что для включения тиристора необходимо задавать в цепь управления ток больший, чем величина $I_{\text{спр}}$.

Главным достоинством тиристора является его способность коммутировать большие токи величиной десятки и даже сотни Ампер при напряжениях, достигающих тысячи Вольт.

Другое ценное свойство тиристора связано с тем, что после включения ток управления можно убирать, так как тиристор сам себя поддерживает в проводящем состоянии. Эта особенность существенно снижает мощность в цепи управления, что особенно важно в силовых ключах.

Отметим также, что тиристоры дешевле других высоковольтных и мощных ключей, например, на основе комбинированной технологии полевых и биполярных транзисторов (см. п. 3.4.3).

4.2.2. Недостатки тиристора

На практике приходится вместе с тем мириться и с определенными недостатками тиристоров, которые описаны ниже.

Для выключения тиристора необходимо отключить его напряжение питания. Для этого в цепях постоянного напряжения приходится усложнять силовую питающую часть. Но в промышленной сети переменного напряжения это делается «автоматически» каждые полпериода, поэтому тиристоры и применяются в основном в устройствах переменного тока.

Отметим в связи с этим, что разработаны и более сложные запираемые тиристоры [5], которые выключаются обратным импульсом достаточно большого управляющего тока.

Для цепей переменного тока оказываются полезными и так называемые симисторы, имеющие симметричную характеристику при обеих полярностях питающего напряжения [5].

Быстродействие тиристора *невелико*, так как включается он в лучшем случае за несколько микросекунд, а выключается за время на порядок большее. Но в силовых цепях, работающих с инерционными внешними объектами, подобные задержки обычно игнорируются.

При резком изменении коммутируемого напряжения ток перезаряда внутренних паразитных емкостей может вызвать протекание ложного управляющего импульса, и тиристор может самопроизвольно перейти в проводящее состояние. Чтобы избежать этого нежелательного переключения рекомендуют принимать меры, уменьшающие резкие изменения питающего напряжения, например, включать параллельно тиристорам небольшие сглаживающий конденсатор и токоограничивающий резистор.

4.2.3. Оптически управляемый тиристорный ключ

выполнен (рис. 4.8) на основе светодиода и фототиристора, которые образуют тиристорный оптрон. Отметим, что мощные устройства такого типа могут иметь внутри единого корпуса дополнительный усилительный каскад, не показанный на рисунке, и их обычно называют опто-тиристорами (например, ТО125-10).

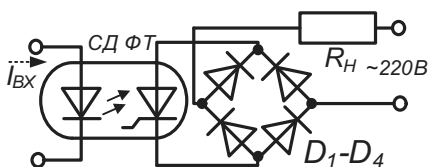


Рис. 4.8

Нагрузка включена в цепь переменного тока, тогда как выпрямительный мост $D_1..D_4$ обеспечивает для тиристора необходимую однополярность напряжения.

В затемненном состоянии тиристор не проводит даже при самых больших значениях сетевого напряжения, что обеспечивается выбором типа оптотиристора по величине $U_{\text{МАКС}}$ (см. рис. 4.7), которое должно быть больше амплитуды сетевого напряжения ($\approx 300 \text{ В}$).

При подаче единичного входного тока $I_{\text{ВХ}}$ светодиод освещает тиристор и у него появляется управляющий фототок, превышающий ток срабатывания. Поэтому тиристор переводится в проводящее состояние и замыкает диагональ выпрямительного моста. Поскольку падение напряжения на тиристоре и диодах моста будут минимальными, то к нагрузке будет приложено почти все сетевое напряжение.

Таким образом, можно управлять состоянием тиристора и, в конечном счете, током через нагрузку. Подобный режим, когда тиристор подобно контактам электромеханического реле включен или выключен, называют релейным.

4.2.4. Особенности релейного режима

рассмотрим на примере временных диаграмм (рис. 4.9), где показаны входной ток, напряжение и ток оптотиристора. В работе ключа можно выделить три этапа.

Первый этап соответствует исходному нулевому значению входного тока. При этом ток через тиристор и нагрузку отсутствует и практически все сетевое напряжение (в виде выпрямленных полуwave) выделяется на тиристоре.

Второй этап начинается при подаче скачка входного тока, который переводит тиристор в проводящее состояние. Сигнал управления тиристора и сетевое напряжение асинхронны, поэтому момент включения может совпасть по времени, например, с максимумом синусоиды. Соответственно и ток проводящего тиристора при активной нагрузке будет в этот момент изменяться от нуля до максимального. В дальнейшем форма тока будет соответствовать полуволнам синусоиды.

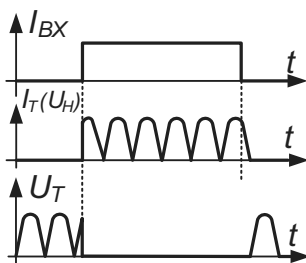


Рис. 4.9

При таких резких изменениях многоамперных токов нагрузки, создаются мощные электромагнитные импульсы помехи, которые могут нарушить работу некоторых электронных устройств, на-

пример, в энергосберегающих лампах. Поэтому на практике стараются уменьшить перепады тока нагрузки, производя включение тиристора вблизи момента перехода синусоиды через ноль.

Чтобы избавить процессор от необходимости контролировать опасное сетевое напряжение, некоторые тиристорные оптроны имеют встроенные схемы, обеспечивающие включение тиристора только вблизи нуля. Этим исключаются резкие изменения тока нагрузки и снижаются электромагнитные помехи.

На *третьем* этапе входной ток становится нулевым, но выключения тиристора сразу не происходит, так как он сам себя удерживает в проводящем состоянии до тех пор, пока полуволна внешнего напряжения не снизится до нуля. Подобная задержка при выключении может достигать почти полупериода сетевого напряжения, но особых проблем для инерционных релейных объектов это в большинстве случаев не создает.

Для более плавного управления мощностью на нагрузке, которое не может обеспечить релейный режим работы тиристора, применяют фазовый метод (см. далее), в основе которого лежит ШИМ (см. п. 1.1.5).

4.2.5. Фазоимпульсное управление тиристором

производится короткими импульсами, включающими тиристор в определенные моменты полупериода сетевого напряжения (рис. 4.10), и основано на способности тиристора сохранять без сигнала управления проводящее состояние до окончания полупериода.

Изменяя фазу управляющих импульсов, можно достаточно плавно регулировать среднюю мощность, выделяющуюся на нагрузке, которая пропорциональна площади заштрихованных на рисунке участков. Причем на тиристоре, работающем в ключевом режиме (см. п. 1.1.5), выделяется мощность значительно меньшая, чем в нагрузке, то есть КПД подобных регуляторов достаточно большой и может достигать 90%.

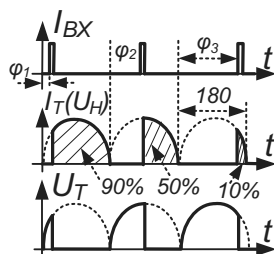


Рис. 4.10

Вместе с тем фазовый метод регулирования имеет и ограничения. Например, нагрузка должна осреднять усеченные полуволны тока. Поэтому подобный способ чаще всего используют для инерционных внешних объектов типа нагревателей или элект-

родвигателей, которые не реагируют на высокочастотные составляющие такой формы напряжения.

Другой недостаток фазоимпульсного управления тиристором связан с большим уровнем помех при включении тиристора и скачках большого тока нагрузки. Чтобы исключить резкие изменения тока нагрузки, включение тиристора можно производить вблизи перехода синусоиды через ноль (см. п. 4.2.4), а ступенчатое регулирование мощности осуществлять пропуском полупериодов сетевого напряжения.

4.2.6. Управление мощностью за счет пропуска полупериодов

можно пояснить, используя временные диаграммы, приведенные на рис. 4.11 для трех случаев мощности в нагрузке. В первом случае процессор разрешает тиристор пропустить на нагрузку один полупериод из десяти, во втором – пять, в третьем – девять. За счет этого принципа осуществляется ступенчатое регулирование мощности в нагрузке, причем управляющий импульс для тиристора генерируется вблизи перехода синусоиды через ноль, что позволяет исключить резкие изменения тока нагрузки и, соответственно, снизить электромагнитные помехи.

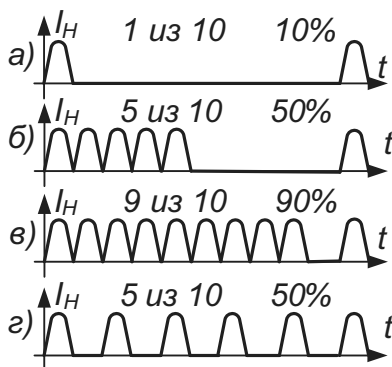


Рис. 4.11

Недостатком такого способа является наличие низкочастотных (звуковых) составляющих тока нагрузки, особенно они заметны на рис. 4.11б. Чтобы уменьшить их, включенные полупериоды можно чередовать (рис. 4.11г), используя, например, алгоритм Брезенхема, применяемый в растровой графике.

§4.3. Особенности электромагнитных цепей

Альтернативой оптической связи для передачи данных в гальванически разделенных цепях является магнитная связь, основные сведения о которой приведены ниже.

4.3.1 Основные магнитные параметры

Магнитное поле характеризуется двумя главными параметрами – напряженностью магнитного поля H , которая измеряется в A/m , и магнитной индукцией B , которая измеряется в $Tл$ – (Теслах).

Индукция B и напряженность H связаны между собой (подобно напряжению и току в электрической цепи) определенной зависимостью – кривой намагничивания. Эта зависимость в общем случае нелинейная, и ее представляют, обычно в виде так называемой петли гистерезиса (рис. 4.12).

Примечание: магнитная характеристика воздуха практически линейна и почти не имеет участка магнитного насыщения, что используется в некоторых случаях (см. п. 4.5.2).

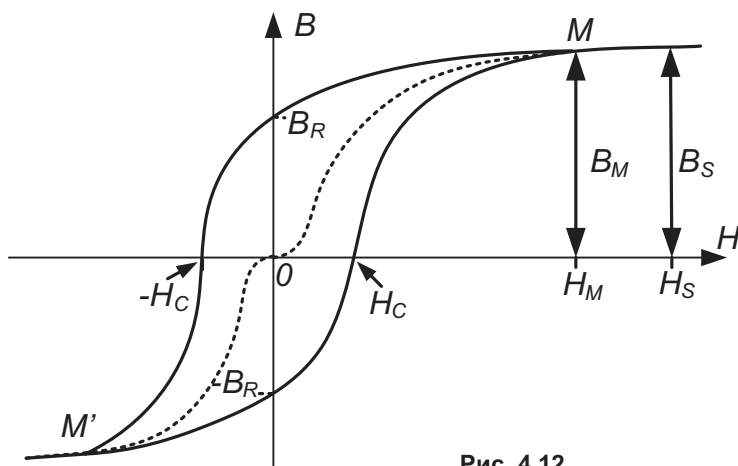


Рис. 4.12

Но на практике для простоты стараются по возможности пренебречь нелинейностью и использовать приближенную пропорциональную зависимость, похожую на закон Ома:

$$B = \mu_A H = \mu \mu_0 H, \quad (4.2)$$

где μ_A (Г/м) – абсолютная магнитная проницаемость вещества, характеризует наклон магнитной характеристики (и в связи с этим является аналогом электрического сопротивления, которое тоже характеризует наклон вольт-амперной характеристики);

$\mu_0 = 4\pi 10^{-7}$ (Г/м) – магнитная проницаемость вакуума (физическая константа);

μ – относительная магнитная проницаемость, показывающая во

сколько раз абсолютная проницаемость больше проницаемости вакуума.

На практике часто вместо значения μ используют импульсную (динамическую) магнитную проницаемость $\mu_{\text{и}}$ отражающую наклон магнитной характеристики в *рабочей* зоне (см. п. 4.5.1).

Кроме того, важным параметром, характеризующим магнитную цепь, является и магнитный поток Φ (измеряется в Веберах). Для равномерного поля:

$$\Phi = B \cdot S, \quad (4.3)$$

где S – площадь (в м^2), через которую проходит магнитный поток.

В электромагнитных расчетах при оценке взаимного преобразования электрических и магнитных параметров используются два основных закона, закон полного тока и закон электромагнитной индукции.

4.3.2 Закон полного тока

характеризует взаимную зависимость электрического тока и напряженности магнитного поля.

Когда источник питания подключается к обмотке трансформатора с числом витков W (рис. 4.13), то ток I , протекающий по обмотке, создает намагничивающую (магнитодвижущую) силу, измеряемую в *А/виток*:

$$F = I \cdot W. \quad (4.4)$$

При этом возникает внешнее магнитное поле, которое пропорционально току и характеризуется, как отметили ранее, напряженностью H , потоком Φ и индукцией B .

Согласно закону полного тока среднее значение напряженности магнитного поля равно намагничивающей силе F , то есть произведению $I \cdot W$:

$$H \cdot \ell_{\text{ср}} = F = I \cdot W, \quad (4.5)$$

где $\ell_{\text{ср}}$ – средняя длина сердечника (в м).

Если обмоток две, то первичный ток тратится не только на создание напряженности магнитного поля, но и на создание вторичного тока:

$$I_1 \cdot W_1 = H \cdot \ell_{\text{ср}} + I_2 \cdot W_2 \quad (4.6),$$

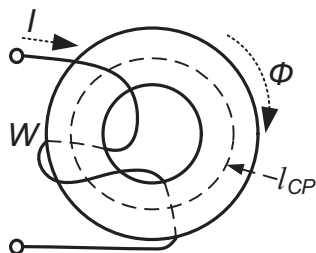


Рис. 4.13

Это важное уравнение является связующим между электрическими токами и магнитной напряженностью, то есть, зная токи, можно найти напряженность или наоборот. Например, если известно, что ток меняется до некоторого максимального значения, то можно оценить и максимальное значение напряженности.

4.3.3. Закон электромагнитной индукции (закон Фарадея)

дает возможность вычисления уже не тока, а напряжения на основе магнитных параметров. Согласно этому закону при всяком изменении магнитного потока в обмотке индуцируется ЭДС, величина которой пропорциональна скорости изменения магнитного потока и числу витков:

$$e = W \cdot d\Phi / dt = W \cdot S \cdot dB / dt. \quad (4.7)$$

Это уравнение часто представляется в другой, интегральной форме, которую мы рассмотрим ниже (см. п. 4.4.2). Но главное в обеих формах – возможность вычисления напряжения на основе индукции или индукции на основе напряжения.

Например, задавая напряжение на одной обмотке, можно найти индукцию в сердечнике, а затем определить напряжение на другой обмотке (см. п. 4.2.3).

Конкретные примеры использования закона полного тока и электромагнитной индукции даются в других параграфах.

Для более сложных расчетов можно использовать и другие соотношения для магнитных цепей, например, закон Ома, первый и второй законы Кирхгофа [20].

4.3.4. Магнитные материалы

В зависимости от магнитных свойств вещества делятся на *ферромагнитные* и *неферромагнитные* (диамагнетики и парамагнетики). Если ферромагнетик поместить во внешнее магнитное поле, то индукция в нем окажется в сотни и тысячи раз больше, чем во внешнем поле.

Для диа- и парамагнетиков такого увеличения поля не наблюдается.

К ферромагнетикам относятся электротехнические стали (железо с кремнием), пермаллой (железо с никелем), ферриты (неметаллические соединения окислов железа и других металлов), ферромагнитные пленки, редкоземельные вещества.

В свою очередь ферромагнитные материалы в зависимости от степени гистерезиса делят на два класса – *магнитотвердые* ($H_c > 10^3 \text{ А/м}$), которые используют для изготовления постоянных магнитов и *магнитомягкие* ($H_c < 10^3 \text{ А/м}$), которые применяются в магнитных сердечниках.

§4.4. Уравнения импульсного трансформатора

Импульсные трансформаторы широко применяют для гальванического разделения как силовых цепей (например, в импульсных источниках питания или в промышленном оборудовании), так и информационных (в локальных сетях и телекоммуникационном оборудовании). Ниже описываются основные соотношения, связывающие напряжения и токи трансформатора с его магнитными параметрами.

4.4.1. Общие сведения

Импульсный трансформатор выполняют обычно на основе тороидального (рис. 4.13), П- или Ш-образного магнитного сердечника, на который наносят две или более обмоток.

Взаимное пространственное расположение обмоток на сердечнике совместно с направлением токов определяют направление магнитных потоков в сердечнике. При переходе к условному графическому обозначению (УГО) трансформатора, приведенному на рис. 4.14, теряется возможность определения направления намагничивающих сил и потоков. Поэтому для УГО трансформатора вводят условную разметку зажимов обмоток.

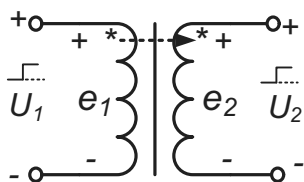


Рис. 4.14

Зажимы двух обмоток принято называть *одноименными* (имеющими одинаковую полярность), если одинаково направленные по отношению к ним токи создают в сердечнике одинаково направленные магнитные потоки. Одноименные выводы помечают «точкой» (звездочкой) или «не точкой».

Иногда выводы обмоток отмечают как начало и конец, но в этом случае может возникнуть неопределенность. Например, два на-

чала обмоток могут быть не одноименными, если они входят в сердечник с разных сторон.

При определении полярностей наводимых на обмотках ЭДС пользуются правилом: *на всех одноименных выводах действуют одинаковые полярности ЭДС.*

Поэтому, если полярность сигнала на одной обмотке известна (задана), то, пользуясь понятием одноименности, можно определить полярности и на других обмотках.

4.4.2. Интегральное уравнение импульсного трансформатора

Подадим на первичную обмотку трансформатора (рис. 4.14) однополярный импульс U_1 . Через обмотку потечет ток и согласно закону электромагнитной индукции в первичной обмотке W_1 наведется обратная по полярности ЭДС e_1 , препятствующая действию входного импульса. И эта ЭДС будет трансформирована во вторичную обмотку в виде e_2 .

Таким образом, полярность e_1 будет обратной U_1 , а полярность e_2 будет определяться фазировкой обмоток – *на одноименных выводах обмоток будут одинаковые знаки.*

Не учитывая активного сопротивления обмотки, а так же пренебрегая влиянием паразитных ёмкостей и рассеянием магнитного потока, можно записать в соответствии с формулами (4.7) и (4.3):

$$U_1 = e_1 = W_1 \cdot d\Phi / dt = W_1 \cdot S \cdot dB / dt. \quad (4.8)$$

Решая дифференциальное уравнение (4.8), полагая, что $U_1 = \text{const}$, можно определить закон изменения индукции во времени:

$$B(t) = B_0 + \frac{1}{W_1 S} \int_0^t U_1(t) dt = B_0 + \frac{U_1 \cdot t}{W_1 \cdot S}, \quad (4.9)$$

где B_0 – начальное значение индукции.

Это *основное уравнение импульсного трансформатора*, которое показывает, что при прямоугольном скачке сигнала индукция в сердечнике изменяется по линейному закону. Причем чаще всего используется на практике приращение индукции, определяемое вторым слагаемым (4.9):

$$\Delta B = U_1 t_n / W_1 \cdot S, \quad (4.10)$$

которое будет пропорционально произведению $U_1 \cdot t_n$, то есть – *вольт-секундной площади импульса.*

Последнее уравнение является достаточно универсальным и позволяет оценивать число витков при прочих заданных величинах, находить максимально возможную длительность импульса для реального трансформатора (см. п. 4.5.3) или определять изменение индукции.

4.4.3. Оценка вторичного напряжения

Напряжение на вторичной обмотке трансформатора U_2 определяется индуцируемой ЭДС e_2 , которая, как и в первичной обмотке согласно (4.7), пропорциональна изменению магнитного потока и числу витков:

$$U_2 = e_2 = W_2 \cdot d\Phi / dt. \quad (4.11)$$

Подставляем сюда значение $d\Phi/dt$, которое можно найти из уравнения (4.8), получим

$$U_2 = U_1 W_2 / W_1, \quad (4.12)$$

то есть в идеализированном трансформаторе выходное напряжение пропорционально входному и повторяет его форму. Реальный же трансформатор несколько искажает форму выходного импульса (см. §4.6).

4.4.4. Оценка токов трансформатора

Следующие, очень важные для практики уравнения трансформатора основаны на законе полного тока. Поделив обе части уравнения (4.6) на W_1 , можно записать в компактной форме ток I_1 , протекающий в первичной обмотке:

$$I_1 = I_L + I_2', \quad (4.13)$$

где $I_L = H \ell_{CP} / W_1$ – так называемый намагничивающий ток, создающий напряженность магнитного поля,

$I_2' = I_2 W_2 / W_1$ – приведенное в первичную обмотку значение вторичного тока.

Для намагничивающего тока I_L , используя уравнения (4.2) и (4.10), можно получить следующее соотношение:

$$I_L = H \ell_{CP} / W_1 = B \cdot \ell_{CP} / \mu \mu_0 \cdot W_1 = U_1 \cdot t \ell_{CP} / \mu \mu_0 S W_1^2. \quad (4.14)$$

Часть сомножителей в уравнении (4.14) обычно обозначают как индуктивность намагничивания сердечника (для идеализированного трансформатора равна индуктивности L первичной обмотки):

$$L = \mu \mu_0 S W_1^2 / \ell_{CP} \quad (4.15)$$

Тогда для оценки тока I_L вместо (4.14) можно использовать упрощенное расчетное соотношение:

$$I_L = U_1 \cdot t / L. \quad (4.16)$$

Из формулы видно, что, если бы $L = \text{const}$, то ток I_L нарастал бы со временем по линейному закону. Однако на самом деле закон изменения I_L нелинеен (см. далее) и связан с законом изменения $H(t)$ при линейном характере $B(t)$, то есть определяется, в конечном счете, формой петли гистерезиса сердечника.

4.4.5. Временные диаграммы

При подаче скачка напряжения на первичную обмотку индукция в сердечнике B согласно уравнению (4.10) линейно растет со временем (рис. 4.15). Аналогично растет и напряженность H , но уже нелинейно, в соответствии с нелинейной магнитной характеристикой сердечника. И, наконец, согласно следствию из закона полного тока (см. п. 4.4.4), ток намагничивания I_L , который пропорционален напряженности H , тоже имеет нелинейную форму.

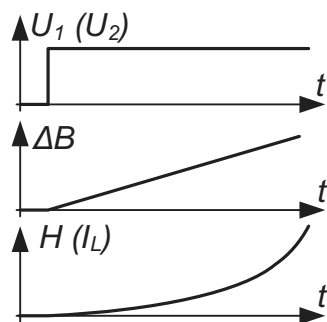


Рис. 4.15

Указанные зависимости предполагают, что магнитный сердечник работает на примерно линейных участках своей характеристики. В этом случае выходное напряжение U_2 повторяет форму входного U_1 . Причиной сильного искажения сигналов при выходе сердечника в режим насыщения может быть, например, большая длительность импульса (см. п. 4.5.3).

§4.5. Режимы работы импульсного трансформатора

4.5.1. Воздействие на сердечник однополярных импульсов

Пусть сердечник находится в начальной точке «0» своей магнитной характеристики (рис. 4.16). После подачи первого импульса индукция в сердечнике изменится в соответствии с (4.10) на величину

ΔB_1 и магнитное состояние будет характеризоваться точкой «1». После исчезновения импульса сердечник назад в нулевую точку не вернется, а из-за гистерезиса попадет в точку «2». Затем после серии импульсов сердечник будет последовательно переходить в точки «3», «4» и далее, перемещаясь вверх по характеристике.

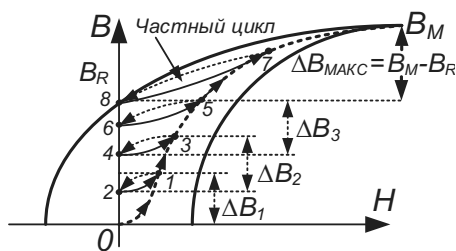


Рис. 4.16

После воздействия серии однополярных импульсов перемагничивание сердечника будет происходить уже по предельному частному циклу (точки «8» и «7»), причём в этом режиме импульсная магнитная проницаемость $\mu_{и}$ (то есть наклон рабочей зоны характеристики) оказывается значительно меньше магнитной проницаемости на предельном цикле, измеренной при воздействии на трансформатор двухполярного напряжения симметричной формы. Особенно мало значение $\mu_{и}$ в сердечниках с прямоугольной петлёй гистерезиса, которые поэтому не рекомендуется использовать для импульсных трансформаторов.

При выходе сердечника в область насыщения резко увеличивается ток первичной обмотки (из-за уменьшения коэффициента μ , а значит и величины L), снижается экономичность схем, искажается форма импульсов, создаются сильные электромагнитные помехи. Поэтому для однополярного режима возможное изменение индукции ограничено, как видно из рисунка, значениями B_M и B_R , то есть

$$\Delta B_{\text{макс}} \approx B_M - B_R \quad (4.17)$$

Тогда как при двухполярном перемагничивании сердечника (рис. 4.12) используется вся предельная петля гистерезиса, и резко возрастают наклон характеристики и возможное изменение индукции

$$\Delta B_{\text{макс}} \approx 2B_M \quad (4.18)$$

Поэтому очевидным недостатком работы импульсного трансформатора с однополярным напряжением являются малые значения $\Delta B_{\text{макс}}$ и $\mu_{и'}$, вызванные работой сердечника на частном цикле.

В еще худших условиях работают дроссели в цепях постоянного тока источников питания, например, в сглаживающих фильтрах. Постоянный ток, достигающий величины в несколько Ампер, изначально смещает сердечник в область насыщения, делая дроссель неработоспособным из-за очень малой магнитной проницае-

мости. Чтобы устранить этот недостаток сердечник дросселя выполняют с зазором.

4.5.2. Введение немагнитного зазора

Чтобы увеличить магнитную проницаемость на частном цикле, магнитную характеристику сердечника деформируют с помощью немагнитного (например, воздушного) зазора, имеющего линейную безгистерезисную характеристику (прямая «0–1» на рис. 4.17а).

В этом случае в сердечнике (рис. 4.17б) образуется состоящая из двух материалов *последовательная* магнитная цепь. И для вычисления результирующей индукции нужно просуммировать значения отдельных составляющих индукции (как напряжения в последовательной электрической цепи).

Например, координата точки «2» результирующей характеристики, приведенной на рисунке пунктиром, получается алгебраическим суммированием отрезков «3–4» и «5–6».

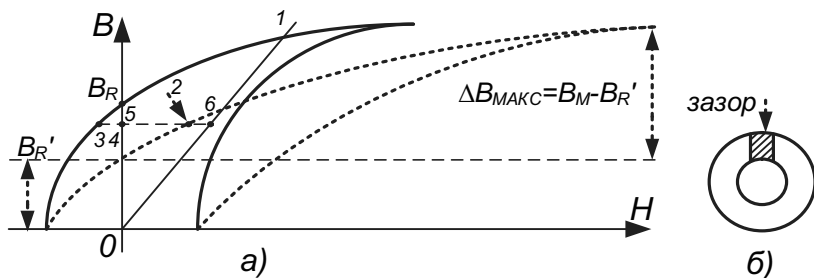


Рис. 4.17

Подобная наклоненная петля гистерезиса, показанная на рисунке пунктиром, имеет частный однополярный цикл со значительно большими магнитной проницаемостью μ и значением $\Delta B_{\text{МАКС}}$.

4.5.3. Действие импульса большой длительности

Из основного уравнения трансформатора не видны ограничения на длительность передаваемого импульса, и можно сделать неправильный вывод, что трансформатор пригоден для работы со сколь угодно длинными импульсами. Действительно, импульсный трансформатор правильно воспроизводит входной импульс, если обеспечить примерно линейное изменение индукции во времени и большую постоянную времени $\tau = L / R_l$ (см. п. 4.6.2).

Однако при больших значениях $t_{\text{и}}$ сердечник может выйти в область насыщения, в которой малым приращениям индукции соответствуют большие приращения напряжённости магнитного поля. При этом резко увеличивается величина тока I_L (см. рис. 4.18), поскольку падает магнитная проницаемость сердечника и, соответственно, значения L .

Большие значения тока I_L источник сигнала, как правило, не способен обеспечить и ток в первичной обмотке ограничивается на уровне $I_{\text{МАКС}}$. Перестают увеличиваться H и B . Резко уменьшается ЭДС, которая пропорциональна приращению индукции, поэтому выходной импульс будет резко искажаться.

Максимальную длительность передаваемого импульса можно оценить, учитывая, что возможное изменение индукции для однополярного сигнала ограничено (см. п. 4.5.1).

Тогда, используя формулу (4.10) с учетом (4.17), можно найти максимальную длительность импульса:

$$t_{\text{МАКС}} = W_1 \cdot S \cdot (B_M - B_R) / U_1 \quad (4.19)$$

Существуют и другие причины искажения выходного импульса (см. §4.6), но в любом случае постоянную составляющую сигнала трансформатор не пропускает.

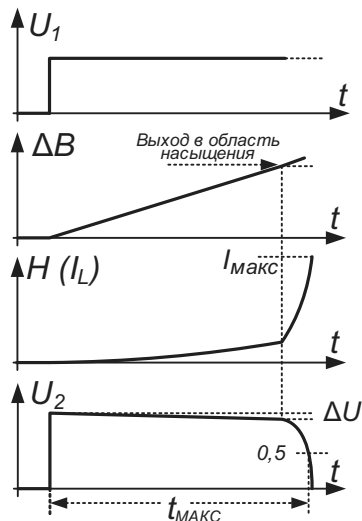


Рис. 4.18

§4.6. Искажения, вносимые трансформатором

4.6.1. Эквивалентная схема трансформатора

При анализе переходных процессов в импульсном трансформаторе можно использовать эквивалентную схему замещения (рис. 4.19), в которой приняты следующие обозначения:

ИдТр – идеальный трансформатор с обмотками W_1 и W_2 ;

r_1 и r_2 – активные (омические) сопротивления обмоток;

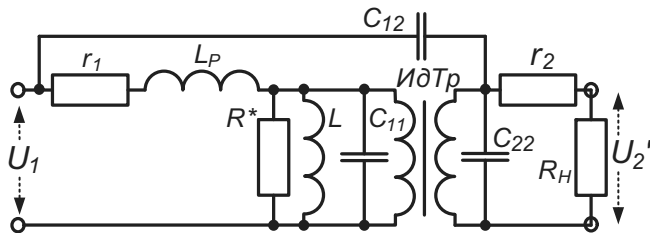


Рис. 4.19

R_H – сопротивление нагрузки;

L – индуктивность намагничивания сердечника;

L_p – индуктивность рассеяния, отражающая то, что часть магнитного потока, создаваемого первичной обмоткой, проходит вне вторичной обмотки, рассеиваясь в пространстве;

C_{11} , C_{12} и C_{22} – эквивалентные паразитные ёмкости обмоток, учитывающие распределённые электростатические связи между витками и обмотками трансформатора;

R^* – сопротивление, отражающее наличие активных потерь в сердечнике трансформатора.

Для упрощения анализа формы выходного импульса трансформатора обычно отдельно рассматривают формирование фронта и вершины импульса. При анализе фронта учитывают, что ток I_L в индуктивности намагничивания за короткое время не успевает нарасти, и пренебрегают наличием L . В случае оценки искажения вершины импульса наоборот можно пренебречь влиянием индуктивности рассеяния L_p и паразитными ёмкостями, поскольку токи и напряжения при этом изменяются сравнительно медленно.

4.6.2. Формирование вершины импульса

Для приближенной оценки процессов во время импульса пренебрежем быстрыми изменениями сигнала, вносимыми малыми паразитными ёмкостями и индуктивностью рассеяния, и используем упрощенную эквивалентную схему (рис. 4.20), где активное сопротивление первичной цепи определяется суммой сопротивлений источника сигнала $r_{и}$ и ключа $r_{кл}$:

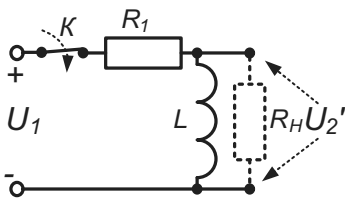


Рис. 4.20

$$R_1 = r_i + r_{и} + r_{кл}. \quad (4.20)$$

Будем также считать, что $R_H \gg R_1$, и далее для простоты не будем учитывать наличие R_H , которое поэтому показано на рисунке пунктиром.

Кроме того отметим, что выходное напряжение в этой схеме U_2' приведено к первичной обмотке, то есть $U_2' = U_2 W_1 / W_2$.

Подадим на вход этой схемы при помощи ключа скачок напряжения U_1 и найдем напряжение U_2' , которое будет отличаться от входного на величину падения напряжения на сопротивлении R_1 :

$$U_2' = U_1 - I_L \cdot R_1. \quad (4.21)$$

Ток I_L в индуктивности, как известно, скачком измениться не может и вначале импульса равен нулю. Поэтому напряжение на выходе в первый момент будет примерно равно U_1 (см. рис. 4.21).

Далее ток в индуктивности начнет увеличиваться по экспоненциальному закону (подобно напряжению на конденсаторе):

$$I_L(t) = I_L(\infty) - [I_L(\infty) - I_L(0)] \exp(-t/\tau), \quad (4.22)$$

где $I_L(0) = 0$ (нулевые начальные условия),

$I_L(\infty) = U_1 / R_1$ (в установившемся режиме индуктивность является короткой для постоянного тока),

$\tau = L / R_1$ (постоянная времени первичной обмотки).

Подставив ток I_L из (4.22) в уравнение (4.21), можно найти выходное напряжение:

$$U_2' = U_1 - I_L \cdot R_1 = U_1 \cdot \exp(-t/\tau), \quad (4.23)$$

которое оказывается спадающим по экспоненте до нуля. Но на практике больших искажений импульса не допускают, выбирая $\tau \gg t_H$. В этом случае как видно из формы импульса на рис. 4.21, нелинейностью вполне можно пренебречь и рассчитать *спад вершины* по упрощенной формуле, используя ток I_L из формулы (4.16) или разложение экспоненты в степенной ряд:

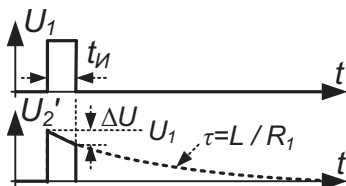


Рис. 4.21

$$\Delta U = I_L(t_H) \cdot R_1 = U_1 \cdot t_H \cdot R_1 / L. \quad (4.24)$$

Последнюю формулу можно использовать и для обратного расчета – нахождения требуемой величины индуктивности L первичной обмотки трансформатора при заданных других параметрах.

4.6.3. Разрыв индуктивной цепи

За время действия импульса в трансформаторе запасается магнитная энергия и при размыкании цепи на обмотках возникает ЭДС с обратной полярностью, как бы стремящаяся поддержать прежнее направление индуктивного тока.

Сердечник находится в точке «7» своей характеристики, приведенной на рис. 4.16, магнитный сердечник при разрыве электрической цепи не получает энергии извне и начинается возврат в исходное состояние (к точке «8»). То есть изменения напряженности и индукции магнитного поля станут отрицательными, при этом генерируется ЭДС самоиндукции обратной полярности, то есть магнитная энергия превращается в электрическую, которая затем рассеивается на активных сопротивлениях, превращаясь в тепло.

На рис. 4.22 представлена эквивалентная схема для указанного процесса. Индуктивность L является элементом, обеспечивающим взаимное преобразование электрической и магнитной энергии. Сопротивление R'' характеризует суммарные активные потери и в простейшем случае равно параллельному соединению сопротивлений, нагружающих обмотки трансформатора и приведенных к первичной обмотке. В нашем случае $R'' = R_n'$, но в некоторых случаях включают дополнительные сопротивления (см. §4.7).

Источник тока I_{L0} отражает предысторию процесса при скачке первичного напряжения (наличие ненулевых начальных условий) и в соответствии с формулой (4.16) равен:

$$I_{L0} = I_L(t_{\mu}) = U_1 t_{\mu} / L. \quad (4.25)$$

В первый момент после размыкания цепи ток в индуктивности L не может, как известно, измениться мгновенно и будет равен нулю. Поэтому ток I_{L0} , учитывающий начальное состояние схемы, будет замыкаться только через сопротивление R'' , определяя амплитуду выброса (см. рис. 4.23):

$$U_B = I_{L0} \cdot R'' = I_L(t_{\mu}) \cdot R''. \quad (4.26)$$

Обратить внимание на направление тока I_{L0} на рис. 4.22 и на то, что полярность выходного напряжения U_2 при этом будет обратной.

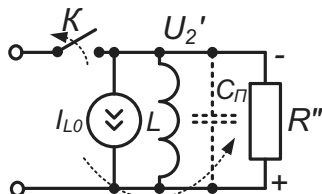


Рис. 4.22

Затем ток в индуктивности будет увеличиваться по экспоненте, а напряжение U_2' будет спадать до нуля, потому что индуктивность для постоянного тока в установившемся режиме представляется короткой.

Важной особенностью выходного напряжения U_2' (рис. 4.23) является то, что площади импульса S_1 и выброса S_2 равны. Другими словами, постоянная составляющая выходного напряжения равна нулю – трансформатор не пропускает постоянную составляющую (см. также п. 4.5.3).

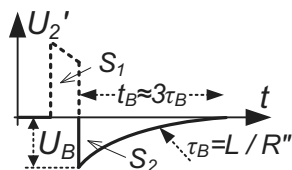


Рис. 4.23

Используя эквивалентную схему рис. 4.22 можно вывести уравнение для напряжения U_2' , похожее на формулу (4.22). Но на практике чаще используют не сам закон изменения напряжения, а два параметра обратного выброса, возникающего при размыкании цепи – амплитуду U_B и длительность восстановления t_B .

Амплитуда U_B , как видно из уравнения (4.26), прямо пропорциональна сопротивлению R'' , и при больших значениях сопротивления может достигать сотен **Вольт**, что может быть опасно для полупроводниковых ключей. Поэтому, чтобы уменьшить выброс, включают параллельно R'' дополнительное сопротивление (см. п. 4.7.1).

Длительность выброса зависит от постоянной времени восстановления $\tau_B = L / R''$ и ее оценивают по приближенной формуле:

$$t_B \approx 3 \tau_B = 3L / R'' \quad (4.27)$$

Пока не закончится выброс (не восстановится исходное состояние магнитного сердечника, когда $H = 0$), новый импульс подавать нельзя. Поэтому в тех случаях, когда время t_B оказывается больше паузы между импульсами, нужно принять меры к уменьшению t_B , например, за счет увеличения R'' . Иначе магнитное состояние сердечника не вернется в исходную точку. При следующих импульсах смещение будет накапливаться, а сердечник окажется в области насыщения, что приведет резким искажениям выходного импульса (см. рис. 4.18).

4.6.5. Колебания при разрыве индуктивной цепи

Иногда обратный выброс сопровождается колебаниями на первичной обмотке (рис. 4.24). Причиной их является ударное возбуждение колебательного контура, обра-



Рис. 4.24

зованного индуктивностью L и паразитной межвитковой емкостью C_{11} первичной обмотки (на рис. 4.22 показана пунктиром).

Для колебательного контура LC_{11} , шунтированного сопротивлением R'' , можно записать характеристическое уравнение [20].

Если корни уравнения действительные, то процесс апериодический (рис. 4.23). Если корни мнимые, то процесс колебательный (рис. 4.24).

Для практических расчетов особый интерес представляет граничный переходной процесс (между колебательным и апериодическим), характеризуемый так называемым критическим сопротивлением колебательного контура

В большинстве практических случаев стараются, чтобы $R'' < R_{кр}$ и выброс на коллекторе транзистора имел характер, близкий к экспоненциальному. Но иногда эти паразитные колебания используют для оценки параметров эквивалентной схемы L и C_{11} .

4.6.6. Экспериментальная оценка параметров трансформатора

Одним из простых способов оценки параметров эквивалентных схем, не требующим сложного оборудования, является исследование затухающих колебаний, возникающих в первичном резонансном контуре. Для этого измеряют их частоту f_1 . Для этой частоты справедливо уравнение [20]:

$$R_{кр} = 0,5 \sqrt{L / C_{11}}. \quad (4.28)$$

Затем к первичной обмотке трансформатора подключают дополнительный конденсатор, известной емкости C_d , и снова измеряют частоту f_2 , для которой можно записать подобное уравнение с учетом дополнительной емкости:

$$2\pi f_1 = 1 / \sqrt{LC_{11}} \quad (4.29)$$

$$2\pi f_2 = 1 / \sqrt{L(C_{11} + C_d)}. \quad (4.30)$$

Решая указанные уравнения (4.29) и (4.30) совместно, находят C_{11} и L .

Примечание: колебания могут возникнуть и по другой причине (см. далее).

4.6.7. Формирование фронта выходного сигнала

При анализе фронтов учитывают, что ток I_L в индуктивности намагничивания за короткое время фронта не успевает нарасти, и

пренебрегают наличием индуктивности первичной обмотки L . Но в упрощенной эквивалентной схеме приходится учитывать индуктивность рассеяния L_p и паразитную межвитковую емкость вторичной обмотки C'_{22} (приведенную к первичной обмотке), которые определяют выходной сигнал в области малых времен. В итоге образуется колебательный LC -контур, шунтированный приведенным к первичной цепи сопротивлением нагрузки R'_H (рис. 4.25).

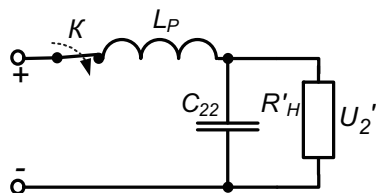


Рис. 4.25

Из анализа характеристического уравнения 2 степени этого контура [20] следует, что переходный процесс может иметь различную форму. Если корни уравнения – мнимые, то процесс колебательный (кривая 1 на рис. 4.26). Если корни – действительные, то процесс аperiodический (кривая 2).

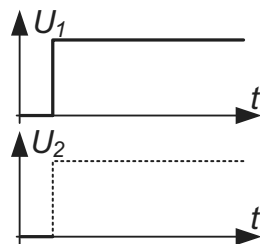


Рис. 4.26

Граница между этими крайними случаями характеризуется еще одним критическим сопротивлением [20]:

$$R_{KP2} = 0,5\sqrt{L_p/C_{22}}. \quad (4.31)$$

На практике при изготовлении трансформатора стараются конструктивными методами по возможности уменьшать паразитные индуктивность рассеяния L_p и емкость C_{22} . Для экспериментальной оценки последних можно использовать уравнения (4.29) и (4.30), заменив в них обозначения индуктивности и емкости.

§4.7. Ключ с импульсным трансформатором

4.7.1. Схема

От традиционного ключа с резистивной связью (см. п. 1.7.1) схема отличается наличием в коллекторной цепи импульсного трансформатора вместо резистора (рис. 4.27).

Трансформатор гальванически изолирует цепь нагрузки от остальной схемы.

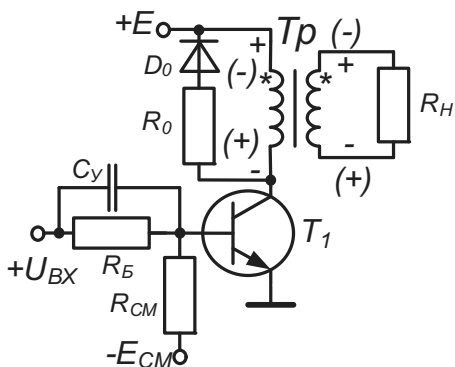


Рис. 4.27

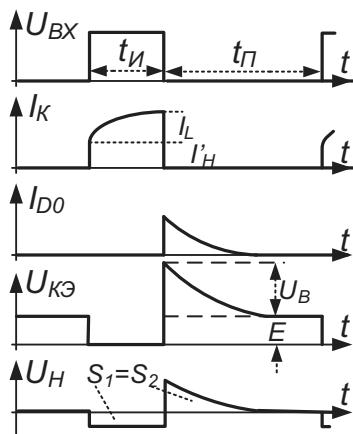


Рис. 4.28

Резистор R_B обеспечивает насыщение транзистора, цепочка R_{CM} и E_{CM} — надежное запирающее устройство, конденсатор C_y улучшает быстродействие.

Диод D_0 и резистор R_0 ограничивают амплитуду выброса на коллекторе (см. п. 4.7.4).

4.7.2. Процесс формирования импульса

При подаче положительного импульса U_{BX} (рис. 4.28) транзистор T_1 насыщается (то есть $U_{KЭ} \approx 0$) и подключает первичную обмотку трансформатора к источнику питания E . Появляется коллекторный ток I_K , вызывающий изменение напряженности H и индукции B . И в результате согласно (4.7) в обмотках возникнет ЭДС (см. п. 4.3.3), препятствующая протеканию тока (полярность показана на рис. 4.27 без скобок).

Индукция в сердечнике, как уже отмечалось (см. п. 4.4.2), будет линейно возрастать. Напряженность H при этом так же будет увеличиваться в соответствии с характеристикой сердечника.

Поскольку диод D_0 при такой полярности не проводит, то коллекторный ток I_K будет равен току первичной обмотке и определяться (см. п. 4.4.4) суммой тока намагничивания I_L и приведенного в первичную обмотку тока нагрузки I'_H . Причем ток намагничивания I_L будет увеличиваться по примерно линейному закону (см. временные диаграммы на рис. 4.28), а ток I_K будет максимален в момент окончания импульса.

Напряжение на нагрузке U_H будет на протяжении импульса пропорциональным напряжению на первичной обмотке (см. п. 4.4.3), которое примерно равно напряжению питания E . То есть форма U_H будет почти прямоугольной в течение времени t_H (более подробно см. п. 4.6.2).

4.7.3. Процесс формирования выброса

При запираании транзистора ($U_{вх} = 0$) ток I_K исчезает, однако индуктивный ток I_L в обмотке W_1 не может мгновенно исчезнуть, так как в сердечнике запасена магнитная энергия.

Поэтому на обмотках трансформатора возникают ЭДС обратной полярности (на рисунке даны в скобках), стремящиеся поддержать прежнее направление тока I_L (см. п. 4.6.3).

Это приводит к тому, что на коллекторе закрытого транзистора появляется выброс напряжения U_B , поскольку полярность ЭДС, наводимой в обмотке W_1 , такова, что напряжение питания суммируется с выбросом:

$$U_{кз} = U_B + E. \quad (4.32)$$

Пропорциональный выброс появится и во вторичной обмотке. Причем важно отметить, что, поскольку за время импульса и паузы изменения индукции будут одинаковы, то и площади выходных импульсов S_1 и выброса S_2 будут равными. Это лишний раз указывает на то, что выходное напряжение не имеет постоянной составляющей, то есть трансформатор не пропускает постоянного тока.

Диод D_0 при такой полярности выброса начинает проводить и подключает R_0 к обмотке W_1 . Эквивалентная схема замещения трансформатора на стадии восстановления исходного состояния сердечника представляет собой (см. п. 4.6.5) параллельный контур из индуктивности L и паразитной межвитковой емкости C_{11} первичной обмотки, шунтированный сопротивлением $R'' = R_H' \parallel R_0$.

Переходный процесс в такой схеме (см. п. 4.6.5) может быть как колебательным (при $R'' > R_{кр}$), так и апериодическим ($R'' < R_{кр}$). Диод D_0 , подключающий резистор R_0 к R_H' и тем самым уменьшающий при правильном расчете сопротивление нагрузки до критического, обеспечивает «успокоение» (демпфирование) колебаний. Поэтому цепочку D_0 и R_0 иногда называют демпфирующей.

4.7.4. Параметры выброса

При правильном выборе величины R_0 выброс на коллекторе транзистора имеет характер, близкий к экспоненциальному с постоянной времени

$$\tau_B = L / R_H'' \approx L / R_O' \quad (4.33)$$

Амплитуда выброса при этом согласно (4.26):

$$U_B = I_L(t_H) R_H'' \approx I_L(t_H) R_O' \quad (4.34)$$

то есть зависит от последнего перед выключением значения тока намагничивания I_L и сопротивления R_O .

Учитывая, что выброс определяет, как следует из уравнения (4.32), максимальное напряжение $U_{кэ}$ непроводящего транзистора, которое должно быть меньше допустимого, можно указать другое назначение цепочки D_O и R_O . Она обеспечивает *защиту транзистора от пробоя*, если выбрать

$$R_O < (U_{кэдоп} - E) / I_L(t_H). \quad (4.35)$$

Но с другой стороны, следует также иметь в виду, что уменьшение R_O приводит к затягиванию выброса, то есть сердечник может не успеть вернуться в исходное состояние (точка «8» на частном цикле рис. 4.16). Для полного восстановления магнитного режима сердечника необходимо, чтобы длительность выброса ($\approx 3\tau$) была меньше длительности паузы t_H между импульсами. Поэтому величина R_O ограничена и снизу:

$$R_O > 3L / t_H \quad (4.36)$$

иначе сердечник перейдет на новый частный цикл в области насыщения, где появятся недопустимые искажения импульса.

4.7.5. Оценка числа витков трансформатора

Чем больше число витков, тем меньше ток намагничивания и лучше для ключа. Но габариты трансформатора и его трудоемкость увеличиваются. Поэтому при оценке числа витков идут на компромисс, задаваясь максимально возможным током I_L , например, равным току нагрузки I_2' . Тогда из формулы (4.16) можно найти требуемое значение индуктивности:

$$L = U_1 t_H / I_L = U_1 t_H / I_2' \quad (4.37)$$

Зная значение L , далее из формулы (4.15) находят число витков первичной обмотки W_1 :

Найденное значение нужно проверить, чтобы оно не было меньше минимально возможного числа витков, ограниченного основным уравнением импульсного трансформатора (4.10). Для этого, считая для однополярного сигнала (см. п. 4.5.1) $\Delta B = \Delta B_{\text{МАКС}} = (B_S - B_R)$, находят:

$$W_1 = \sqrt{\frac{L \cdot \ell_{CP}}{\mu_0 \cdot \mu \cdot S}}, \quad (4.38)$$

$$W_{1\text{МИН}} = E_K \cdot t_{\text{И}} / S (B_S - B_R). \quad (4.39)$$

Из двух найденных значений W_1 и $W_{1\text{МИН}}$ выбирают большее.

Число витков W_2 , которое определяет необходимый коэффициент трансформации, находят из уравнения (4.12):

$$W_2 = W_1 \cdot U_2 / U_1. \quad (4.40)$$

4.7.6. Расчет сопротивлений ключа

Расчет величин R_B и R_{CM} производится, как в ключе с резистивной связью, из условий запираания и насыщения. Только при расчете R_B коллекторный ток в режиме насыщения принимают равным

$$I_{KH} = E \cdot t_{\text{И}} / L + I'_{2'}. \quad (4.41)$$

5

ОСНОВЫ СХЕМОТЕХНИКИ ИОУ

Микропроцессорные системы в ряде случаев получают от датчиков или из линии связи сигналы низкого уровня, которые нуждаются перед обработкой в предварительном усилении. В настоящее время для этой цели используются интегральные операционные усилители (ИОУ), выполненные в виде микросхем. Кроме того, ИОУ входят в состав множества других устройств, применяемых в вычислительной технике – генераторов (см. п. 6.7.2), цифро-аналоговых (см. п. 16.1.4) и аналого-цифровых (см. п. 18.5.1) преобразователей.

Чтобы на практике правильно применять ИОУ, необходимо знать общие особенности построения входных и выходных усилительных каскадов, их характеристики и типовые схемы включения ИОУ.

§5.1. Обратная связь в усилителях

5.1.1. Общие сведения об усилителях

Усилитель осуществляет преобразование маломощного информационного сигнала низкого уровня (напряжения или тока) в больший по мощности и по уровню сигнал. Главным параметром усилителя поэтому является его коэффициент усиления K , чаще всего определяемый как *отношение приращений* выходного и входного сигналов и показывающий во сколько раз *изменение* выходного сигнала $\Delta Y_{\text{Вых}}$ больше *изменения* входного $\Delta X_{\text{Вх}}$:

$$K = \Delta Y_{\text{Вых}} / \Delta X_{\text{Вх}} \quad (5.1)$$

В дальнейшем входной и выходной сигнал в большинстве случаев будут представлены напряжениями. Кроме того, знак приращения Δ для простоты записи будем опускать.

По частотным свойствам усилители делятся на усилители постоянного тока (УПТ), способные усиливать сигналы с «нулевой» час-

тотой (рис. 5.1) и усилители переменного тока. Среди последних в свою очередь условно выделяют усилители низкой частоты (УНЧ – рис. 5.2), усилители высокой частоты (УВЧ – рис. 5.2), избирательные (резонансные) усилители (рис. 5.3) и др. [18].

Основное внимание в пособии уделено более универсальным УПТ, потому что усилители переменного тока могут быть представлены, как частный случай УПТ (см. §6.5) с разделительными элементами, не пропускающими постоянный ток, такими как конденсаторы или трансформаторы.



Рис. 5.1

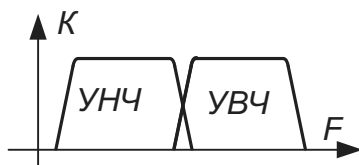


Рис. 5.2

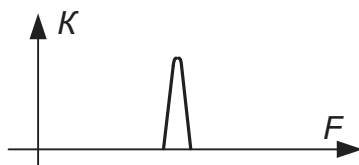


Рис. 5.3

В упрощенном виде усилитель, включаемый между источником сигнала и нагрузкой (рис. 5.4), представляется как устройство, имеющее две пары зажимов – входных и выходных.

Способность входной цепи усилителя нагружать источник сигнала отражается на схеме входным сопротивлением усилителя $R_{ВХ}$. Со стороны выходных зажимов усилитель чаще всего представляется источником ЭДС $E_{ВЫХ}$, зависящей от величины $U_{ВХ}$ и внутренним сопротивлением усилителя $R_{ВЫХ}$.

В редких случаях выходная цепь усилителя может быть изображена в виде источника тока $I_{ВЫХ}$, зависящего от входного сигнала, с параллельно включенным сопротивлением $R_{ВЫХ}$.

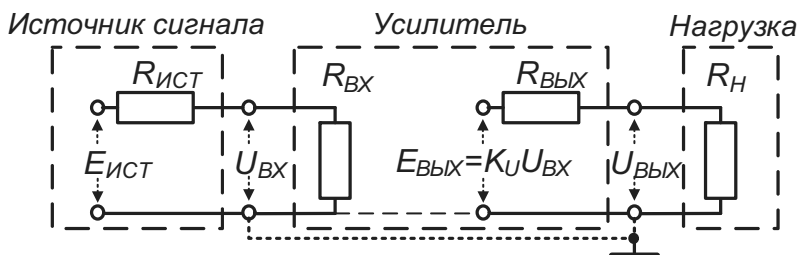


Рис. 5.4

Как видно из рис. 5.4, при усилении действуют два основных контура передачи электроэнергии: от источника сигнала к входной цепи усилителя и от усилителя к нагрузке. При этом в зависимости от соотношения сопротивлений каждого контура можно выделить (часто условно) три режима работы по входу и выходу:

- токовый (при $R_{\text{и}} \gg R_{\text{вх}}, R_{\text{вых}} \gg R_{\text{н}}$),
- потенциальный ($R_{\text{и}} \ll R_{\text{вх}}, R_{\text{вых}} \ll R_{\text{н}}$),
- мощностной ($R_{\text{и}} \approx R_{\text{вх}}, R_{\text{вых}} \approx R_{\text{н}}$).

Соответственно различают усилители тока, напряжения, мощности и комбинированные усилители.

В реальном усилителе входная и выходная цепь взаимосвязаны. В простейшем случае они имеют один объединенный (общий) вывод, показанный на рис. 5.4. пунктиром. Наличие подобной общей точки у входного и выходного сигналов облегчает построение многокаскадных устройств обработки информации.

В более сложных случаях взаимосвязь организуют путем передачи части энергии с выхода на вход, в виде так называемой обратной связи (ОС). ОС играет очень большую роль в работе усилителя и существенно влияет на его характеристики. Передачу энергии при введении ОС в усилителе осуществляет четырехполюсник, выполняемый чаще всего на основе резисторов (но не обязательно).

5.1.2. Разновидности обратной связи

В усилителях используют два основных вида ОС – по току и по напряжению, в зависимости от рода сигнала на входе звена ОС. С учетом двойственности режима работы входной и выходной цепей можно реализовать четыре основные схемы введения ОС в усилителях (рис. 5.5).

Нетрудно видеть, что в схемах «А» и «В» в цепь ОС поступает напряжение $U_{\text{вых}}$, поэтому говорят, что действует *ОС по напряжению*. В схемах «Б» и «Г» в цепь ОС поступает сигнал с эталонного резистора $R_{\text{э}}$, выполняющего роль датчика тока. Этот сигнал ($I_{\text{вых}} R_{\text{э}}$) пропорционален выходному току. Поэтому говорят, что в схеме действует *ОС по току*.

С другой стороны схемы различаются способами введения ОС во входную цепь. Так схемы «А» и «Б» используют потенциальный режим объединения сигналов – суммируются напряжения $U_{\text{вх}}$ и $U_{\text{ОС}}$, соединенные последовательно. Отсюда следует термин – *последовательная ОС*.

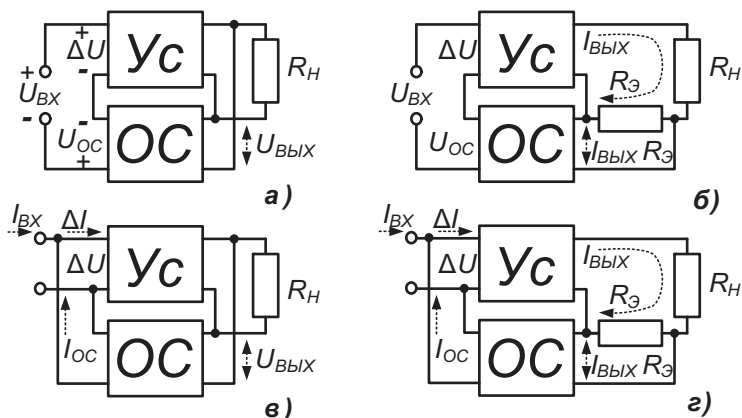


Рис. 5.5

В схемах «В» и «Г» суммируются токи, причем источники сигналов включены параллельно. Поэтому такие схемы принято называть усилителями с *параллельной* ОС.

Очевидно, что действие ОС будет зависеть от полярности сигнала на выходе звена ОС. Если сигнал ОС добавляется ко входному и происходит увеличение сигнала ΔU , поступающего на усилитель, то говорят о *положительной* ОС (ПОС). В противном случае, когда сигнал ОС вычитается из входного, в схеме действует *отрицательная* ОС (ООС).

5.1.3. Коэффициент усиления усилителя с ПОС

Рассмотрим влияние ОС на коэффициент усиления на примере схемы «А» рис. 5.5, предположив, что в ней (согласно оставленной полярности напряжений) действует *положительная* ОС, то есть сигнал ОС добавляется к входному

$$\Delta U = U_{ВХ} + U_{ОС} \quad (5.2)$$

В этой формуле ΔU и $U_{ОС}$ можно выразить через значение $U_{ВЫХ}$, чтобы найти зависимость выходного напряжения от входного. Используя формулу (5.1) и учитывая, что входным напряжением для усилителя является ΔU , можно записать для нашего случая коэффициент усиления усилителя без ОС:

$$K = U_{ВЫХ} / \Delta U \quad (5.3)$$

и найти из этой формулы

$$\Delta U = U_{\text{ВЫХ}} / K. \quad (5.4)$$

Звено ОС имеет свой коэффициент передачи, равный отношению напряжений:

$$\beta = U_{\text{ОС}} / U_{\text{ВЫХ}} \quad (5.5).$$

Отсюда

$$U_{\text{ОС}} = \beta \cdot U_{\text{ВЫХ}} \quad (5.6)$$

Подставим в (5.2) значения ΔU из (5.4) и $U_{\text{ОС}}$ из (5.6) и решим полученное уравнение относительно $U_{\text{ВЫХ}}$. Затем результат поделим на $U_{\text{ВХ}}$ и найдем коэффициент усиления схемы с учетом действия положительной ОС:

$$K_{\text{ПОС}} = \frac{U_{\text{ВЫХ}}}{U_{\text{ВХ}}} = \frac{K}{1 - K\beta}. \quad (5.7)$$

Произведение $K\beta$, характеризующее *глубину* ОС, существенно влияет на усиление. При $K\beta$, отличном от нуля, значение $K_{\text{ПОС}}$, как видно из формулы (5.7), получается большим K . Причем по мере приближения величины $K\beta$ к единице, усиление резко возрастает и при $K\beta = 1$ коэффициент $K_{\text{ПОС}}$ должен быть бесконечным, но при этом усилитель превращается в *генератор* (см. п. 6.7.2) и перестает усиливать сигнал.

Поэтому, несмотря на то, что при введении ПОС коэффициент усиления увеличивается, положительная ОС имеет ограниченное применение в линейных схемах из-за присущих ей неустойчивости, склонности к неустойчивой работе и самопроизвольной генерации колебаний. Указанное, впрочем, не исключает использование ПОС в генераторах, триггерах (см. п. 12.1.1) и пороговых схемах (см. п. 17.2.1).

В усилительных схемах с линейными характеристиками, применяется в основном ООС, имеющая ряд ценных свойств и, в первую очередь, повышенную стабильность параметров.

5.1.4. Коэффициент усиления усилителя с ООС

Коэффициент усиления с учетом действия ООС, которая отличается от положительной ОС полярностью $U_{\text{ОС}}$, можно получить формально, сменив знак плюс на минус при β в формуле (5.7). Тогда

$$K_{\text{ООС}} = \frac{U_{\text{ВЫХ}}}{U_{\text{ВХ}}} = \frac{K}{1 + K\beta}. \quad (5.8)$$

Вследствие действия ООС коэффициент усиления $K_{оос}$, как видно из формулы (5.8), уменьшается, так как знаменатель дроби больше единицы, и $K_{оос}$ таким образом, всегда *меньше* K . Но уменьшается и его неустойчивость, что особенно заметно при $K\beta \gg 1$. Чтобы показать это, используют другую форму записи для $K_{оос}$, которую получают, умножив и поделив значение $K_{оос}$ на β :

$$K_{оос} = \frac{U_{ввх}}{U_{вх}} = \frac{K}{1 + K\beta} = \frac{1}{\beta} \cdot \frac{K\beta}{1 + K\beta}. \quad (5.9)$$

Из этой формулы видно, что при $K\beta \gg 1$ дробь практически превращается в единицу и коэффициент $K_{оос}$, рассчитываемый по упрощенной формуле

$$K_{оос} \approx 1 / \beta, \quad (5.10)$$

оказывается практически не зависящим от свойств усилителя (значения K), а определяется величиной β . А коэффициент β сравнительно легко сделать неизменным, выполнив, например, звено ООС из высокоточных и термостабильных резисторов.

5.1.5. Общие свойства отрицательной ОС

Не останавливаясь на подробном изучении влияния ООС на характеристики усилителя, что при необходимости можно найти в литературе [18], отметим некоторые ценные свойства ООС.

В первую очередь ООС влияет на входное и выходное сопротивления усилителя. В дальнейшем будет показано, что наличие последовательной ООС приводит к увеличению $R_{вх}$ (см. п. 5.5.3) а параллельной ОС уменьшает его (см. п. 5.6.3). В результате входной потенциальный контур (схемы «А» и «Б» на рис. 5.5) переводится в режим, близкий к холостому ходу, а входной токовый (схемы «В» и «Г») – к короткому замыканию, что способствует в обоих случаях более точной передаче информационного сигнала во входной цепи.

Аналогичное улучшение происходит и в выходном контуре усилителя. Выходное сопротивление усилителя с ООС по напряжению уменьшается [18], то есть выходная цепь приближается по своим свойствам к идеализированному источнику напряжения. А для случаев ООС по току значение $R_{ввх}$ наоборот увеличивается и усилитель можно рассматривать со стороны выходных зажимов как идеализированный источник тока.

При введении в усилитель ООС происходит также уменьшение в $(1 + K\beta)$ раз нелинейных, частотных и фазовых искажений [18].

5.1.6. Самовозбуждение усилителя

Важно также отметить, что существует опасность самовозбуждения усилителя из-за превращения при определенных условиях отрицательной ОС в положительную. Одной из причин этого является инерционность транзисторов усилителя, которая приводит возникновению дополнительного фазового сдвига между входным и выходным сигналами.

Для пояснения этого явления рассмотрим векторы напряжений $U_{вх}$ и $U_{ос}$, которые для случая ООС должны вычитаться и поэтому на рис. 5.6 они направлены в разные стороны.

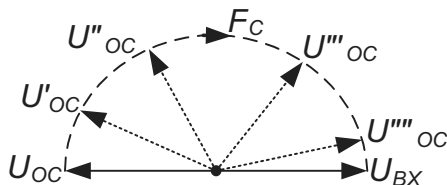


Рис. 5.6

Однако транзисторы в усилителе являются инерционными элементами и на высоких частотах вносят отстающий фазовый сдвиг, сказывающийся на отклонении $U'_{ос}$ от первоначального вектора $U_{ос}$. Чем больше частота сигнала F_c , тем больше угол, на который отстают сигналы ОС, и соответствующий вектор $U''_{ос}$ будет по мере увеличения частоты приближаться к вектору $U_{вх}$.

И на какой-то достаточно высокой частоте (сотни $\kappa\Gamma\text{ц}$ или единицы МГц) сигнал $U'''_{ос}$, вместо того чтобы вычитаться из входного, станет суммироваться с ним. Это означает, что в усилителе будет действовать положительная ОС на высокой частоте, которая, в принципе, может превратить усилитель в генератор самопроизвольных колебаний (см. п. 6.7.2), искажающих информационный сигнал.

5.1.7. Методы срыва генерации

Для устранения (срыва) такой генерации применяют различные RC -цепочки, включаемые в цепь усиления. Чаще всего применяют цепочки одного из двух типов.

Дифференцирующие цепи служат для создания *опережающего фазового сдвига*, компенсирующего отстающий сдвиг транзисторов. Однако при этом не всегда удастся правильно подобрать параметры такой цепочки, так как свойства транзисторов внутри усилителя бывают неизвестны.

Большее применение получили цепочки *интегрирующего* типа, которые, не внося существенного дополнительного фазового сдвига, *уменьшают величину $K\beta$* на высоких частотах до значения, меньшего

единицы (см. формулу (5.7)), и устраняют тем самым возможность генерации.

Поэтому при использовании корректирующих цепочек следует помнить, что они в большинстве случаев ухудшают частотные свойства усилителей (см. п. 5.4.6) и лучше бы обойтись без них, особенно при малой глубине ООС, когда опасность самовозбуждения невелика. На практике для каждого ИОУ в справочниках приведены стандартные корректирующие цепочки, зависящие от особенностей микросхемы и используемой глубины ООС.

5.1.8. Основные требования к усилительным каскадам

Для обеспечения большого коэффициента усиления обычно используют усилитель, состоящий из отдельных каскадов (рис. 5.7), выполненных, например, на одиночных транзисторах (см. п. 5.2.1). Причем требования к отдельным каскадам зависят от расположения каскада в усилительной цепочке. Например, самый первый (входной) каскад определяет входное сопротивление усилителя, а последний каскад – его выходное сопротивление.

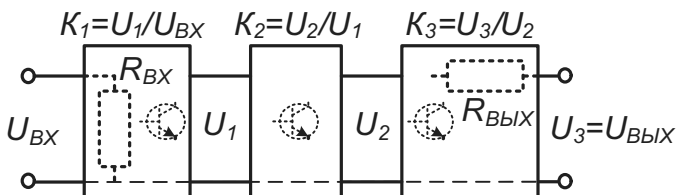


Рис. 5.7

Главный параметр многокаскадного усилителя – коэффициент усиления, легко найти, учитывая, что выходное напряжение отдельного каскада больше входного в коэффициент усиления раз. Поэтому можно записать

$$U_{BЫX} = U_3 = K_3 U_2 = K_3 K_2 U_1 = K_3 K_2 K_1 U_{BX} \quad (5.11)$$

и, поделив полученное выражение на U_{BX} , найти

$$K = U_{BЫX} / U_{BX} = K_3 K_2 K_1. \quad (5.12)$$

То есть коэффициент усиления многокаскадного усилителя равен произведению коэффициентов усиления отдельных каскадов.

§5.2. Параметры усилительного каскада

5.2.1. Схемы включения транзисторов

Существуют три основных схемы включения биполярного транзистора – с общей базой (ОБ), с общим эмиттером (ОЭ) и с общим коллектором (ОК). Они отличаются электродом, подключенным к общей точке, а также входными и выходными токами.

Схема с ОБ (рис. 5.8) имеет самые лучшие частотные характеристики, но меньший коэффициент усиления, и поэтому ее используют в основном в специализированных высокочастотных усилителях, которые здесь не рассматриваются.

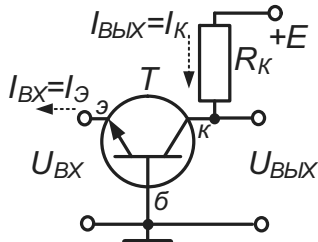


Рис. 5.8

Ниже описаны основные свойства схемы с ОЭ (рис. 5.9), которая отличается наибольшим коэффициентом усиления по напряжению, и схемы с ОК (рис. 5.11), которая хотя и имеет коэффициент, близкий к единице, но обладает большим входным и малым выходным сопротивлениями, что оказывается полезным во многих случаях.

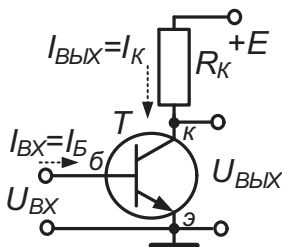


Рис. 5.9

Полезно также помнить, что схема с ОЭ, как и логический ключ с ОЭ, инвертирует фазу входного сигнала, а схема с ОК (эмиттерный повторитель) – нет (см. п. 2.3.2).

В усилителях могут также применяться и полевые транзисторы, схемы включения которых подобны указанным с точностью до именования электродов. Но в данном пособии каскады на полевых транзисторах для простоты не рассматриваются, потому что практические выводы, сделанные ниже для каскадов на биполярных транзисторах, в основном (кроме, естественно, входного сопротивления) могут быть расширены и на каскады на полевых.

5.2.2. Эквивалентная схема усилительного каскада с ОЭ

Транзистор имеет нелинейные характеристики, поэтому для практических расчетов его стараются заменить линейными элементами.

В усилительном режиме изменения токов и напряжений обычно невелики, поэтому при анализе пользуются малосигнальными схемами замещения, одну из которых («Т-образную» [28]) используем для перехода от принципиальной схемы с ОЭ (рис. 5.9) к эквивалентной схеме усилительного каскада с ОЭ, представленной на рис. 5.10.

Усилительные свойства транзистора отражены на рис. 5.10 наличием источника тока, величина которого в B раз больше базового тока I_B , где B – коэффициент усиления транзистора на низких частотах.

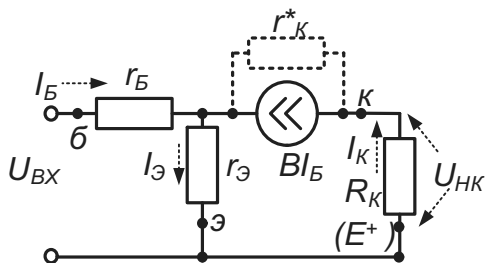


Рис. 5.10

Когда параметры транзистора определяются как у четырехплосника, вместо коэффициента усиления B используют коэффициент передачи $h_{21э}$ [28].

Инерционностью транзистора и эффектом модуляции ширины базы (то есть выходным сопротивлением транзистора) для простоты пренебрежем.

Кроме того, в схеме замещения транзистора учитывается объемное сопротивление базового слоя r_B и динамическое сопротивление эмиттерного перехода $r_э$, которое зависит от тока $I_э$ и рассчитывается по формуле [28]:

$$r_э = \varphi_T / I_э, \quad (5.13)$$

где φ_T – температурный потенциал, при комнатной температуре примерно равный 25 мВ.

Для простоты в эквивалентной схеме не будем учитывать нагрузку каскада и цепи, задающие режим работы, то есть кроме транзистора в эквивалентной схеме каскада будет только внешний резистор R_K . Отметим также, что для переменных составляющих постоянное напряжение питания E (на рис. 5.10 показано в скобках) представляется перемычкой.

5.2.3. Входное сопротивление схемы с ОЭ

Обычно при определении сопротивления используют закон Ома, который для входного контура (рис. 5.10) можно записать в виде

$$R_{BX} = U_{BX} / I_{BX} = (I_B r_B + I_3 r_3) / I_B = r_B + r_3 I_3 / I_B. \quad (5.14)$$

Считая, что

$$I_3 / I_B \approx B + 1, \quad (5.15)$$

хотя эта оценка несколько завышена [28], получим:

$$R_{BX} \approx r_B + r_3 (B + 1). \quad (5.16)$$

Это компактное уравнение удобно для запоминания и практических оценок. Например, пусть $r_B = 100 \text{ Ом}$, а $B = 50$. Для расчета r_3 по формуле (5.13) примем ток $I_3 = 1 \text{ мА}$. Тогда $r_3 = \varphi_T / I_3 = 25 \cdot 10^{-3} / 10^{-3} = 25 \text{ Ом}$ и можно оценить $R_{BX} = 100 + 25 \cdot 51 = 1375 \text{ Ом}$.

Этот числовой пример позволяет сделать вывод – входное сопротивление схемы с ОЭ имеет величину около тысячи Ом, зависит от величины r_3 и, в конечном счете, от тока I_3 .

5.2.4. Коэффициент усиления напряжения для схемы с ОЭ

Он определяется, как известно, отношением выходного сигнала к входному. Для схемы с ОЭ (рис. 5.10) $U_{ВЫХ} = U_{HK} = I_K R_{HK}$, а $U_{ВХ} = I_B R_{BX}$, тогда

$$K_{OЭ} = U_{HK} / U_{BX} = I_K R_{HK} / I_B R_{BX} = B \cdot R_{HK} / R_{BX}. \quad (5.17)$$

Оценим численно величину $K_{OЭ}$ для схемы с ОЭ, используя из предыдущего примера значения $B = 50$ и $R_{BX} = 1375 \text{ Ом}$. Тогда, считая $R_K = 10 \text{ кОм}$, получим $K_{OЭ} = 50 \cdot 10000 / 1375 \approx 364$.

Вывод: одиночный каскад с ОЭ может усиливать сигнал в десятки и даже в сотни раз, причем коэффициент усиления зависит от сопротивления нагрузки. Но с ее увеличением возрастает роль не учитываемого в данном случае выходного сопротивления транзистора, которое приводит к уменьшению коэффициента усиления каскада по току. Поэтому существует [28] предельно возможный коэффициент усиления, однако он обеспечивается за счет существенного усложнения принципиальной схемы одиночного каскада.

5.2.5. Выходное сопротивление схемы с ОЭ

При его определении считают, что $U_{BX} = 0$, а значит и $I_B = 0$. Тогда источник тока в схеме рис. 5.10 будет нулевым и эквивалентным разрыву. Из этих рассуждений следует [28], что выходное сопротивление

$$R_{\text{ВыХоз}} \approx R_K \quad (5.18)$$

В этой формуле не учтено шунтирующее влияние выходного сопротивления транзистора, но оно в реальных случаях незначительно.

Вывод: выходное сопротивление каскада с ОЭ зависит от внешнего сопротивления R_K и имеет поэтому в большинстве случаев ее значительную величину, что является одним из недостатков такой схемы.

5.2.6. Схема и входное сопротивление эмиттерного повторителя

Каскад с ОК (рис. 5.11), чаще называемый эмиттерным повторителем (ЭП), можно заменить для малых приращений сигналов его линейной эквивалентной схемой. В этой схеме (рис. 5.12), как и прежде, напряжение питания E для приращений представляется перемычкой, поэтому коллектор транзистора является общим электродом для входного и выходного сигналов, что определяет название каскада – с ОК.

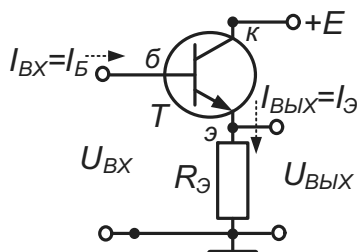


Рис. 5.11

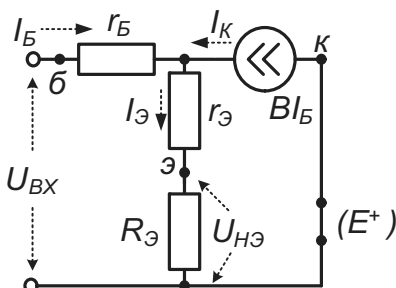


Рис. 5.12

По аналогии с п. 5.2.3 можно записать закон Ома для входного контура рис. 5.12 и, учитывая уравнение (5.15) для коэффициента усиления по току, получить

$$R_{\text{ВХэл}} = r_B + (R_Э + r_Э) I_Э / I_Б \approx (R_Э + r_Э)(B + 1). \quad (5.19)$$

Численная оценка входного сопротивления для ЭП при $R_Э = 10 \text{ кОм}$, $r_Э = 25 \text{ Ом}$, $B = 50$ дает значение $R_{\text{ВХэл}} = (10000 + 25) 51 \approx 511 \text{ кОм}$.

Отсюда следует, что входное сопротивление ЭП имеет величину в сотни тысяч Ом (то есть значительно больше значения $R_{\text{ВХоз}}$ для схемы с ОЭ), и зависит от величины $R_Э$.

Последняя зависимость может натолкнуть на мысль, что, увеличивая бесконечно $R_Э$, можно получить огромное $R_{\text{ВХэл}}$. Однако

при таком рассуждении не учитывается, что с ростом $R_э$ одновременно падает коэффициент усиления каскада по току, и поэтому существует [28] некоторое предельное значение входного сопротивления ЭП, равное примерно выходному сопротивлению транзистора r_k .

5.2.7. Коэффициент усиления ЭП по напряжению

Определяется аналогично коэффициенту для схемы с ОЭ (см. п. 5.2.4) с учетом того, что в схеме рис. 5.12 напряжения опять определяются по закону Ома, то есть $U_{нэ} = I_э \cdot R_э$ и $U_{вх} = I_б \cdot R_{вх}$. Далее, учитывая коэффициент усиления ЭП по току из уравнения (5.15), можно получить

$$K_{эп} = U_{нэ} / U_{вх} = I_э \cdot R_э / I_б \cdot R_{вх} = (B + 1) \cdot R_э / R_{вх} \quad (5.20)$$

Подставляя приближенное значение $R_{вх}$ из (5.19) и проводя простейшие преобразования, запишем коэффициент усиления для ЭП в виде

$$K_{эп} \approx 1 / (1 + r_э / R_э). \quad (5.21)$$

Разлагая далее дробь в степенной ряд и ограничиваясь линейным членом, можно получить приближенное, но достаточно верное соотношение:

$$K_{эп} \approx 1 - r_э / R_э. \quad (5.22)$$

Из этого уравнения следует, что коэффициент передачи эмиттерного повторителя весьма близок к единице, что оправдывает применение термина «повторитель» вместо термина «схема с ОК».

Например, при $r_э = 25 \text{ Ом}$, $R_э = 10 \text{ кОм}$, получаем $K_{эп} = 1 - 25/10^4 = 0,9975$. Или, иными словами, погрешность передачи сигнала в данном примере не превышает 0,25%. Причем при необходимости эту погрешность можно уменьшить за счет увеличения $R_э$, но в любом случае коэффициент $K_{эп}$ останется меньшим единицы.

Поэтому эмиттерный повторитель применяют в основном как согласующий каскад с большим входным сопротивлением и, как показано ниже, малым выходным сопротивлением.

5.2.8. Выходное сопротивление ЭП

При определении $R_{выхэп}$ будем полагать входной сигнал отсутствующим, однако влияние источника сигнала учтем, введя в эквивалентную схему (рис. 5.13) его внутреннее сопротивление $R_{и}$.

Мысленно подключим к выходным зажимам эквивалентной схемы пробный источник напряжения E_0 (на рис. 5.13 показан пунктиром). Тогда, используя закон Ома, можно записать:

$$R_{\text{БЫХэл}} = E_0 / I_0 = E_0 / (I_{R_3} + I_3). \quad (5.23)$$

Далее, поделив числитель и знаменатель дроби на E_0 , выражение (5.22) можно представить в виде:

$$R_{\text{БЫХэл}} = 1 / (I_{R_3} / E_0 + I_3 / E_0) = 1 / (1 / R_3 + 1 / R_{T3}). \quad (5.24)$$

Таким образом, видно, что выходное сопротивление ЭП равно параллельному соединению сопротивления R_3 и выходного сопротивления транзистора со стороны эмиттера R_{T3} , причем в большинстве практических случаев $R_3 \gg R_{T3}$.

Поэтому

$$R_{\text{БЫХэл}} \approx R_{T3} = E_0 / I_3. \quad (5.25)$$

Записывая напряжение E_0 через падения напряжения на сопротивлениях r_B , r_3 и $R_{\text{И}}$

$$E_0 = I_3 r_3 + I_B (r_B + R_{\text{И}}) \quad (5.26)$$

и, поделив последнее выражение на I_3 , определим:

$$R_{\text{БЫХэл}} \approx r_3 + (r_B + R_{\text{И}}) / (B + 1). \quad (5.27)$$

Оценивая численно выходное сопротивление эмиттерного повторителя при $R_{\text{И}} = 1 \text{ кОм}$ и данных из предыдущих примеров, можно получить $R_{\text{БЫХэл}} = 25 + (100 + 1000) / 51 = 47 \text{ Ом}$.

Вывод: сопротивление $R_{\text{БЫХэл}}$ имеет величину около нескольких десятков Ом, зависит от сопротивлений r_3 и $R_{\text{И}}$ и значительно меньше выходного сопротивления каскада с ОЭ $R_{\text{БЫХоз}}$.

§5.3. Простейшие УПТ

5.3.1. УПТ на одном транзисторе

При усилении сигналов постоянного тока возникают некоторые проблемы, которые мы рассмотрим на примере одиночного каскада на биполярном транзисторе с ОЭ (рис. 5.14), имеющего при минимуме элементов все основные черты УПТ.

Каскад выполнен без реактивных элементов, таких как конденсатор или трансформатор, которые, как известно, не пропускают постоянную составляющую, и имеет ООС параллельного вида через резисторы $R_{\text{ОС}}$ и R_T .

Найдем выходное напряжение, записав значения токов в схеме:

$$I_{\text{ОС}} = I_T + I_B, \quad (5.28)$$

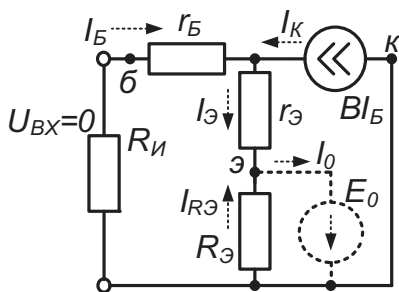


Рис. 5.13

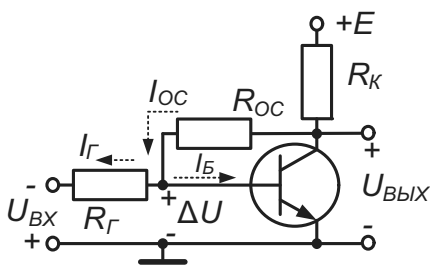


Рис. 5.14

где

$$I_{Г} = (U_{BX} + \Delta U) / R_{Г} \quad (5.29)$$

$$I_{ОС} = (U_{ВЫХ} - \Delta U) / R_{ОС} \quad (5.30)$$

Отсюда

$$U_{ВЫХ} = U_{BX} R_{ОС} / R_{Г} + \Delta U (R_{ОС} / R_{Г} + 1) + I_{Б} R_{ОС} \quad (5.31)$$

Из выражения видно, что выходной сигнал каскада определяется тремя слагаемыми. Первое из них связано с входным напряжением и является полезным. Второе и третье слагаемые не имеют информационной ценности, суммарное значение их обычно называют *нулевым уровнем УПТ*, поскольку определяют в процессе настройки при $U_{BX} = 0$ и обозначают $U_{ВЫХ0}$.

5.3.2. Дрейф нулевого уровня УПТ

Наличие нулевого уровня следует учитывать на практике, например, вычитая при обработке из выходного напряжения соответствующее постоянное значение. Однако такая операция предполагает неизменность напряжения $U_{ВЫХ0}$, зависящего согласно (5.31) от напряжения ΔU (равном в нашем случае $U_{ЭБ}$) и тока $I_{Б}$.

Вместе с тем, известно, что напряжение на прямосмещенном эмиттерном переходе $U_{ЭБ}$ и базовый ток $I_{Б}$ изменяются с температурой. Это приводит к определенной погрешности за счет медленного температурного дрейфа выходного напряжения, который часто называют дрейфом нулевого уровня, так как оценка его, как отмечалось, происходит при $U_{BX} = 0$.

Дрейф нуля, не позволяющий отличить изменения входного сигнала от паразитных составляющих, может достигать существенных значений.

Например, температурный коэффициент напряжения $U_{БЭ}$ может достигать $-2..3$ мВ на градус [28]. Тогда в температурном диапазоне от 0 до 50 градусов напряжение ΔU будет изменяться более чем на 100 мВ, то есть в ряде случаев может стать сравнимым с входным сигналом. Поэтому одной из основных задач при построении УПТ является снижение нулевого уровня и его температурного дрейфа.

Как видно из уравнения (5.31), токовую составляющую можно уменьшить за счет уменьшения резистора R_{OC} или тока I_B . Оба варианта имеют свои ограничения. Соображения по выбору R_{OC} изложены в п. 5.6.4, 5.6.6 и 5.7.3. Что касается тока I_B , то он определяется свойствами транзистора и в первую очередь *нелинейно* зависит от коэффициента усиления по току, так как $I_B = I_K / \beta$.

Для радикального уменьшения входного тока вместо биполярных используют полевые транзисторы, ток затвора которых значительно меньше базового тока.

Ниже рассмотрены меры воздействия на второе (потенциальное) слагаемое уравнения (5.31), которые более разнообразны благодаря практически *линейной* зависимости напряжения $U_{БЭ}$ от температуры, свойственной прямосмещенным переходам.

5.3.3. Усилительный каскад с термокомпенсацией

Для уменьшения нулевого уровня последовательно с эмиттерным переходом усилительного транзистора можно включить дополнительный прямосмещенный переход другого транзистора, но во встречном направлении (рис. 5.15). В этом случае

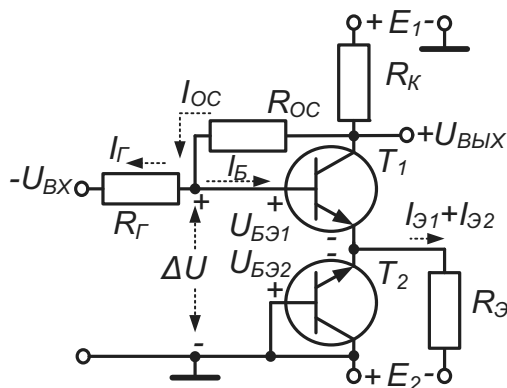


Рис. 5.15

$$\Delta U = U_{БЭ1} - U_{БЭ2} \quad (5.32)$$

кроме того, уменьшается (примерно на порядок) как величина ΔU , так и ее температурный дрейф.

Отметим, что резистор R_3 и источник E_2 обеспечивают прямое смещение компенсирующего перехода и создают путь для протекания встречных токов $I_{Э1}$ и $I_{Э2}$. Если бы этой цепи не было, то токи обоих транзисторов оказались нулевыми, а каскад – неработоспособным.

Однако полной термокомпенсации достигнуть в этой схеме не удастся. Дело в том, что транзисторы T_1 и T_2 работают в разных режимах. Напряжение на коллекторном переходе транзистора T_1 , равное $U_{ВЫХ}$ и зависящее от входного сигнала, изменяется в широких пределах, тогда как в компенсирующем транзисторе T_2 работает только эмиттерный переход, а $U_{КБ2} = 0$.

Поэтому, для улучшения свойств каскада нужно обеспечить транзисторам одинаковые режимы работы, например, подключив коллектор транзистора T_2 через резистор к источнику питания E_1 (см. далее).

5.3.4. Дифференциальный каскад

При симметричном включении (рис. 5.16) оба транзистора могут работать как усилительные, поэтому в схеме можно организовать дополнительные входной и выходной сигналы.

Такой каскад называют балансным или дифференциальным. Первое название отражает способность схемы к балансировке подобно измерительному мосту при $R_{K1} = R_{K2}$, так как в случае $U_{ВХ1} = U_{ВХ2}$ через идентичные транзисторы T_1 и T_2 будут проте-

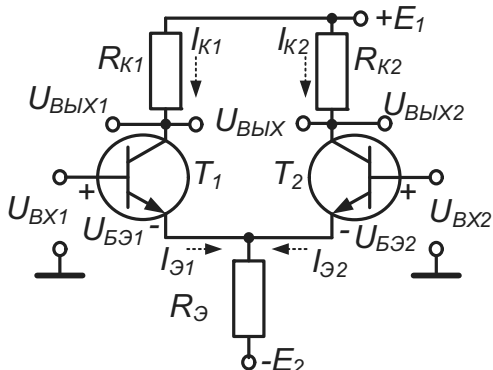


Рис. 5.16

каль одинаковые токи и $U_{\text{ВЫХ}} = U_{\text{ВЫХ1}} - U_{\text{ВЫХ2}}$ будет равно нулю. Если входные напряжения не равны, то токи транзисторов также будут различаться и возникнет разностное напряжение между выходами. Поэтому выходное напряжение каскада оказывается зависящим от разности $U_{\text{ВХ1}}$ и $U_{\text{ВХ2}}$. Отсюда следует другое название – разностный или дифференциальный каскад.

В дифференциальном каскаде дрейф нуля оказывается еще меньшим, чем в схеме с компенсацией, и дальнейшее его уменьшение происходит за счет схемотехнических и технологических мер, направленных на создание идентичных условий для обеих ветвей каскада. Даже небольшие различия, например, в температуре транзисторов, приводят к увеличению дрейфа нуля. Поэтому транзисторы T_1 и T_2 нужно пространственно сближать, уравнивая температуру, и для получения возможно близких характеристик изготавливать их в едином технологическом потоке. Эти требования хорошо согласуются с технологией изготовления полупроводниковых интегральных схем. Поэтому дифференциальный каскад получил широкое применение в более сложных усилителях и, в первую очередь, во входных цепях ИОУ.

Существуют и другие методы построения УПТ с малым дрейфом нулевого уровня (типа модулятор-демодулятор, со схемами коррекции нуля и т. д. [5]), на которых останавливаться не будем из-за достаточно редкого использования их на практике.

§5.4. Структура и параметры ИОУ

5.4.1. Описание иллюстративной схемы ИОУ

Не вдаваясь в подробную схемотехнику достаточно сложных современных усилителей, которую при необходимости можно найти в [2], рассмотрим очень упрощенную схему ИОУ, приведенную на рис. 5.17.

Для получения большого коэффициента усиления ИОУ выполняют в виде нескольких последовательно соединенных усилительных каскадов, при этом каждый каскад наряду с усилением должен обеспечивать выполнение некоторых дополнительных требований.

Так, *первый каскад* должен иметь возможно меньшие входные токи и дрейф нуля. Поэтому на входе ИОУ используется дифференциальный каскад (см. п. 5.3.4) на транзисторах T_1 , T_2 .

Выходной каскад, к которому подключена нагрузка, должен иметь свойства источника напряжения, то есть малое выходное сопротив-

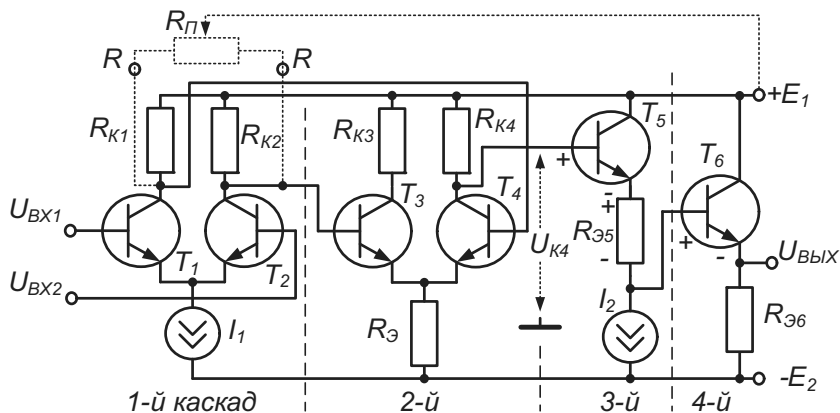


Рис. 5.17

ление и заданную нагрузочную способность по току. В наибольшей мере указанным требованиям удовлетворяет эмиттерный повторитель (транзистор T_6).

Промежуточные каскады обычно обеспечивают необходимый коэффициент усиления, причем второй каскад (в нашем случае на транзисторах T_3, T_4), работающий с достаточно малыми уровнями сигнала, выполняют тоже по дифференциальной схеме.

В одном из промежуточных каскадов (транзистор T_5) иногда осуществляют сдвиг уровня сигнала. Дело в том, что при $U_{Bx1} = U_{Bx2}$ выходной сигнал с целью согласования должен возрастать от каскада к каскаду и на выходе необходимо компенсировать этот уровень.

Один из способов вычитания, основанный на использовании источника тока (I_2), показан на рисунке 5.17, причем параметры цепочки сдвига выбирают так, чтобы при $U_{Bx1} = U_{Bx2}$ выполнялось равенство

$$U_{BvIX} = U_{K4} - U_{B35} - I_2 R_{35} - U_{B36} = 0. \quad (5.33)$$

Однако на производстве точно подстроить нулевой уровень достаточно сложно, и эта технологическая составляющая несколько увеличивает нулевой уровень ИОУ.

Кружком (клеммой) на рис. 5.17 помечены внешние выводы микросхемы, в частности предусмотрены выводы для балансировки ИОУ (показаны пунктиром), помеченные буквой R.

5.4.2. Условное графическое обозначение (УГО)

Реальные ИОУ отличаются большим разнообразием и сложностью схем на основе как биполярных, так и полевых транзисторов.

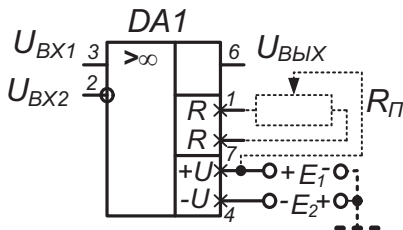


Рис. 5.18

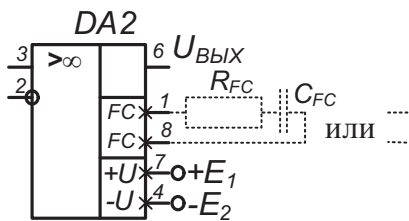


Рис. 5.19

Чтобы не загромождать принципиальные схемы вычислительных устройств, принято внутреннюю структуру ИОУ, как и других интегральных схем, не раскрывать, а изображать ИОУ в виде прямоугольника [3] (рис. 5.18 и 5.19) с двумя входами, выходом и вспомогательными выводами.

Отметим, что входы должны размещаться с левой стороны, а выходы – с правой, что позволяет при первом взгляде на принципиальную схему правильно понять назначение выводов. Неинформационные выводы (питания, балансировки и др.) необходимо помечать крестом.

Один из входов называют инвертирующим (его на УГО помечают кружком), так как при увеличении напряжения на нем напряжение на выходе будет уменьшаться (инвертировать знак или фазу). При использовании другого, неинвертирующего входа изменение сигнала на выходе будет совпадать по знаку с входным.

При подключении входов ИОУ к внешним элементам нужно обеспечивать по возможности одинаковые условия. Во всяком случае, совершенно недопустимо оставлять вход неподключенным, так как при этом базовый ток одного из транзисторов входного дифференциального каскада будет равен нулю, что нарушит работоспособность усилителя.

К вспомогательным выводам ИОУ (помимо выводов для источников питания) относятся балансировочные и корректирующие, к которым при необходимости подключают потенциометр для точной установки нуля (рис. 5.18) и *RC* цепочки для срыва высокочастотной генерации (рис. 5.19).

Выбор параметров корректирующих цепочек для разных типов ИОУ достаточно сложен, и в большинстве случаев используют стандартные цепочки и схемы включения, рекомендуемые в справочной литературе.

Некоторые микросхемы могут иметь внутренние элементы для срыва генерации. Однако поскольку действие корректирующих

цепочек ухудшает частотные свойства усилителя, в некоторых ИОУ внутренняя коррекция включается только при необходимости и обеспечивается при замыкании определенных выводов («1» и «8» на рис. 5.19).

5.4.3. Параметры входной цепи

Входное сопротивление ($R_{вх}$) ИОУ можно определять разными способами – между входами, между входом и общей точкой, для постоянного тока или переменного. В большинстве случаев при расчетах схем с ИОУ используют входное сопротивление переменному току (приращениям) со стороны одного из входов, когда другой заземлен. Величина $R_{вх}$ зависит в основном от свойств и типа транзисторов и находится в пределах 10 кОм...10 МОм. Меньшая цифра относится к ИОУ на биполярных транзисторах, большая – на полевых.

Напряжение смещения ($U_{см}$) характеризует разность напряжений $U_{бэ}$ транзисторов входного каскада и некомпенсированный сдвиг выходного напряжения (смотри уравнения (5.31), (5.33) и пояснения к ним), приведенный к входу. Численно величина $U_{см}$ равна напряжению, которое нужно подать между входами, чтобы $U_{вых}$ стало равным нулю. Точное значение $U_{см}$ для конкретного ИОУ можно определить только опытным путем. Поэтому в справочниках приводятся только предельные значения, равные примерно $\pm 5...10$ мВ для ИОУ на основе биполярных транзисторов, и $\pm 50...100$ мВ – на основе полевых. Специализированные *прецизионные* ИОУ имеют значительно меньшие значения $U_{см}$.

Средний входной ток ИОУ ($I_{вх}$) определяется как полусумма токов по обоим входам, и зависит, опять-таки, от типа транзисторов во входном каскаде. Входной ток у ИОУ на биполярных транзисторах имеет величину порядка долей мкА, у ИОУ на полевых транзисторах – в сотни раз меньше. Входные токи ИОУ, замыкаясь через сопротивления источников сигнала, создают падения напряжения, вызывающие дополнительный сдвиг нулевого уровня. В идеализированном случае, когда $I_{вх1} = I_{вх2}$, этот сдвиг можно скомпенсировать (см. п. 5.7.3). Однако в реальном ИОУ входные токи различны.

Разность токов по обоим входам ($I_p = I_{вх1} - I_{вх2}$) отражает степень неидентичности транзисторов входного каскада и существенно влияет при определенных условиях на погрешность ИОУ (см. п. 5.7.3).

Заканчивая перечисление основных входных параметров, укажем на существование для каждого типа ИОУ предельных значений синфазного и дифференциального входных сигналов, которые нельзя превышать. Под синфазным понимают напряжение на од-

ном из входов относительно общей точки, под дифференциальным – напряжение между входами.

5.4.4. Параметры выходной цепи

Максимальное выходное напряжение ИОУ ($U_{\text{ИОУм}}$) даже при очень большом входном сигнале, очевидно, не может превысить напряжения питания $E_{\text{пит}}$. Поэтому для большинства ИОУ общего назначения $U_{\text{ИОУм}} = E_{\text{пит}} - 1..2 \text{ В}$.

Некоторые ИОУ могут работать практически с полным размахом выходного напряжения от $-E_2$ до $+E_1$, что обозначают термином «Rail-To-Rail» («от рельса до рельса» – от отрицательного питания до положительного).

Максимальный выходной ток ($I_{\text{ИОУм}}$), обычно равный 5..20 мА, ограничивает снизу величину сопротивления нагрузки $R_{\text{нр}}$ подключаемого к выходу ИОУ. Нельзя превышать значение $I_{\text{ИОУм}}$ из-за опасности повредить ИОУ. Следует также отметить, что некоторые ИОУ имеют не только больший выходной ток, но и внутреннюю защиту от перегрузок по току.

Иногда вместо $I_{\text{ИОУм}}$ в справочниках приводят минимальное сопротивление нагрузки $R_{\text{нмин}} = U_{\text{ИОУм}} / I_{\text{ИОУм}}$. Из этого соотношения можно найти $I_{\text{ИОУм}}$.

Выходное сопротивление ($R_{\text{Вых}}$) зависит от схемы выходного каскада и равно примерно 10..200 Ом (как у эмиттерного или истокового повторителя).

5.4.5. Передаточные параметры

Основной характеристикой ИОУ, отражающей зависимость между выходным и входным напряжениями, является *амплитудная характеристика* (рис. 5.20). Вид ее можно описать тремя параметрами: напряжением $U_{\text{см}}$ (сдвиг по оси $U_{\text{Вх}}$), напряжением $U_{\text{ИОУм}}$ (ограничение величины $U_{\text{Вых}}$) и коэффициентом усиления K (наклон линейного участка). Напряжения $U_{\text{см}}$ и $U_{\text{ИОУм}}$ были описаны ранее.

Коэффициент усиления обычно находится как отношение изменения выходного напряжения к вызвавшему его изменению напряжения на дифференциальном входе:

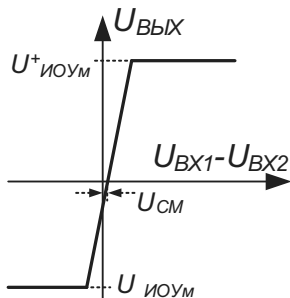


Рис. 5.20

$$K = U_{\text{ВЫХ}} / (U_{\text{ВХ1}} - U_{\text{ВХ2}}). \quad (5.34)$$

Значения K превышают в большинстве случаев 100 тысяч и могут достигать миллиона и более.

5.4.6. Частотные и временные параметры

С увеличением частоты коэффициент усиления уменьшается (рис. 5.21), причем для параметризации частотных свойств ИОУ при работе с малым уровнем сигнала принято использовать частоту единичного усиления F_1 . На этой частоте модуль коэффициента усиления уменьшается до единицы. Есть и другие подобные параметры для частотной характеристики, например, граничная частота $F_{\text{ГР}}$, на которой модуль коэффициента усиления уменьшается в корень из двух раз по сравнению с номинальным K_0 :

$$K(F_{\text{ГР}}) = K_0 / \sqrt{2} \approx 0,7 K_0. \quad (5.35)$$

При работе с прямоугольными сигналами на практике более удобно использовать для описания динамики ИОУ *временные* параметры.

В этом случае на выходе наблюдают реакцию усилителя на перепад входного прямоугольного сигнала (рис. 5.22). Свойства ИОУ в этом случае описываются временем установления в зону заданной погрешности $t_{\text{УСТ}}$ и скоростью нарастания выходного напряжения, определяемой в В/мкс.

Следует также отметить, что по виду переходного процесса на выходе ИОУ с ОС можно судить о склонности схемы к самовозбуждению. Если реакция усилителя колебательная, то в схеме при изменении, например, температуры может возникнуть самопроизвольная генерация. Если реакция усилителя экспоненциальная, то в схеме имеется определенный запас устойчивости.

5.4.7. Типы ИОУ

На практике к ИОУ в зависимости от технической надобности предъявляются различные требования. Поэтому существует большое разнообразие типов ИОУ. Самыми массовыми и дешевыми яв-

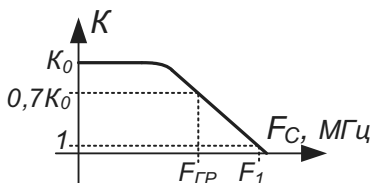


Рис. 5.21

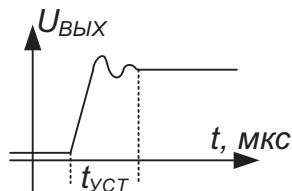


Рис. 5.22

ляются ИОУ *общего назначения*, которые имеют средние характеристики и предназначены для применений, не требующих точности и быстродействия.

Для высокоточных устройств используют *прецизионные ИОУ*. Свои ниши применения имеют *быстродействующие, высоковольтные, мощные* или наоборот *микрощебляющие ИОУ*.

В последнее время в связи с экономией электроэнергии все более востребованы *однополярные ИОУ* с низковольтным питанием, например, от батарейки 2..3 В, имеющие расширенный диапазон выходного напряжения, обозначаемый термином «*Rail-To-Rail*» (см. п.5.4.4).

В сложных усилительных схемах, требующих нескольких ИОУ, рационально применять *многоканальные* микросхемы, в которых на одном кристалле выполнены два, четыре или даже более отдельных ИОУ с общим питанием [19].

§5.5. Неинвертирующий УПТ на основе ИОУ

5.5.1. Схема

Входным для цепи ОС (рис. 5.23), состоящей из резисторов R_1 и R_2 , является напряжение $U_{Вых}$, значит, в усилителе реализуется ОС *по напряжению*.

Сигнал обратной связи $U_{ОС}$ объединяется с входным по *последовательной* схеме (см. п. 5.1.2) и поступает на инвертирующий вход ИОУ, то есть усилитель вносит фазовый сдвиг, равный 180 угловым градусам. Резистивный делитель в цепи ОС фазу не сдвигает. Поэтому суммарный фазовый сдвиг тоже равен 180 градусам, следовательно, в схеме действует последовательную отрицательную ОС (см. п. 5.1.2).

Цепи питания и другие вспомогательные выводы для простоты на рисунке не показаны.

Чтобы получить расчетные соотношения, поставим полярности сигналов в схеме. Полярность напряжения $U_{Вх}$ ставим произвольно, полярность ΔU будет совпадать с ней. Так как ΔU поступает на неинвертирующий вход ИОУ, то полярность выходного напряжения сов-

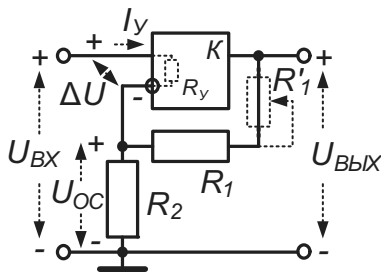


Рис. 5.23

падает с полярностью ΔU и входного, поэтому такое включение ИОУ называют *неинвертирующим*. Кроме того заметим, что напряжение U_{oc} вычитается из $U_{вх}$, то есть действительно в схеме осуществляется *отрицательная ОС*.

5.5.2. Коэффициент усиления

Для простоты примем для ИОУ значения $U_{см}$ и $I_{вх}$ равными нулю. Их влияние сказывается на величине $U_{ввых}$, которую учтем при анализе нулевого уровня (см. §5.7). Но коэффициент усиления K и входное сопротивление R_y будем считать для ИОУ конечными и учитывать в дальнейшем.

Найдем коэффициент усиления рассмотренной схемы с учетом ОС как отношение выходного и входного напряжений. Для этого запишем уравнение для входной цепи

$$U_{вх} = U_{oc} + \Delta U. \quad (5.36)$$

Затем выразим U_{oc} и ΔU через $U_{ввых}$:

$$\Delta U = U_{ввых} / K, \quad (5.37)$$

$$U_{oc} = U_{ввых} R_2 / (R_1 + R_2). \quad (5.38)$$

Подставляя (5.37) и (5.38) в (5.36) и вынося затем $U_{ввых}$ в левую часть, получим:

$$U_{ввых} = U_{вх} K / (1 + K\beta), \quad (5.39)$$

где $\beta = R_2 / (R_1 + R_2)$ – коэффициент ОС, определенный без учета входного сопротивления ИОУ (см. п. 5.6.4).

Поделив выражение (5.39) на $U_{вх}$, получим коэффициент усиления УПТ с последовательной ООС:

$$K_{oc} = \frac{R_1 + R_2}{R_2} \cdot \frac{K\beta}{1 + K\beta}. \quad (5.39)$$

На практике, благодаря большому коэффициенту усиления ИОУ, глубина ОС $K\beta$ оказывается также большой, то есть $K\beta \gg 1$. Поэтому правая дробь оказывается практически равной единице, и выражение (5.40) чаще всего применяется в упрощенном виде:

$$K_{oc} \approx (R_1 + R_2) / R_2 = 1 + R_1 / R_2 \quad (5.41).$$

Отсюда видно, что коэффициент усиления всей схемы K_{oc} почти не зависит от большого и нестабильного коэффициента усиления ИОУ K , а определяется отношением резисторов обратной связи и может быть неизменным, если резисторы взять точными и высокостабильными.

5.5.3. Входное сопротивление

Определяется по методике, применявшейся при анализе одиночного каскада в п. 5.2.3 с использованием закона Ома для входной цепи и уравнения (5.36):

$$R_{BX} = U_{BX} / I_{BX} = (U_{OC} + \Delta U) / I_y = U_{OC} / I_y + \Delta U / I_y \quad (5.42)$$

Здесь ток I_y – это ток, вызванный приращениями входного сигнала, и он не равен постоянному току I_{BX} ИОУ.

Нетрудно заметить, что последнее слагаемое в (5.42) $\Delta U / I_y$ равно R_y . Кроме того, и напряжение U_{OC} можно выразить через $U_{ВЫХ}$ и, в конечном счете, через ΔU :

$$U_{OC} = \beta U_{ВЫХ} = K\beta \Delta U. \quad (5.43)$$

Тогда, подставляя значение U_{OC} в (5.42), получим:

$$R_{BX} = K\beta \Delta U / I_y + \Delta U / I_y = K\beta R_y + R_y = R_y(1 + K\beta). \quad (5.44)$$

Из последнего выражения следует, что в схеме с последовательной ООС входное сопротивление УПТ в $(1 + K\beta)$ раз больше сопротивления усилителя без ОС.

5.5.4. Главные особенности неинвертирующего УПТ

Подведем в краткой форме итоги для неинвертирующего включения ИОУ, которые желательно запомнить, поскольку они будут полезны в дальнейшем при пояснении других схем и практических расчетах.

- Обратная связь в данной схеме подается на инвертирующий вход.
- Полярность напряжения $U_{ВЫХ}$ совпадает с полярностью U_{BX} .
- Величину $\Delta U = U_{ВЫХ} / K$, благодаря очень большому коэффициенту усиления K , в большинстве случаев можно считать исчезающе малой, то есть в дальнейшем будем полагать $\Delta U \approx 0$. Для примера, пусть $U_{ВЫХ} = 2 \text{ В}$, $K = 10^6$, тогда величина $\Delta U = 2/10^6 = 2 \text{ мкВ}$, то есть действительно получается очень малой.
- Вследствие малости ΔU выражение (5.36) можно использовать в упрощенной форме $U_{BX} \approx U_{OC}$.
- Коэффициент усиления УПТ, как видно из формулы (5.41), в большинстве случаев не зависит от ИОУ ($K_{OC} = 1 + R_1 / R_2$). И можно получить требуемое усиление, выбирая соответствующие сопротивления ОС (см. п. 5.5.5).

- Входное сопротивление неинвертирующего усилителя оценивается по формуле (5.44), то есть $R_{вх} = R_y (1 + K\beta)$. Благодаря большой глубине ОС $K\beta$ значение $R_{вх}$ получается весьма большим, но нестабильным, так как коэффициент усиления K зависит от свойств конкретной микросхемы и от температуры окружающей среды.

5.5.5. Порядок расчета

Обычно при расчете бывает задан коэффициент $K_{ос}$, или известны номинальные значения напряжений $U_{ввхн}$ и $U_{вхн}$. В последнем случае легко найти и требуемый коэффициент:

$$K_{ос} = U_{ввхн} / U_{вхн} \quad (5.45)$$

На практике чаще всего $K\beta \gg 1$, поэтому для расчета можно использовать уравнение (5.41), из которого нужно найти величины сопротивлений R_1 и R_2 . Но в этом уравнении имеем два неизвестных.

Поэтому для расчета необходимо привлекать еще одно условие, например, заключающееся в том, что ток через резисторы R_1 и R_2 должен быть значительно больше, чем входной ток ИОУ $I_{вх}$. Это необходимо для того, чтобы входная цепь не нагружала делитель и не влияла на коэффициент передачи звена ОС. Задавшись током через делитель R_1 и R_2 , например в 100 раз больше тока $I_{вх}$, получим второе расчетное соотношение:

$$U_{ввхн} / (R_1 + R_2) > 100 I_{вх} \quad (5.46)$$

Отсюда можно определить сумму сопротивлений. Но обычно $K_{ос} \gg 1$ и тогда $R_1 \gg R_2$, то есть выражение (5.46) можно записать в упрощенном виде и найти, таким образом,

$$R_1 \approx U_{ввхн} / 100 I_{вх} \quad (5.47)$$

Зная R_1 , можно, используя (5.41), найти

$$R_2 = R_1 / (K_{ос} - 1). \quad (5.48)$$

Примечание: вместо условия (5.46) для определения R_1 можно пользоваться и другими соотношениями, например, основанными на заданной аддитивной погрешности (см. п. 5.7.3).

5.5.6. Числовой пример расчета

Пусть задано напряжение $U_{ввхн} = 4 \text{ В}$ при $U_{вхн} = 20 \text{ мВ}$. Тогда, используя (5.45), можно получить требуемый коэффициент усиления $K_{ос} = 4 / 0,020 = 200$.

Кроме того, пусть из справочника [19] известен для выбранного ИОУ (например, КР544УД1А) его максимальный входной ток $I_{BX} = 0,05 \text{ нА}$, что согласно (5.46) позволяет определить $R_1 = 4 / 5 \cdot 10^{-9} = 800 \text{ МОм}$. Однако такое мегаомное сопротивление сравнимо с сопротивлением изоляции и поэтому на практике не применяется. Выбираем из стандартного числового ряда значений резисторов [5] более реальную величину, например, $R_1 = 820 \text{ кОм}$.

Затем по формуле (5.48) находим $R_2 = 820000 / 200 = 4,1 \text{ кОм}$. Такое числовое значение отсутствует в стандартном ряде значений, поэтому приходится выполнять R_2 из двух последовательно соединенных резисторов, например, номиналом $3,9 \text{ кОм}$ и 200 Ом . Для более точного задания коэффициента K_{OC} можно поставить подстроечный потенциометр R'_1 , который показан на рис. 5.23 пунктиром и сопротивление которого входит в состав расчетного R_2 .

В заключение расчета обычно проверяют значение глубины ОС $K\beta$. Для этого из справочника находят коэффициент усиления K . Например, для микросхемы КР544УД1А значение $K = 200$ тысяч [19]. Тогда, выражая из (5.10) значение $\beta = 1 / K_{OC}$, вычислим $K\beta = 200000 / 200 = 1000$. То есть предположение о том, что $K\beta \gg 1$, вполне оправдано.

При необходимости по формуле (5.44) проводят оценку входного сопротивления неинвертирующего УПТ.

Примеры использования неинвертирующего УПТ с последовательной ООС приведены в §6.2.

§5.6. Инвертирующий УПТ на основе ИОУ

5.6.1. Схема

Во входной цепи инвертирующего УПТ (рис. 5.24) суммируются токи, а не напряжения, как в предыдущей схеме. Входной и разностный сигналы имеют общую точку, то есть включены как бы параллельно. Поэтому подобный способ объединения сигналов называют *параллельной* ОС. Причем входным для цепи ОС является напряжение, значит, как и в неинвертирующем УПТ, действует ОС по напряжению.

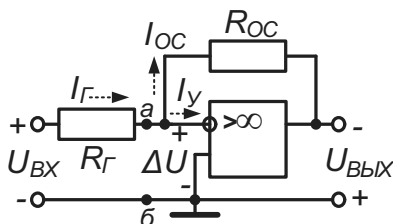


Рис. 5.24

Чтобы определить направления сравниваемых токов, проставим полярности напряжений. Полярность U_{BX} для определенности будем считать положительной, полярность ΔU будет совпадать с ней. Полярность же выходного напряжения $U_{ВЫХ}$ оказывается обратной, так как ΔU поступает на инвертирующий вход ИОУ и происходит изменение полярности выходного сигнала. Поэтому такой усилитель называют инвертирующим.

5.6.2. Коэффициент усиления

Как и прежде пренебрежем влиянием U_{CM} и $I_{ВХ}$, действие которых учтем в §5.7 при определении нулевого уровня, и запишем для инвертирующего УПТ алгебраическую сумму токов для узла суммирования:

$$I_y = I_r - I_{OC} \quad (5.49)$$

Ток обратной связи I_{OC} отнимается от входного I_r и, следовательно, имеем *отрицательную* ОС параллельного вида (см. п. 5.1.2). Эти токи задаются при помощи резисторов R_{OC} и R_r и не показанным для простоты входным сопротивлением ИОУ:

$$I_r = (U_{BX} - \Delta U) / R_r \quad (5.50)$$

$$I_{OC} = (U_{ВЫХ} + \Delta U) / R_{OC} \quad (5.51)$$

$$I_y = \Delta U / R_y \quad (5.52)$$

Причем ток I_{OC} , как видно из формулы (5.51), пропорционален $U_{ВЫХ}$, то есть, действительно, в схеме существует обратная связь *по напряжению*.

Повторно отметим, что ток I_y вызван приращениями входного сигнала и он не равен постоянному току $I_{ВХ}$ ИОУ.

$$\frac{U_{ВХ}}{R_r} = \frac{U_{ВЫХ}}{R_{OC}} + \Delta U \left(\frac{1}{R_r} + \frac{1}{R_y} + \frac{1}{R_{OC}} \right). \quad (5.53)$$

Найдем коэффициент усиления усилителя с параллельной ООС по напряжению. Подставляя в (5.49) значения токов из (5.50), (5.51) и (5.52), получим:

$$\frac{1}{R_r} = \left(\frac{1}{R_r} + \frac{1}{R_y} + \frac{1}{R_{OC}} \right), \quad (5.54)$$

Затем учтем, что $\Delta U = -U_{ВЫХ} / K$ (с учетом инвертирования полярности) и, обозначив выражение в круглых скобках, равное параллельному соединению R_r , R_y и R_{OC} , как получим

$$\frac{U_{BX}}{R_r} = \frac{U_{BbIX}}{R_{OC}} - \frac{U_{BbIX}}{K} \frac{1}{R_n} = \frac{U_{BbIX}}{R_{OC}} \left(1 - \frac{R_{OC}}{KR_n} \right). \quad (5.55)$$

Далее, обозначая отношение сопротивлений R_n и R_{OC} как коэффициент передачи цепи ООС:

$$\beta = R_n / R_{OC}, \quad (5.56)$$

получим уравнение, решая которое относительно U_{BbIX} и поделив результат на U_{BX} , найдем коэффициент усиления усилителя с учетом действия ООС:

$$K_{OC} = \frac{U_{BbIX}}{U_{BX}} = - \frac{R_{OC}}{R_r} \frac{K \beta}{1 + K \beta}. \quad (5.57)$$

Обратим внимание на две особенности формулы (5.57). Во-первых, знак минус перед формулой указывает на то, что выходной сигнал будет иметь обратную (инверсную) полярность по сравнению с входным.

Во-вторых, благодаря очень большому значению K для современных ИОУ, произведение $K\beta$ на практике оказывается во много раз больше единицы (однако нужно помнить про уточнение в п. 5.6.4), и потому для расчетов можно воспользоваться упрощенным уравнением

$$K_{OC} = -R_{OC} / R_r \quad (5.58)$$

5.6.3. Входное сопротивление

Найдем входное сопротивление УПТ с параллельной ООС (рис. 5.24), представляя входную цепь как последовательное соединение резистора R_r и сопротивления промежутка R_{ag} :

$$R_{BX} = R_r + R_{ag} = R_r + \Delta U / (I_{OC} + I_y). \quad (5.59)$$

Несложная оценка показывает, что при больших значениях коэффициента усиления K , присущих современным усилителям, второе слагаемое значительно меньше первого, так как $\Delta U \approx 0$ (см. п. 5.5.4). И с достаточной для большинства применений точностью можно определять входное сопротивление УПТ с параллельной ООС по формуле

$$R_{BX} = R_r \quad (5.60)$$

5.6.4. Зависимость коэффициента β от сопротивления R_y

В отличие от неинвертирующей схемы, где нет необходимости в значительном увеличении значения сопротивления R_r , в инвертирующем УПТ коэффициент передачи цепи ОС β для инвертиру-

ющего УПТ, зависит, как видно из (5.55), от R_{Π} и в соответствии с (5.53) от R_{γ} , что ограничивает в некоторых случаях возможность увеличения R_{oc} .

Дело в том, что при значительном увеличении величин R_{oc} , R_{γ} при заданном их отношении в соответствии с формулой (5.58) может получиться так, что R_{oc} и R_{γ} будут сравнимыми или даже много большими R_{γ} . В этом случае, как следует из формулы (5.54), R_{Π} будет определяться в основном входным сопротивлением ИОУ:

$$R_{\Pi} \approx R_{\gamma} \quad (5.61)$$

Поэтому коэффициент передачи цепи ОС β при больших R_{oc} будет обратно пропорционально зависеть от величины R_{oc} :

$$\beta = R_{\Pi} / R_{oc} \approx R_{\gamma} / R_{oc} \quad (5.62)$$

а значит, и глубина ООС $K\beta$ станет также уменьшаться при значительных R_{γ} и R_{oc} . Поэтому при расчете инвертирующего УПТ проверка значения $K\beta$ является обязательной, для того чтобы при $K\beta$ менее 50..100 провести повторный расчет с меньшим значением R_{oc} или с другим ИОУ, имеющим больший коэффициент усиления.

5.6.5. Главные особенности инвертирующего УПТ

Эти особенности будут полезны в дальнейшем при практических расчетах и при анализе для других схем с инвертирующим включением ИОУ (см. §6.1), поэтому желательно запомнить следующие выводы.

- Сопротивление ОС, как и в прежней схеме, подключается к инвертирующему входу.
- Неинвертирующий вход нельзя оставлять никуда не подключенным, его обязательно нужно соединять с общей точкой схемы или к другой точке с фиксированным потенциалом, если этого требует принцип действия схемы. Через оборванный неинвертирующий вход постоянный ток I_{Bx} не идет, поэтому входной дифференциальный каскад ИОУ, который должен быть сбалансирован (см. п. 5.3.4), не будет в этом случае работать.
- Полярность напряжения U_{Bx} противоположна полярности $U_{Bx'}$.
- Величину ΔU , как и прежде, благодаря очень большому коэффициенту усиления K , в дальнейшем будем полагать исчезающе малой, то есть $\Delta U \approx 0$.

- Вследствие малости ΔU ток $I_y = \Delta U / R_y$ можно считать ничтожным, поэтому выражение (5.49) можно использовать в упрощенной форме

$$I_{oc} \approx I_r \quad (5.63)$$

- Коэффициент усиления УПТ, как видно из формулы (5.58), практически не зависит от ИОУ, то есть требуемое усиление ($K_{oc} = -R_{oc} / R_r$) определяется выбором сопротивлений ОС при расчете.
- Входное сопротивление инвертирующего УПТ определяется в соответствии с формулой (5.60) внешним резистором R_r , то есть является стабильным, но сравнительно небольшим по величине.
- В инвертирующем УПТ обязательно нужно учитывать влияние входного сопротивления ИОУ R_y на коэффициент передачи цепи ОС (см. п. 5.6.4), особенно при значениях R_r близких к величине R_y .

5.6.6. Порядок расчета

Пусть заданы значения K_{oc} и R_{bx} , которые позволяют из формулы (5.60) выбрать значение $R_r \geq R_{bx}$, а затем из (5.57) рассчитать $R_{oc} = R_r \cdot K_{oc}$.

Затем необходимо найти с использованием (5.54) величину R_n и далее с учетом вычисленного по формуле (5.56) значения β , проверить значение глубины ОС $K\beta$. Если величина $K\beta$ получилась меньше, чем 50..100, то повторяют расчет с другим ИОУ, имеющим большее входное сопротивление и коэффициент усиления.

Кроме того, при оценке R_{oc} нужно, как и в неинвертирующем УПТ, учитывать соотношение (5.46), требующее, чтобы ток I_{oc} , вычисляемый по формуле (5.51), был больше I_{bx} выбранного ИОУ. По сути, это требование связано с погрешностью, обусловленной нулевым уровнем УПТ (см. п. 5.7.4), и при необходимости погрешность можно рассчитать и уменьшить за счет включения тококомпенсирующего сопротивления $R_T = R_r \parallel R_{oc}$ (см. п. 5.7.3).

5.6.7. Числовой пример расчета

Пусть задан $K_{oc} = 200$ и $R_{bx} = 10 \text{ кОм}$. Тогда выбираем $R_r = R_{bx} = 10 \text{ кОм}$ и рассчитываем $R_{oc} = R_r K_{oc} = 10000 \cdot 200 = 2 \text{ МОм}$.

Отметим, что полученные значения сопротивлений (10 кОм и 2 МОм) совпадают со значениями стандартного ряда [5].

Затем для ИОУ типа КР544УД1А ($K = 200000$, $R_y = 10 \text{ МОм}$) проверим глубину ООС $K\beta = KR_{\Gamma} / R_{OC} \approx KR_{\Gamma} / R_{OC} = 2 \cdot 10^5 \cdot 10^3 / 2 \cdot 10^6 = 100$, то есть она получилась значительно большей единицы, что подтверждает правомерность использования упрощенных формул.

После этого проведем оценку тока ОС, нагружающего ИОУ, $I_{OC} \approx U_{ВЫХ} / R_{OC} = 4 / 2 \cdot 10^6 = 2 \text{ мкА}$, что значительно больше входного тока $I_{BX} = 0,05 \text{ нА}$ выбранного ИОУ.

Для оценки погрешностей УПТ можно пользоваться соотношениями, полученными далее в следующем параграфе.

§5.7. Погрешности УПТ

5.7.1. Общие сведения

В идеальном усилителе (прямая «1» на рис. 5.25) выходное напряжение зависит только входного:

$$U_{ВЫХ} = K_{ОСНОМ} U_{ВХ} \quad (5.64)$$

где $K_{ОСНОМ}$ – номинальный (расчетный) коэффициент усиления УПТ.

Со временем параметры усилителя могут измениться, особенно при изменении внешней температуры. Это может привести к изменению выходного напряжения даже при неизменном входном. Например, может появиться нулевой уровень (характеристика сдвинется параллельно вверх или вниз – прямая «2» на рис. 5.25).

Или может измениться коэффициент усиления схемы (характеристика изменит наклон – прямая «3»). Поэтому выходное напряжение окажется зависящим не только от входного:

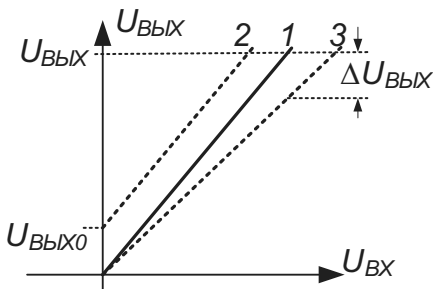


Рис. 5.25

$$U_{ВЫХ} = (K_{ОСНОМ} \pm \Delta K_{OC}) \cdot U_{ВХ} \pm U_{ВЫХ0} \quad (5.65)$$

Подобные изменения параметров усилителя со временем и с температурой характеризуют обычно относительной погрешностью [12], причем различают две основные составляющие погрешности: *аддитивную*, вызванную наличием нулевого уровня:

$$\delta_A = U_{\text{ВЫХ0}} / U_{\text{ВЫХмакс}} \quad (5.66)$$

и *мультипликативную*, связанную с изменением коэффициента усиления:

$$\delta_M = \Delta K_{\text{ОС}} / K_{\text{ОСном}} = \Delta U_{\text{ВЫХ}} / U_{\text{ВЫХмакс}} \quad (5.67)$$

Рассмотрим вначале аддитивную погрешность усилителя на основе ИОУ, обусловленную наличием $U_{\text{СМ}}$ и $I_{\text{ВХ}}$. Будем считать, что указанные параметры действуют независимо друг от друга и создают соответствующие составляющие нулевого уровня, то есть

$$U_{\text{ВЫХ0}} = U_{\text{ВЫХсм}} + U_{\text{ВЫХТ}} \quad (5.68)$$

Назовем первую составляющую $U_{\text{ВЫХсм}}$ потенциальной, а вторую $U_{\text{ВЫХТ}}$ – соответственно токовой. Оценим их по отдельности, считая, что $K\beta \gg 1$.

5.7.2. Потенциальная аддитивная составляющая

Поскольку нулевой уровень усилителя определяется при $U_{\text{ВХ}} = 0$, то инвертирующий и неинвертирующий УПТ в этом случае будут выглядеть одинаково, и для анализа можно пользоваться обобщенной схемой (рис. 5.26). Назначение резистора R_T поясним позднее (см. п. 5.7.3).

Определим $U_{\text{ВЫХсм}}$ (считая $I_{\text{ВХ1}} = 0$ и $I_{\text{ВХ2}} = 0$) из следующих рассуждений. Пусть на выходе действует напряжение $U_{\text{ВЫХсм}}$, обусловленное наличием $U_{\text{СМ}}$, и на входе ИОУ через звено ОС появляется $U_{\text{ОСсм}}$, которое стремится скомпенсировать напряжение $U_{\text{СМ}}$ усилителя и прикладывается между входами.

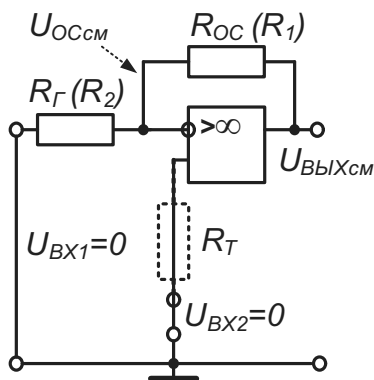


Рис. 5.26

Но полной компенсации быть не может, так как при этом $U_{\text{ВЫХ}}$ должно стать нулевым, и исчезнет сигнал ОС.

Поэтому (при $K\beta \gg 1$)

$$U_{\text{ОСсм}} \approx U_{\text{СМ}} \quad (5.69)$$

Учитывая, что с другой стороны

$$U_{\text{ОСсм}} = U_{\text{ВЫХсм}} \cdot \beta, \quad (5.70)$$

где β – как и прежде, коэффициент передачи цепи ОС; подставим (5.70) в левую часть (5.69) и найдем

$$U_{ВЫХСМ} \approx \pm U_{СМ} / \beta \approx \pm U_{СМ} \cdot K_{ОС}. \quad (5.71)$$

Знак $\pm$ в последней формуле подставлен формально, с учетом того, что ни величина, ни полярность напряжения $U_{СМ}$ для конкретного усилителя не известны (см. п. 5.4.3), и являются для партии ИОУ случайными величинами. Поэтому, чтобы не производить вероятностную оценку [12] величины $U_{ВЫХСМ}$, для простоты под $U_{СМ}$ будем понимать максимально возможное значение, приводимое в справочниках [19]. Вместе с тем нужно помнить, что такая оценка является завышенной.

Оценим $U_{ВЫХСМ}$ для усилителя с $K_{ОС} = 200$, рассчитанного в п. 6.1.6, если максимальное значение $U_{СМ}$ для ИОУ типа КР544УД1 [19] равно 20 мВ. Тогда максимальное значение $U_{ВЫХСМ} = 0,02 \cdot 200 = 4$ В. Такой большой нулевой уровень недопустим, и необходимо выбрать другой ИОУ с меньшим $U_{СМ}$, причем для выбора можно воспользоваться соотношением, полученным из (5.71):

$$U_{СМ\text{треб}} = U_{ВЫХСМ} / K_{ОС}. \quad (5.72)$$

5.7.3. Токовая аддитивная составляющая

Для нахождения токового слагаемого уравнения (5.68) будем считать, что $U_{СМ} = 0$ и действуют только входные токи. На первый взгляд $U_{ВЫХ}$ должно в этом случае равняться нулю, так как $U_{СМ} = 0$. Однако это не так.

Входные токи (рис. 5.27), втекая в усилитель, встречают на своем пути разные сопротивления. Ток $I_{ВХ2}$ протекает через резистор R_T , а ток $I_{ВХ1}$ – через $R_T \parallel R_{ОС}$. Поэтому на сопротивлениях выделяются неодинаковые падения напряжения U_{T1} и U_{T2} , и между входами ИОУ образуется разность потенциалов, которая усиливается и приводит к появлению $U_{ВЫХТ}$.

Найдем указанные потенциалы входов. Проще всего найти

$$U_{T1} = -I_{ВХ2} \cdot R_T \quad (5.73)$$

Напряжение на другом входе U_{T2} зависит от двух источников энергии – тока $I_{ВХ1}$ и напряжения $U_{ВЫХ}$. Воспользуемся принципом супер-

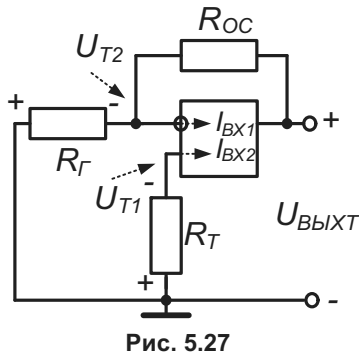


Рис. 5.27

позиции и найдем составляющие U_{T2} по отдельности для каждого источника:

$$U_{T2} = -I_{BX1} \cdot R_{\Pi} + U_{ВЫХ} \cdot R_{\Pi} / R_{OC}, \quad (5.74)$$

где R_{Π} – параллельное соединение резисторов R_r и R_{OC} .

Далее, считая, что напряжения U_{T1} и U_{T2} примерно равными (при $K\beta \gg 1$), приравниваем правые части уравнений (5.73) и (5.74) и получаем

$$-I_{BX2} \cdot R_r = -I_{BX1} \cdot R_{\Pi} + U_{ВЫХ} \cdot R_{\Pi} / R_{OC}. \quad (5.75)$$

Решая это уравнение относительно $U_{ВЫХ}$, находим токовую составляющую

$$U_{ВЫХТ} = R_{OC} \left(I_{BX1} - I_{BX2} \frac{R_r}{R_{\Pi}} \right). \quad (5.76)$$

Для того чтобы уменьшить влияние входных токов, желательно уравнивать сопротивления в цепях протекания тока. Именно для этого включают дополнительное сопротивление R_r .

Вообще говоря, можно так подобрать сопротивление R_r , чтобы падения напряжения от протекания токов были одинаковыми, и тогда выражение в круглых скобках в (5.76) и $U_{ВЫХТ}$ в целом будут равны нулю.

Но индивидуальный подбор сопротивлений нежелателен. Поэтому на практике выбирают $R_r = R_{\Pi} = R_r \parallel R_{OC}$ и пользуются упрощенным уравнением (5.76) для расчета

$$U_{ВЫХТ} = (I_{BX2} - I_{BX1}) \cdot R_{OC} = I_p \cdot R_{OC}. \quad (5.77)$$

Эту формулу можно использовать для ограничения максимального сопротивления R_{OC} (см. п. 5.6.6) при известных значениях $U_{ВЫХТ}$ и I_p .

5.7.4. Оценка общей аддитивной погрешности

Знак разностного тока I_p , как и знак напряжения U_{CM} , для конкретного ИОУ неизвестны, поэтому, объединяя составляющие (5.71) и (5.77) для уравнения (5.68), запишем величину нулевого уровня УПТ в следующем виде:

$$U_{ВЫХ0} = \pm U_{CM} K_{OC} \pm I_p \cdot R_{OC}. \quad (5.78)$$

Можно оценить, используя эту формулу, напряжение $U_{ВЫХ0}$ в самом худшем случае, когда отдельные составляющие имеют одинаковый знак и максимальны. Такая оценка позволяет провести границы (правило «трех сигма» из теории вероятностей) для разброса нулевого уровня. Практически эти границы почти недостижимы. Однако та-

кая максимальная оценка часто используется на практике, как более простая. Следует только помнить, что она завышена.

Некоторые усилители имеют специальные выводы, к которым подключается балансирующий потенциометр для устранения сдвига нуля (см. п. 5.4.2). На этапе настройки подают напряжение $U_{вх} = 0$ и вращением движка потенциометра добиваются, чтобы напряжение на выходе усилителя стало близким к нулю.

При эксплуатации усилителя из-за температурного дрейфа $U_{см}$ и $I_{вх}$ нулевой уровень может снова появиться. Формулу (5.78) можно использовать и для оценки дрейфа нулевого уровня, если под $U_{см}$ и I_p понимать их изменения в заданном температурном диапазоне.

5.7.5. Мультипликативная погрешность

При определении мультипликативной погрешности усилителя по формуле (5.67) нужно найти $\Delta K_{ос}$ – абсолютное изменение величины $K_{ос}$.

Для наглядности повторим формулу (5.40) для коэффициента усиления неинвертирующего УПТ:

$$K_{ос} = \frac{R_1 + R_2}{R_2} \cdot \frac{K\beta}{1 + K\beta} \quad (5.79)$$

и формулу (5.57) для инвертирующего УПТ:

$$K_{ос} = -\frac{R_{ос}}{R_r} \cdot \frac{K\beta}{1 + K\beta}. \quad (5.80)$$

Изменение величины $K_{ос}$, как следует из формул, может быть вызвано изменением сопротивлений обратной связи и изменением глупины ОС $K\beta$, которое в основном связано с нестабильностью K .

Известен общий метод определения малых приращений для функции нескольких переменных, заключающийся в нахождении ее дифференциала. Поясним этот метод на примере формулы (5.80). Предположим, что значения $\Delta K_{ос}$ достаточно малые и их без большой погрешности можно заменить бесконечно малым значением дифференциала $dK_{ос}$, который для функции нескольких переменных (5.80) вычисляется по формуле:

$$dK_{ос} = dR_{ос} \cdot \frac{dK_{ос}}{dR_{ос}} + dR_r \cdot \frac{dK_{ос}}{dR_r} + d(K\beta) \cdot \frac{dK_{ос}}{d(K\beta)}. \quad (5.81)$$

Вычисляя частные производные, можно найти

$$dK_{ос} = dR_{ос} \cdot \frac{1}{R_r} - dR_r \cdot \frac{R_{ос}}{R_r^2} + d(K\beta) \cdot \frac{R_{ос}}{R_r} \cdot \frac{1}{(1 + K\beta)^2}. \quad (5.82)$$

Затем делаем обратный переход от бесконечно малых приращений к конечным, но достаточно малым приращениям, то есть, полагаем $dK_{oc} \approx \Delta K_{oc}$, $dR_{oc} \approx \Delta R_{oc}$, $dR_r \approx \Delta R_r$, $d(K\beta) \approx \Delta(K\beta)$. То есть можно определить на основе (5.82) абсолютное изменение коэффициента усиления:

$$\Delta K_{oc} = \Delta R_{oc} \cdot \frac{1}{R_r} - \Delta R_r \cdot \frac{R_{oc}}{R_r^2} + \Delta(K\beta) \cdot \frac{R_{oc}}{R_r} \cdot \frac{1}{(1+K\beta)^2}. \quad (5.83)$$

Далее в соответствии с формулой (5.67) делим (5.83) на номинальный коэффициент ($K_{ocном} \approx R_{oc} / R_r$) и находим относительную мультипликативную погрешность:

$$\delta_M = \frac{\Delta K_{oc}}{K_{ocном}} = \frac{\Delta R_{oc}}{R_{oc}} - \frac{\Delta R_r}{R_r} + \frac{\Delta(K\beta)}{(1+K\beta)^2} \approx \delta_{oc} - \delta_r + \delta_{K\beta} / K K \beta. \quad (5.84)$$

Обратим внимание на две важные особенности полученного уравнения.

Во-первых, знак минус между первым и вторым слагаемыми показывает, что происходит взаимная компенсация погрешностей, вызванных температурными изменениями резисторов. Если выбрать термостабильные резисторы с одинаковым температурным коэффициентом, то первые две составляющие могут оказаться практически равными нулю.

Во-вторых, величина третьего слагаемого зависит от глубины ОС $K\beta$, чем она больше, тем меньше погрешность. Например, если при изменении температуры коэффициент усиления ИОУ K изменился на 5%, то, считая β неизменным, а $K\beta = 100$, погрешность от изменения коэффициента усиления УПТ в целом составит $\delta_{K\beta} / K\beta = 5\% / 100 \approx 0,05\%$.

Формулу (5.84) можно использовать для случаев, когда изменения $K\beta$ не превосходят примерно десяти процентов. При больших изменениях формула дает заниженную оценку, и надо использовать другую методику оценки мультипликативной погрешности [12].

6 ПРИМЕРЫ ИСПОЛЬЗОВАНИЯ УПТ НА ИОУ

§6.1. Использование инвертирующего УПТ

В свое время инвертирующие УПТ широко использовались в аналоговой вычислительной технике для реализации операций суммирования, интегрирования и дифференцирования входных переменных. В современной технике подобные функциональные устройства до сих пор применяются при частичной обработке аналоговых информационных сигналов.

6.1.1. Инвертирующий сумматор напряжений

В ряде случаев необходимо суммировать несколько напряжений с учетом знака. Например, в модемной технике используется много-частотная кодировка при наборе номера, когда нужно сложить сигналы двух различных частот. Для этой цели можно применить, например, инвертирующий УПТ, в котором суммируются токи, добавив дополнительные источники сигналов и соответствующие резисторы (рис. 6.1).

Определим зависимость $U_{Вых}$ от входных сигналов, записав уравнение для выходной цепи по контуру R_{OC} и ΔU с учетом инвертирующих свойств УПТ:

$$U_{Вых} = -U_{ROC} - \Delta U = -I_{OC} R_{OC} - \Delta U. \quad (6.1)$$

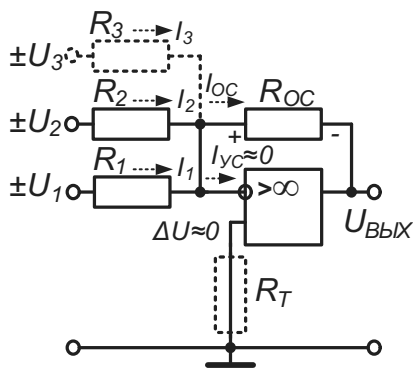


Рис. 6.1

Будем считать далее (см. п. 5.6.5), что $\Delta U \approx 0$, $I_{yc} \approx 0$ и $I_{oc} \approx I_r$. В нашем случае:

$$I_r = I_1 + I_2 \approx \pm U_1 / R_1 \pm U_2 / R_2. \quad (6.2)$$

С учетом этого уравнение (6.1) можно представить в виде:

$$U_{Вых} = -I_{oc} R_{oc} = -(I_1 + I_2) R_{oc} = \mp U_1 \frac{R_{oc}}{R_1} \mp U_2 \frac{R_{oc}}{R_2}. \quad (6.3)$$

Таким образом, схема выполняет алгебраическое суммирование входных напряжений (с инвертированием). Количество входных сигналов нетрудно увеличить добавлением дополнительных резисторов (см. пункт и напряжение U_3 на рис. 6.1). Выбирая соотношение сопротивлений резисторов, можно задавать требуемые коэффициенты суммирования.

Отметим также, что описанный сумматор может являться основой для построения и других схем, например, цифро-аналогового преобразователя ЦАП (см. п. 16.1.4), в котором суммируются токи, пропорциональные весам двоичных разрядов кода.

6.1.2. Интегратор

имеет в цепи ОС вместо резистора R_{oc} конденсатор C_{oc} (рис. 6.2), через который протекает ток $I_{oc} \approx I_r$, создавая на конденсаторе падение напряжения, пропорциональное интегралу.

Выходной сигнал при нулевых начальных условиях можно найти, как и в предыдущем пункте, записывая для контура C_{oc} и ΔU уравнение, подобное (6.1) и учитывая, что напряжение на конденсаторе связано с током конденсатора, равным I_{oc} , интегральной зависимостью:

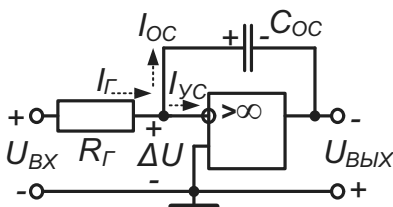


Рис. 6.2

$$U_{Вых} = -U_C - \Delta U \approx -U_C = -\frac{1}{C_{oc}} \cdot \int_0^t I_{oc} dt. \quad (6.4)$$

Считая далее, что ток $I_{oc} \approx I_r$ (см. п. 5.6.5), и подставляя значение $I_r \approx U_{Bx} / R_r$ найдем выходное напряжение:

$$U_{Вых} = -\frac{1}{C_{oc}} \cdot \int_0^t I_{oc} dt = -\frac{1}{R_r C_{oc}} \cdot \int_0^t U_{Bx} dt, \quad (6.5)$$

то есть схема действительно выполняет интегрирование входного сигнала. Например, если входной сигнал постоянен, то напряжение на выходе линейно изменяется со временем.

Отметим вместе с тем, что в настоящее время аналоговое интегрирование при возможности стараются все чаще заменить цифровым.

Подобные устройства, конечно, уже не применяются для решения дифуравнений, но широко используются в преобразователях аналогового сигнала в цифровой код, например, в АЦП двухтактного интегрирования (см. п. 18.5.1 и диаграмму на рис. 6.3), в сигма-дельта АЦП (см. п. 18.6.3), в генераторах линейно изменяющегося напряжения и в других устройствах.

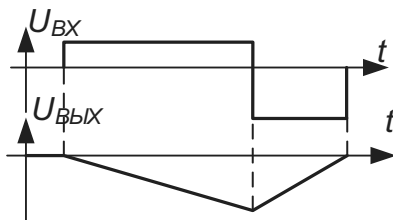


Рис. 6.3

Комбинируя резисторы и конденсаторы в цепи ОС, реализуют устройства с различными свойствами, например, поменяв местами R_f и $C_{ос}$ в схеме рис. 6.2, можно получить схему для дифференцирования входного сигнала.

6.1.3. Преобразователь тока в напряжение

В предыдущих схемах входное напряжение превращалось в ток I_f при помощи сопротивления R_f , но некоторые источники информации имеют информационный сигнал в виде тока, который можно сразу подать для усиления в точку суммирования УПТ без резистора. Примером такого датчика является фотодиод, характеристика которого при освещении сдвигается вниз (рис. 6.4) за счет так называемого фототока, вызванного генерацией носителей заряда под действием света.

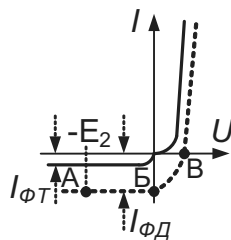


Рис. 6.4

Фотодиод является простейшим датчиком освещенности и может использоваться в охранных системах, в производстве (при подсчете продукции на конвейере), в измерителях числа оборотов, в инфракрасных каналах передачи данных.

Характеристика освещенного фотодиода (рис. 6.4) имеет, как известно [18], два участка – вентильный («А–Б») и гальванический

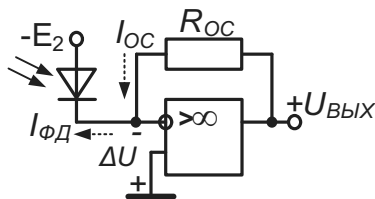


Рис. 6.5

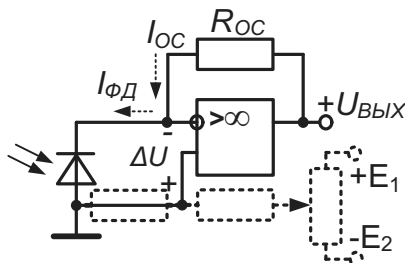


Рис. 6.6

(«Б–В»). Соответственно для усиления фототока можно использовать две схемы. В вентильном режиме на фотодиод подают обратное напряжение (рис. 6.5). В гальваническом режиме можно обойтись без внешнего напряжения – фотодиод сам работает как источник энергии (рис. 6.6).

Отметим, что в схеме рис. 6.5 фотодиод работает в точке «А» своей характеристики, а в схеме рис. 6.6 – в точке «Б» (при практически нулевом напряжении ΔU). Тогда, как видно из рис. 6.4, в обеих схемах фотодиод является источником почти одинакового фототока $I_{\text{ФД}}$. Поэтому выходной сигнал для этих схем описывается общей формулой

$$U_{\text{ВЫХ}} = U_{\text{РОС}} + \Delta U \approx U_{\text{РОС}} \approx I_{\text{ОС}} R_{\text{ОС}} = I_{\text{ФД}} R_{\text{ОС}}, \quad (6.6)$$

полученной, как и прежде, в предположении, что весь фототок уходит в цепь обратной связи.

Для смены полярности выходного напряжения меняют полярность подключения фотодиода (и питающего напряжения, если оно есть).

Для увеличения быстродействия резистор $R_{\text{ОС}}$ иногда шунтируют конденсатором для ускорения перезаряда паразитной емкости фотодиода [2].

Формулу (6.6) можно использовать для разных целей. Например, зная фототок, можно найти $U_{\text{ВЫХ}}$ при известном сопротивлении $R_{\text{ОС}}$. Или, наоборот, при заданных значениях $I_{\text{ФД}}$ и $U_{\text{ВЫХ}}$ рассчитать сопротивление

$$R_{\text{ОС}} = U_{\text{ВЫХ}} / I_{\text{ФД}} \quad (6.7)$$

Иногда фототок неизвестен и приходится определять его из опыта – при известном $R_{\text{ОС}}$ (например, равном 1 МОм). В этом случае освещают фотодиод, измеряют соответствующее приращение $U_{\text{ВЫХ}}$ и находят из (6.6) величину $I_{\text{ФД}}$.

6.1.4. Устранение неинформационного нулевого уровня

Обе рассмотренные схемы имеют определенное неинформационное смещение выходного напряжения, вызываемое посторонней засветкой, остаточным (темновым) током $I_{\text{от}}$ и дрейфом нулевого уровня УПТ. Чтобы избавиться от этого, включают элементы подстройки начального уровня (на рис. 6.6 показаны пунктиром). Однако при изменении температуры или освещенности смещение снова возникает и может нарушить работу устройства.

Для автоматического поддержания определенного уровня $U_{\text{ВЫХ}}$ включают дополнительный усилитель, который сравнивает $U_{\text{ВЫХ}}$ с заданным напряжением покоя $U_{\text{П}}$ и при возникновении разности вырабатывает поправку, которая возвращает выходной уровень к заданному (рис. 6.7). Чтобы дополнительный усилитель не влиял на переменный информационный сигнал, он выполнен в виде интегратора, который практически не усиливает переменную информационную составляющую.

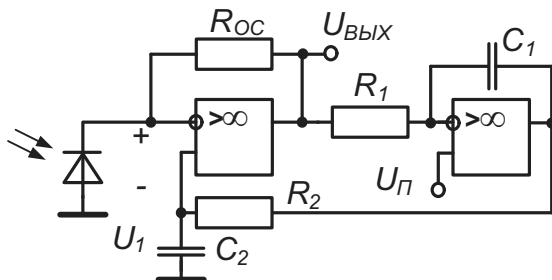


Рис. 6.7

Однако и этого часто бывает недостаточно – при большом коэффициенте усиления наряду с полезным сигналом усиливаются также шумы и помехи. Чтобы уменьшить их влияние, в качестве основного усилителя используют резонансный (полосовой) усилитель, усиливающий только сигналы определенной частоты.

Подобные избирательные фотоприемные устройства, применяемые, например, в инфракрасной связи, иногда реализуют в одной микросхеме, внутри которой размещают фотодиод, резонансный усилитель с однополярным питанием и другие вспомогательные цепи. Такая микросхема может иметь только три вывода - питание, землю и выход.

§6.2. Использование неинвертирующего УПТ

6.2.1. Повторитель напряжения

В некоторых случаях источник информации имеет значительное внутреннее сопротивление и при подключении к нему нагрузки сигнал уменьшается. Чтобы устранить этот недостаток, нагрузку подключают через устройство с большим входным сопротивлением. В качестве последнего может использоваться неинвертирующий УПТ, который, как показано в п. 5.5.3, имеет большое входное сопротивление, зависящее от глубины ОС. В тех случаях, когда нужно получить максимальное значение $R_{вх}$, приносят в жертву коэффициент усиления, и в цепь ОС подают весь выходной сигнал (рис. 6.8).

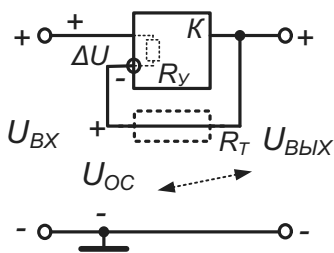


Рис. 6.8

Данная схема имеет единичную ООС, то есть $\beta = 1$. Тогда, согласно (5.41) коэффициент усиления $K_{ос}$ будет равен единице, а величина $K\beta = K$. Поэтому уравнение (5.44) можно переписать для входного сопротивления повторителя в виде:

$$R_{вх} = R_Y \cdot (1 + K). \quad (6.8)$$

Эта формула характеризует главное качество повторителя – его *огромнейшее входное сопротивление*.

Нужно отметить, что под $R_{вх}$ понимается сопротивление приращению сигнала (переменному току). Постоянный входной ток по-прежнему нагружает источник сигнала и создает вместе с напряжением смещения аддитивную погрешность (см. п. 5.7.3). Для минимизации токовой составляющей рекомендуется включать резистор R_T (показан на рис. 6.8 пунктиром), равный выходному сопротивлению источника сигнала.

Мультипликативная погрешность в данном случае, как следует из (5.84), при большой глубине ОС и отсутствии резисторов, чрезвычайно мала.

Повторитель напряжения применяется не только для согласования маломощных датчиков с измерительной схемой, но и в аналоговых коммутаторах (см. п. 15.2.3) и схемах выборки-хранения перед

АЦП (см. п. 15.3.1) для уменьшения влияния сопротивления источника сигнала на точность.

6.2.2. Неинвертирующий сумматор напряжений

Как и прежде (см. п. 6.1.1), сумматор реализуется при помощи резистивной схемы, но используется неинвертирующий УПТ (рис. 6.9), входным для которого является напряжение U'_{BX} .

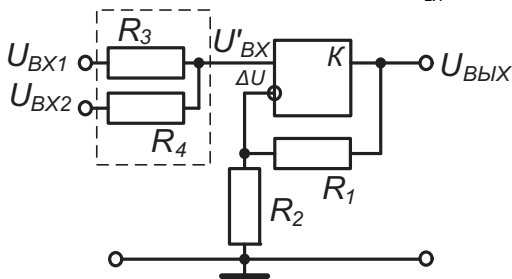


Рис. 6.9

Используя принцип суперпозиции, то есть, считая U_{BX1} и U_{BX2} действующими независимо и поочередно, можно найти напряжение

$$U'_{BX} = U_{BX1} R_4 / (R_4 + R_3) + U_{BX2} R_3 / (R_4 + R_3). \quad (6.9)$$

Или для частного случая, считая, что $R_3 = R_4$, представить в более наглядной форме

$$U'_{BX} = U_{BX1} / 2 + U_{BX2} / 2. \quad (6.10)$$

Выходное напряжение схемы в целом нетрудно найти, используя формулу (5.41) для коэффициента усиления неинвертирующего УПТ.

Отметим, что входное сопротивление для каждого из сигналов напрямую зависит от значений резисторов R_3 и R_4 , которые нельзя выбирать большими, чтобы не увеличивать токовую аддитивную погрешность (см. п. 5.7.3). Поэтому, чтобы не нагружать источники сигналов, их можно подключать через выше рассмотренные повторители с большим входным сопротивлением (см. п. 6.2.1).

6.2.3. Усилитель с программируемым усилением

В многоканальных системах сигналы от датчиков изменяются в большом диапазоне. И для корректной работы АЦП с разными уровнями напряжения (от очень малого до большого) микропроцессорная система должна иметь возможность управления коэффициентом усиления

усилителя. Подобное цифровое управление основано на дискретном изменении при помощи аналоговых ключей (см. п. 15.1.1) сопротивлений цепи ОС в соответствии с формулами (5.41) и (5.58).

Для примера на рис. 6.10 приведена схема неинвертирующего УПТ с тремя возможными коэффициентами усиления.

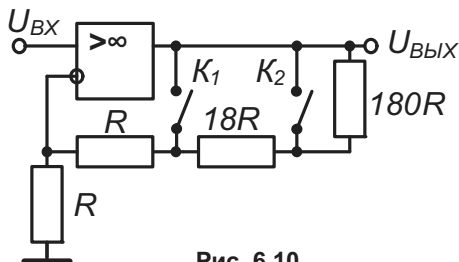


Рис. 6.10

Когда проводит только ключ K_1 , то в соответствии с формулой (5.41) $K_{oc} = 1 + R / R = 2$. Когда проводит только ключ K_2 , то $K_{oc} = 1 + (R + 18R) / R = 20$. Когда оба ключа не проводят, то $K_{oc} = 1 + (R + 18R + 180R) / R = 200$.

Отметим, что дискретно управлять можно и другим резистором цепи ОС, а также реализовать инвертирующий УПТ с программируемым усилением, аналогично ступенчато изменяя R_f и R_{oc} . Отметим только, что аналоговые ключи обладают конечным внутренним сопротивлением в проводящем состоянии (см. п. 15.1.2), и поэтому для уменьшения погрешности, их используют в основном для коммутации более высокоомного резистора ОС. Иногда в программируемых усилителях используют ЦАП, имеющие внутри набор ключей и резисторов (см. п. 16.1.4).

Усилители с цифровым управлением могут выполняться и внутри процессорных микросхем (перед встроенным АЦП), что позволяет упростить внешнее обрамление системы. Нужно только иметь в виду, что входной сигнал для встроенных усилителей (как, впрочем, и для АЦП) обязательно должен быть однополярным (см. §6.6).

§6.3. Дифференциальные УПТ с обратной связью

6.3.1. Усилитель разности однополярных напряжений

На практике бывает необходимо вычислять разность двух напряжений, имеющих одинаковую полярность. Например, напряжение в линии связи $U_{лс}$ при дуплексном обмене между модемами одновре-

менно содержит известный сигнал собственного передатчика $U_{\text{ПРД}}$ и неизвестный сигнал удаленного абонента $U_{\text{Х}}$:

$$U_{\text{ЛС}} = U_{\text{ПРД}} + U_{\text{Х}} \quad (6.11)$$

Для правильной работы приемника нужно вычислить разность известных сигналов:

$$U_{\text{Х}} = U_{\text{ЛС}} - U_{\text{ПРД}} \quad (6.12)$$

Такое вычитание можно осуществить при помощи усилителя разности, приведенного на рис. 6.11, который в отличие от алгебраических сумматоров, рассмотренных ранее (см. п.п. 6.1.1 и 6.2.2), может вычислять разность напряжений одинаковой полярности.

Запишем уравнение для входной цепи, не учитывая $I_{\text{ВХ}}$ и $U_{\text{СМ}}$. Поскольку напряжение между входами ИОУ $\Delta U \approx 0$ (см. п.п. 5.5.4 и 5.6.5), то напряжения в точках «а» и «б», будут практически равны:

$$U_{\text{а}} = U_{\text{б}} \quad (6.13)$$

Проще всего найти величину $U_{\text{б}}$, зависящую только от напряжения $U_{\text{ВХ2}}$ (как напряжение на выходе делителя из резисторов R_5 и R_6):

$$U_{\text{б}} = U_{\text{ВХ2}} R_6 / (R_5 + R_6). \quad (6.14)$$

Напряжение $U_{\text{а}}$ зависит от двух напряжений $U_{\text{ВХ1}}$ и $U_{\text{ВЫХ}}$, поэтому найдем его, используя принцип наложения и считая напряжения независимыми и действующими поочередно:

$$U_{\text{а}} = U_{\text{ВХ1}} R_8 / (R_7 + R_8) + U_{\text{ВЫХ}} R_7 / (R_7 + R_8). \quad (6.15)$$

Подставляя значения напряжений в (6.13) и приняв для простоты $R_5 + R_6 = R_7 + R_8$, получаем уравнение, решая которое, можно найти выходное напряжение:

$$U_{\text{ВЫХ}} = U_{\text{ВХ2}} R_6 / R_7 - U_{\text{ВХ1}} R_8 / R_7 \quad (6.16)$$

Таким образом, рассмотренная схема усиливает разность двух однополярных напряжений, причем коэффициент усиления зависит, как и прежде, от соотношения сопротивлений. В частном случае можно выбрать $R_6 = R_8 = K_{\text{ТР}} R_7$, тогда

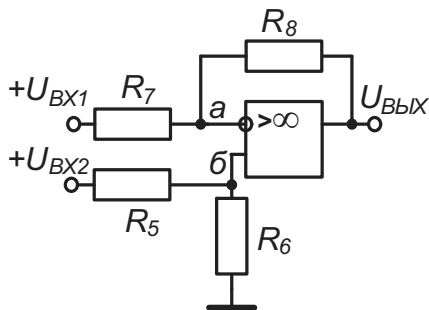


Рис. 6.11

$$U_{ВЫХ} = K_{TP} (U_{ВХ2} - U_{ВХ1}), \quad (6.17)$$

где $K_{TP} = R_8 / R_7 = R_6 / R_5$ – требуемый коэффициент усиления разности. В дальнейших схемах усилителя разности для определенности будем полагать $R_5 = R_7 = R$, а $R_6 = R_8 = K_{TP}R$.

Некоторым недостатком такого усилителя разности является то, что входное сопротивление напрямую зависит от значений резисторов, которые нежелательно выбирать большими, чтобы не увеличивать токовую аддитивную погрешность (см. п. 5.7.3).

6.3.2. Усилитель разности с повышенным входным сопротивлением

В тех случаях, когда необходимо повышенное входное сопротивление усилителя разности, (например, при усилении сигналов от датчиков с большим внутренним сопротивлением), рекомендуется применять имеющие очень большое $R_{ВХ}$ повторители напряжения (см. п. 6.2.1).

Схема в целом при этом усложняется незначительно (рис. 6.12), потому что можно использовать микросхему, в которой все три ИОУ выполнены на одном кристалле.

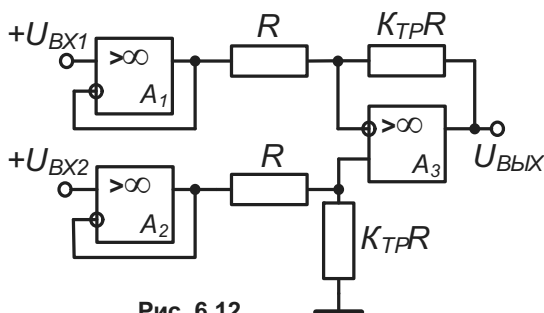


Рис. 6.12

6.3.3. Усилитель разности с повышенным коэффициентом усиления

Усилительные свойства ИОУ в повторителях предыдущей схемы не реализуются, поэтому при необходимости вместо повторителей можно использовать неинвертирующие УПТ (рис. 6.13), которые не только обладают достаточно большим входным сопротивлением, но и обеспечивают дополнительное усиление сигнала.

В этой схеме коэффициент усиления предварительных каскадов на ИОУ A_1 и A_2 одинаков и равен $K_{пр} = 1 + R_1 / R_2$ (см. п. 5.5.2). Затем

разность напряжений усиливается каскадом на ИОУ A_3 дополнительно в K_{TP} раз.

Следует отметить, что коэффициент усиления K_{TP} не может быть очень большим, что связано с тем, что для правильного вычисления разности ИОУ A_1 и A_2 должны работать в линейном режиме, то есть

$$K_{TP} \leq U_M / U_{ВХМ} \quad (6.18)$$

где U_M – максимально возможное выходное напряжение ИОУ, которое для приближенной оценки можно считать равным напряжению питания схемы, $U_{ВХМ}$ – максимальный уровень напряжений $U_{ВХ1}$ и $U_{ВХ2}$, определенные в техническом задании.

Подобная симметричная схема, приведенная на рис. 6.13, является прообразом, так называемого измерительного усилителя разности (см. далее), который иногда называют также дифференциальным или инструментальным усилителем [2].

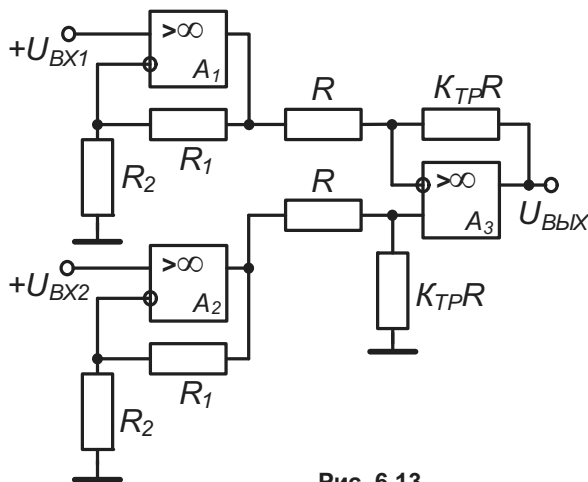


Рис. 6.13

6.3.4. Измерительный усилитель разности

Симметричность входной цепи вышерассмотренной схемы позволяет в предварительных УПТ объединить два резистора ОС и использовать общий резистор R_2 для задания коэффициента усиления одновременно двух УПТ (рис. 6.14).

Такая особенность оказывается весьма полезной при исполнении такого усилителя в виде одной микросхемы, когда все резисторы, кроме R_2 , выполняются твердотельными внутри кристалла. И доступ

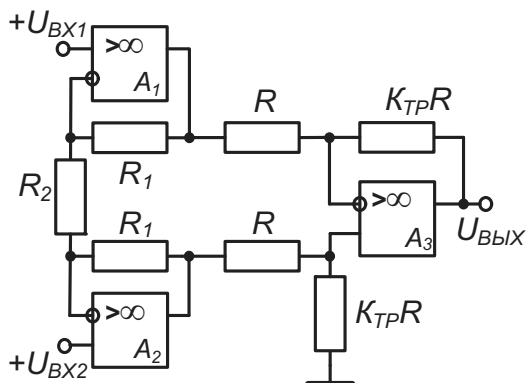


Рис. 6.14

к ним для изменения общего коэффициента усиления невозможен. Резистор R_2 в этом случае подключается к внешним выводам микросхемы и позволяет задать необходимый коэффициент усиления.

Подобные инструментальные усилители широко используются для усиления маломощных сигналов на фоне больших синфазных составляющих, например, в мостовых схемах или электрокардиографии.

§6.4. Усилитель с мощным выходным каскадом

6.4.1. Общие сведения

Микросхемы ИОУ общего назначения имеют типовой ток нагрузки около 5..20 мА. Если требуются большие токи, то применяют специальные мощные ИОУ [19]. Но в ряде практических случаев можно обойтись добавлением к обычному ИОУ несложного усилительного каскада на внешнем транзисторе.

Подобный выходной каскад должен быть по своим свойствам близок к идеальному источнику напряжения, то есть в первую очередь иметь малое выходное сопротивление. Поэтому в большинстве случаев в выходном каскаде применяют различные варианты эмиттерного повторителя (см. п. 5.2.8), поскольку по этому параметру он является лучшим из одиночных каскадов.

Для согласования с низкоомной нагрузкой в мощных каскадах иногда применяют трансформаторы, которые обеспечивают макси-

мальное усиление по мощности [5]. Но трансформаторы являются нетехнологичными и трудоемкими в изготовлении элементами, поэтому ниже рассмотрены чаще используемые на практике бестрансформаторные каскады.

В мощных каскадах транзисторы могут работать в различных режимах, условно называемых классами, которые принято обозначать латинскими буквами *A, B, C, D*.

6.4.2. Эмиттерный повторитель класса A

Входное напряжение может иметь произвольную полярность, например, быть синусоидальным (рис. 6.15). Поэтому для получения двуполярного выходного сигнала сопротивление нагрузки R_3 эмиттерного повторителя, приведенного на рис. 6.16, должно быть подключено к источнику отрицательного напряжения E_2 .

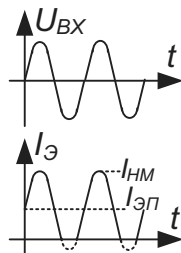


Рис. 6.15

Отметим далее, что токи транзистора и резистора всегда однонаправлены, тогда как ток нагрузки изменяет свое направление в такт с входным сигналом. Причем положительная полуволна тока I_H формируется за счет увеличения тока I_3 протекающего через транзистор от $+E_1$, а отрицательная – за счет уменьшения тока I_{R3} , протекающего через резистор R_3 от $-E_2$.

Если $U_{BX} = 0$, то переменный ток нагрузки I_H и U_{BbIX} также равны нулю, а ток эмиттера I_3 и I_{R3} равны току покоя $I_{ЭП}$

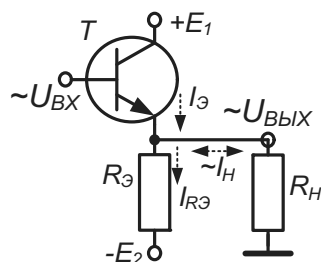


Рис. 6.16

$$I_{ЭП} = I_{R3П} = (E_2 - U_{BbIX}) / R_3 = E_2 / R_3. \quad (6.19)$$

Из рис. 6.15 видно, что для работы без искажений ток покоя $I_{ЭП}$ обязательно должен быть больше амплитуды тока нагрузки I_{HM}

$$I_{ЭП} \geq I_{HM} \quad (6.20)$$

Тогда, из уравнения (6.19), с учетом (6.20), следует расчетное соотношение для выбора резистора

$$R_3 = E_2 / I_{ЭП} \leq E_2 / I_{HM} \quad (6.21)$$

Неравенство (6.20) иллюстрирует главный недостаток режима A – даже при отсутствии входного сигнала через транзистор течет ток $I_{ЭП}$ и соответственно выделяется мощность

$$P = U_{кз} I_{эп} \approx E_1 I_{эп} \quad (6.22)$$

Если предположить, что $I_{эп} = I_{нм} = 1$ А, то при $E_1 = 5$ В и отсутствии сигнала на транзисторе выделяется мощность $P = 5 \cdot 1 = 5$ Вт, что требует установки на транзистор теплоотводящего радиатора.

Не останавливаясь на определении коэффициента полезного действия (КПД) усилительного каскада класса А, отметим, что он имеет величину менее 50% [5]. Наряду с указанным недостатком следует отметить и *достоинство* такого каскада – транзистор работает на линейном участке входной характеристики (рис. 6.17), чем обеспечиваются достаточно малые нелинейные искажения по сравнению с энергетически более выгодным каскадом класса В (см. далее).

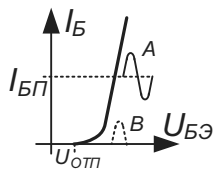


Рис. 6.17

6.4.3. Двухтактный эмиттерный повторитель класса В

Чтобы повысить КПД мощного каскада, используется режим, в котором объединены два разнополярных эмиттерных повторителя, работающих на общую нагрузку (рис. 6.18). В этом режиме транзисторы имеют нулевой ток покоя, что обеспечивает высокий КПД. А благодаря разному типу проводимости усиливается как положительная полуволна (транзистор T_1), так и отрицательная (T_2), причем на общей нагрузке эти полуволны суммируются (рис. 6.19), образуя двупольный выходной сигнал.

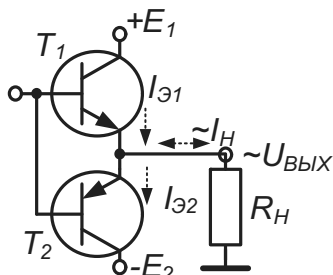


Рис. 6.18

Иногда суммирование полуволн осуществляется магнитным путем, при этом используются обмотки трансформатора, подключаемые к однотипным транзисторам [5].

6.4.3. Достоинства и недостатки каскада класса В

Главным *преимуществом* является то, что при отсутствии сигнала энергия от источника питания транзисторами выходного каскада практически не потребляется.

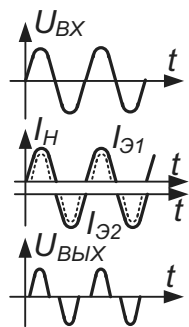


Рис. 6.19

При наличии сигнала каскад работает в два такта, транзисторы поочередно усиливают ток и поочередно обесточены. Причем непроводящий транзистор находится в режиме глубокой отсечки тока, который обеспечивается за счет обратного напряжения $U_{БЭ}$, действующего на эмиттерном переходе транзистора другого типа проводимости.

КПД такой схемы поэтому существенно больше, чем у каскада класса А, и может достигать 80% [5].

Но двухтактный каскад имеет и *недостаток* – транзисторы работают на начальном участке резко нелинейной входной характеристики (см. рис. 6.17), на которой есть некоторое напряжение отпирания $U_{отп}$ [28]. Поэтому при малых входных сигналах ток транзистора появляется не сразу, и на выходе возникают искажения в виде ступенек (см. диаграмму для $U_{ВЫХ}$ на рис. 6.19).

Для уменьшения искажений, можно приоткрыть каждый из транзисторов, задав небольшой ток покоя (перейти в комбинированный режим АВ), но КПД при этом, конечно, уменьшится.

В некоторых случаях, когда нагрузка нечувствительна к нелинейным искажениям, например, в мощных усилителях следящих систем, управляющих двигателями, используется более грубый усилительный режим С, в котором для увеличения КПД транзисторы, наоборот, подзапираются.

Основным средством уменьшения нелинейных искажений является введение *общей* ООС, которая уменьшает в $K\beta$ раз все искажения [18].

6.4.4. Усилитель с мощным выходным каскадом

Двухтактный повторитель, используемый в качестве мощного выходного каскада, не изменяет фазу сигнала, поэтому, обратная связь в усилителе подается, как и прежде, на инвертирующий вход ИОУ, но охватывает не только ИОУ, но и выходной каскад (рис. 6.20), компенсируя его существенные нелинейные искажения.

Временные диаграммы, поясняющие процесс уменьшения искажений на выходе усилителя, приведены на рис. 6.21. Вблизи нуля

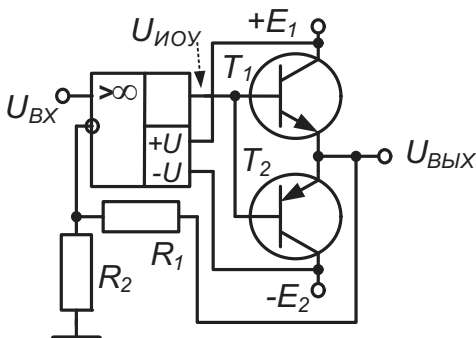


Рис. 6.20

сигнал $U_{\text{ИОУ}}$ будет меньше $U_{\text{отп}}$ и напряжение на выходе будет отсутствовать (см. «ступеньку» на рис. 6.19). Это означает, что сигнала ООС также не будет, и ИОУ поэтому будет работать без ООС, то есть с огромным коэффициентом усиления, что вызовет появление напряжения $U_{\text{ИОУ}}$ приподнятого на величину $U_{\text{отп}}$.

Таким образом, схема скомпенсирует искажения, вносимые выходным каскадом, благодаря ООС, которая стремится поддержать выходной сигнал в точности пропорциональным входному.

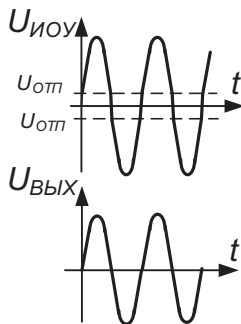


Рис. 6.21

Следует отметить, что при работе на повышенных частотах ИОУ должен иметь хорошие частотные свойства, чтобы успевать отрабатывать искажения типа «ступеньки», содержащие гармоники основной частоты.

Мощные каскады, достаточно часто работают на границе допустимых параметров и нуждаются поэтому в защите выходных транзисторов от перегрузок по току, возникающих, например, при случайном коротком замыкании нагрузки. Простейшей схемой защиты для выходных транзисторов может являться ограничитель тока, рассмотренный в п. 3.3.4.

6.4.6. Выбор типа транзисторов выходного каскада

Выбор типа транзисторов происходит по току (больше $I_{\text{НМ}}$), напряжению (больше $E_1 + E_2$) и частотным свойствам. Кроме того важным параметром при выборе является коэффициент усиления транзистора

$$B \geq I_{\text{НМ}} / I_{\text{ИОУм}} \quad (6.23)$$

При необходимости в мощном каскаде можно использовать составные транзисторы (рис. 6.22), которые имеют увеличенный коэффициент усиления. Но в этом случае нежелательно применять составные

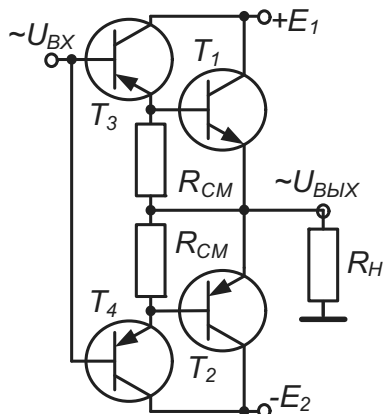


Рис. 6.22

транзисторы, выполненные в одном корпусе, так как к ним нельзя подключить резисторы $R_{см}$, служащие для лучшего запираания транзисторов T_1 и T_2 за счет создания пути для протекания их обратного тока $I_{ко}$ (см. п. 1.6.4).

В последнее время все чаще в выходных каскадах используются мощные полевые транзисторы [2].

6.4.7. Усилительный каскад класса D

Мощный каскад в этом режиме (рис. 6.23) использует времяимпульсное представление информации, которое часто называют широтно-импульсной модуляцией (ШИМ) сигнала [2], когда входное напряжение преобразуется в длительность (ширину) прямоугольного импульса $U_{ШИМ}$ (см. рис. 6.24) с коэффициентом пропорциональности $K_{ШИМ}$:

$$t_{и} = K_{ШИМ} \cdot U_{ВХ} \quad (6.24)$$

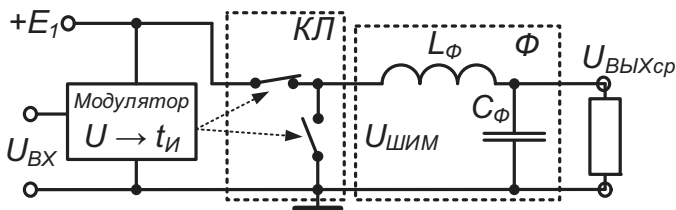


Рис. 6.23

В этом случае для усиления тока в нагрузке после преобразователя можно использовать мощные ключи (на рис. 6.23 они обозначены КП), обладающие КПД, близким к 100% [2]. Обратное преобразование импульсного сигнала в информационный обычно реализуется при помощи сглаживающего фильтра Φ , который осредняет импульсный сигнал, подавляет высшие гармоники и выделяет информационную среднюю составляющую, пропорциональную длительности импульса, а значит и входному напряжению:

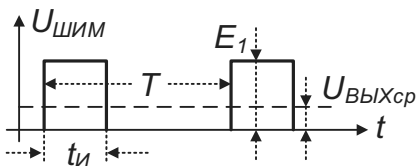


Рис. 6.24

$$U_{ВЫХср} = E_1 t_{и} / T = E_1 K_{ШИМ} \cdot U_{ВХ} / T. \quad (6.25)$$

Для двуполярного входного сигнала дополнительно используются пара ключей и отрицательный источник питания.

К недостаткам каскада класса *D* можно отнести невысокую частоту усиливаемого сигнала, повышенный уровень радиопомех и шумов (несглаженные пульсации) в выходном напряжении. Поэтому подобные каскады применяют в основном при управлении двигателями, в схемах мощных низкочастотных генераторов и изредка в усилителях звука.

Отметим также, что времяимпульсное представление информации может использоваться и для реализации ЦАП и АЦП (см. п.п. 16.3.1 и 18.6.2).

§6.5. Усилители переменного напряжения

Далее для простоты будут рассмотрены только усилители низкой частоты (УНЧ), в которых можно не учитывать инерционность ИОУ.

6.5.1. Расчет разделительного конденсатора

В тех случаях, когда информация переносится сигналом переменного напряжения, усилитель имеет разделительные элементы (трансформаторы или чаще конденсаторы), не пропускающие неинформативную постоянную составляющую. А переменную составляющую заданного частотного диапазона разделительный элемент наоборот должен пропускать практически без ослабления.

Последнее условие для иллюстративного последовательного включения разделительного конденсатора (рис. 6.25) означает, что емкостное сопротивление X_C конденсатора C_P на минимальной рабочей частоте $F_{\text{РМИН}}$ должно быть много меньше сопротивления резистора $R_{\text{ПС}}$, включенного последовательно с конденсатором:

$$X_C = 1 / (2\pi \cdot F_{\text{РМИН}} \cdot C_P) \ll R_{\text{ПС}} \quad (6.26)$$

Не останавливаясь на расчете коэффициента частотных искажений [5], укажем, что для большинства практических случаев бывает достаточно, чтобы емкостное сопротивление было, например, в 10 раз меньше активного. Тогда емкость конденсатора можно рассчитать по упрощенной формуле:

$$C_P \geq 10 / (2\pi \cdot R_{\text{ПС}} \cdot F_{\text{РМИН}}) \approx 2 / (R_{\text{ПС}} \cdot F_{\text{РМИН}}) \quad (6.27)$$

Пусть $R_{\text{ПС}} = 1 \text{ кОм}$, $F_{\text{РМИН}} = 1000 \text{ Гц}$, тогда $C_P = 2 / (1000 \cdot 1000) = 2 \cdot 10^{-6} \text{ Ф} = 2 \text{ мкФ}$. Учитывая технологический разброс и изме-

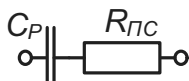


Рис. 6.25

нение емкости с температурой, выбираем значение емкости из стандартного ряда [5] с запасом, то есть $C_p = 5 \cdot 10$ мкФ.

6.5.2. Инвертирующий УНЧ

Схема УНЧ (рис. 6.26) отличается от схемы инвертирующего УПТ только наличием разделительных конденсаторов. Причем C_r отделяет постоянную составляющую входного сигнала, а C_n не пропускает на нагрузку неинформативный нулевой уровень УПТ.

Поскольку емкостные сопротивления были выбраны при расчете значительно меньшими сопротивлений соответствующих им резисторов, то конденсаторы при определении коэффициента усиления схемы можно не учитывать. Поэтому для УНЧ можно использовать формулу (5.58), то есть $K_{oc} = -R_{oc} / R_r$.

Аналогично и входное сопротивление усилителя, как и прежде, равно R_r в соответствии с формулой (5.60).

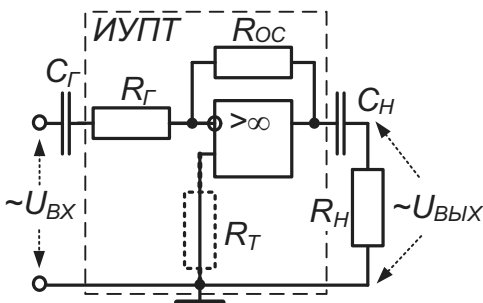


Рис. 6.26

6.5.3. Влияние нулевого уровня УПТ

Несмотря на то, что постоянная составляющая от УПТ ($U_{ВЫХ0}$) отсекается конденсатором C_r , она оказывает влияние на работу УНЧ.

Из рис. 6.27 видно, что при достаточно большом выходном сигнале возникает несимметричное искажение, связанное с величиной нулевого уровня $U_{ВЫХ0}$ и максимально возможным выходным напряжением ИОУ $U_{ИОУМ}$. Считая, что величина $U_{ИОУМ}$ примерно равна напряжению питания ($E_1 = |E_2| = E$), можно найти максимально возможную амплитуду сигнала на нагрузке:

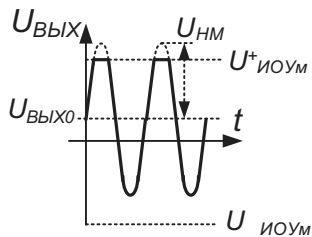


Рис. 6.27

$$U_{НМ} \leq U_{ИОУМ} - U_{ВЫХ0} \approx E - U_{ВЫХ0} \quad (6.28)$$

Для расчета нулевого уровня можно использовать формулу (5.78), в которой нужно учесть, что благодаря конденсатору C_r который не

пропускает постоянный ток, в схеме существует *единичная* ООС по *постоянному* току, то есть $\beta = 1$ и $K_{OC} = 1$. Поэтому напряжение смещения не усиливается, а влияние входных токов минимизируется при помощи $R_T = R_{OC}$. Поэтому формула (5.78) для расчета (при наличии резистора R_T) упрощается:

$$U_{ВЫХО} = \pm U_{CM} K_{OC} \pm I_P \cdot R_{OC} = \pm U_{CM} 1 \pm I_P \cdot R_{OC} \approx \pm I_P \cdot R_{OC}. \quad (6.29)$$

Отметим, что в усилителях переменного напряжения можно использовать однополярное питание ИОУ (см. п. 6.6.3).

6.5.4. Порядок расчета инвертирующего УНЧ

Поскольку схема выполнена на основе инвертирующего УПТ, то вначале (при заданных K_{OC} и R_{BX}) определяют его сопротивления $R_T \geq R_{BX}$ и $R_{OC} = R_T \cdot K_{OC}$ (см. п. 5.6.6).

Для уменьшения токовой составляющей нулевого уровня используют дополнительный резистор $R_T = R_{OC}$.

Затем производят расчет емкостей разделительных конденсаторов по формуле (6.27), учитывая, что C_T включено последовательно с R_T а $C_H - с R_H$.

После этого вычисляют максимальное значение нулевого уровня $U_{ВЫХО}$ по формуле (6.29), при необходимости оценивают его температурный дрейф и проверяют возможное неискаженное значение U_{HM} по формуле (6.28), чтобы оно было больше, чем заданное номинальное выходное напряжение $U_{ВЫХН}$.

Инвертирующий УНЧ имеет не слишком большое входное сопротивление, равное R_T . А увеличивать значение R_T нежелательно, чтобы не уменьшать глубину ООС (см. п. 5.6.4).

6.5.5. Неинвертирующий УНЧ

Для увеличения входного сопротивления УНЧ можно использовать неинвертирующий УПТ (рис. 6.28), который, как известно, обладает повышенным входным сопротивлением (см. п. 5.5.3). Как и прежде, УНЧ должен иметь разделительные конденсаторы на входе и

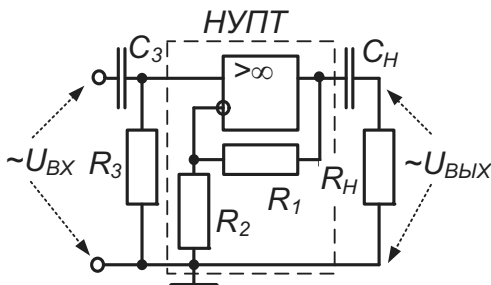


Рис. 6.28

на выходе, емкостные сопротивления которых на рабочих частотах при надлежащем расчете учитывать не будем (см. п. 6.5.1).

Но при включении на входе разделительного конденсатора C_3 прерывается цепь протекания постоянного тока входного тока ИОУ, что делает первый дифференциальный каскад ИОУ полностью неработоспособным (см. п. 5.4.1). И чтобы создать путь для протекания постоянной составляющей тока $I_{вх}$, приходится использовать дополнительный резистор R_3 , который пропускает и часть переменного тока от источника сигнала, уменьшая тем самым входное сопротивление УНЧ. Поэтому последнее оказывается равным параллельному соединению R_3 и входному сопротивлению неинвертирующего УПТ, которое рассчитывается по формуле (5.44):

$$R_{вхУНЧ} = R_3 \parallel (R_y \cdot (1 + K\beta)). \quad (6.30)$$

Сопротивление R_3 для минимизации токовой составляющей нулевого уровня (см. п. 5.7.3) выбирают примерно равным $R_1 \parallel R_2$. А поскольку обычно $R_1 \gg R_2$ (см. п. 5.6.7), то $R_1 \parallel R_2 \approx R_2$, то есть $R_3 \approx R_2$.

Кроме того, учтем, что входное сопротивление неинвертирующего УПТ, вычисляемое по формуле (5.44), во много раз больше входного сопротивления ИОУ и тем более значения величины R_2 , а значит и R_3 . Отсюда следует, что

$$R_3 \ll R_y \cdot (1 + K\beta). \quad (6.31)$$

Поэтому уравнение (6.33) можно переписать в виде

$$R_{вхУНЧ} = R_3 \parallel (R_y (1 + K\beta)) \approx R_3. \quad (6.32)$$

И отсюда следует вывод, что входное сопротивление УНЧ получается практически равным величине R_3 .

Поэтому неинвертирующий УНЧ по сравнению с инвертирующей схемой не имеет ощутимого преимущества по величине входного сопротивления.

Для коэффициента усиления неинвертирующего УНЧ можно использовать формулу (5.41), то есть $K_{ос} \approx 1 + R_1/R_2$. Порядок расчета сопротивлений этих резисторов изложен в п. 5.5.5, выбор емкостей – в п. 6.5.1.

Основным недостатком неинвертирующего УНЧ является то, что его нулевой уровень оказывается значительно большим, чем в инвертирующем УНЧ, поскольку, в отличие от предыдущей схемы, ОС для постоянного тока такая же, как и для переменного. Поэтому напряжение смещения ИОУ в соответствии с формулой (6.78) усиливается

в K_{OC} раз, а токовая составляющая минимизируется при правильном выборе R_3 :

$$U_{ВЫХО} = \pm U_{CM} (1 + R_1/R_2) \pm I_P \cdot R_3. \quad (6.33)$$

Например, при $U_{CM} = 10$ мВ, $K_{OC} = 200$, $I_P = 0,01$ мкА, $R_3 = 10$ кОм получается достаточно большое значение $U_{ВЫХО} = 10 \cdot 10^{-3} \cdot 200 + 0,01 \cdot 10^{-6} \cdot 10 \cdot 10^3 \approx 2$ В, не позволяющее получить на выходе большой размах неискаженного выходного напряжения (см. п. 6.5.3).

Указанный недостаток устранен в следующей схеме.

6.5.6. Неинвертирующий УНЧ с раздельной ООС по переменному и постоянному токам

Чтобы уменьшить нулевой уровень в предыдущей схеме, нужно усилить обратную связь по постоянному току, например, сделать так, чтобы через резистор R_2 постоянный ток не протекал. Для этого последовательно с резистором R_2 включают конденсатор C_2 (рис. 6.29), емкостное сопротивление которого на рабочей частоте много меньше значения R_2 . Поэтому коэффициент усиления на переменном токе практически не изменится. Он будет таким же, как в предыдущей схеме, и равным $K_{OC} \approx 1 + R_1/R_2$.

Но для постоянного тока конденсатор C_2 представляет разрыв, в схеме будет действовать единичная ООС по постоянному току, и напряжение U_{CM} ИОУ поэтому усиливаться не будет. Вследствие этого нулевой уровень можно будет оценивать по упрощенной формуле (6.29), а токовую составляющую нулевого уровня минимизировать, выбрав $R_3 = R_1$. С учетом этого получим (при $R_{OC} = R_1$)

$$U_{ВЫХО} \approx \pm I_P \cdot R_1. \quad (6.34)$$

Обобщая изложенное, отметим, что нулевой уровень в этой схеме УНЧ имеет небольшую величину, так же как и в инвертирующем УНЧ (рис. 6.26), что позволяет получить максимально возможное напряжение на выходе (см. п. 6.5.3). С другой стороны, значение R_3 , определяющее входное сопротивление схемы, можно выбрать равным R_1 , которое значительно больше, чем R_2 , то есть схема имеет наибольшую величину $R_{ВХ}$ по сравнению с ранее рассмотренными УНЧ.

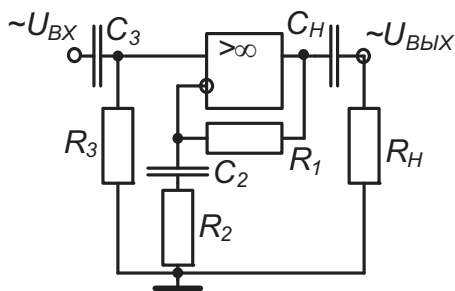


Рис. 6.29

Но величина R_1 ограничена сверху токовой составляющей нулевого уровня и зависимостью коэффициента β от $R_{\text{ИОУ}}$ (см. п. 5.6.4). Поэтому R_3 и R_1 имеют сопротивления от сотен кОм до единиц МОм .

Чтобы получить большие значения $R_{\text{ВХ}}$, нужно уменьшить переменный ток, который ответвляется через резистор R_3 и уменьшает входное сопротивление схемы.

Ток, как известно из закона Ома, определяется отношением напряжения к сопротивлению. Если нет возможности увеличить значение R_3 , то следует уменьшить разность потенциалов на этом резисторе. Сделать это можно подключив нижний вывод резистора R_3 к точке с потенциалом близким к потенциалу верхнего вывода, то есть к величине $U_{\text{ВХ}}$.

6.5.6. Неинвертирующий УНЧ с повышенным входным сопротивлением

В этой схеме (рис. 6.30) на нижний вывод резистора R_3 подается напряжение, которое образуется на резисторе R_2 и которое примерно равно напряжению обратной связи, потому что конденсатор C_2 на рабочей частоте эквивалентен перемычке.

Учитывая далее, что $U_{\text{ОС}} \approx U_{\text{ВХ}}$ (см. п. 5.5.4), можно считать потенциалы точек «а» и «б» примерно равными, поэтому переменный ток в резистор R_3 практически ответвиться не будет и исчезнет шунтирующее влияние R_3 на входное сопротивление.

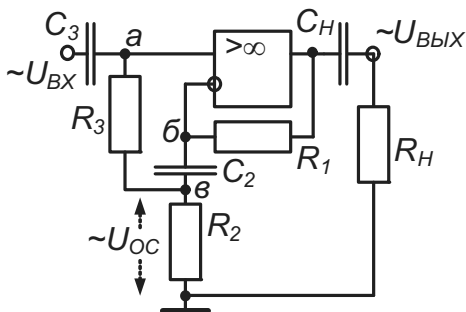


Рис. 6.30

Для постоянного входного тока ИОУ цепь практически останется прежней, так как конденсатор C_2 для постоянного тока представляет разрыв, сопротивление $R_2 \ll R_1$, а $R_1 \approx R_3$. То есть резистор R_2 много меньше R_3 , на протекание постоянного входного тока ИОУ почти не влияет.

Оценим входное сопротивление рассматриваемой схемы, используя формулу (6.30), в которой вместо R_3 будем использовать

$$R_{\text{ЭВ}} = U_{\text{ВХ}} / I_{R_3} \quad (6.35)$$

Тогда

$$R_{BX} = R_{ЭКВ} \parallel (R_{ИОУ} (1 + K\beta)). \quad (6.36)$$

Для оценки $R_{ЭКВ}$ запишем уравнение для тока

$$I_{R3} = (\varphi_a - \varphi_c) / R_3 = (U_{BX} - U_{OC}) / R_3 = (U_{BX} - \beta U_{ВЫХ}) / R_3. \quad (6.37)$$

Подставляя полученное значение тока I_{R3} в (6.38) и учитывая, что $U_{ВЫХ} = K_{OC} \cdot U_{BX}$, где K_{OC} определяется по формуле (5.10), найдем

$$R_{ЭКВ} = U_{BX} R_3 / (U_{BX} - K_{OC} \beta U_{BX}) = R_3 (1 + K\beta), \quad (6.38)$$

то есть эквивалентное сопротивление оказалось в $K\beta$ раз больше, чем значение R_3 .

Пусть для примера $R_3 = 100 \text{ кОм}$, а $K\beta = 100$, тогда $R_{ЭКВ} = 100 \cdot 10^3 \cdot 100 = 100 \text{ МОм}$, то есть величина $R_{ЭКВ}$ стала сравнимой со входным сопротивлением неинвертирующего УПТ.

Учитывая формулу (6.38), выражение (6.36) можно переписать в виде

$$R_{BX} = (1 + K\beta) (R_3 \parallel R_{ИОУ}) = (1 + K\beta) R_3 R_{ИОУ} / (R_3 + R_{ИОУ}). \quad (6.39)$$

§6.6. Однополярное питание усилителей

6.6.1. Смещение информационной составляющей

Однополярное питание ИОУ все чаще используется для снижения энергопотребления и упрощения взаимодействия с микропроцессорными системами, которые имеют только один положительный источник питания, например, 5 В, 3,3 В или даже меньше.

Отметим, что при однополярном питании приходится мириться с уменьшенным диапазоном изменения выходного сигнала.

В этом случае информационную составляющую усилительного устройства $U_{ВЫХ}$ необходимо сместить вверх (рис. 6.31), чтобы при любых значениях U_{BX} значения $U_{ВЫХ}$ помещались только в диапазоне от 0 до $+E_1$. И далее при обработке однополярного напряжения $U_{ВЫХ}$ отнимают напряжение сдвига $E_{сдв}$ и получают составляющую, пропорциональную только U_{BX} .

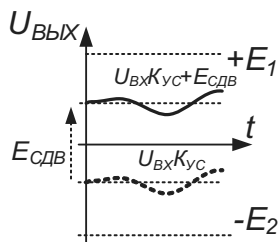


Рис. 6.31

Делают подобное смещение $U_{ВЫХ}$ путем добавления к входному сигналу некоторого постоянного напряжения с использованием резистивного сумматора или дифференциального входа. Для примера, на рис. 6.32 показан инвертирующий УПТ с однополярным питанием.

ем, в котором на неинвертирующий вход ИОУ подан смещающий сигнал $U_{\text{сдвех}}$, образованный на выходе резистивного делителя R_{C1} и R_{C2} , подключенного к единому положительному источнику питания $+E_1$.

Примем для простоты $U_{\text{вх}} = 0$, тогда для сигнала $U_{\text{сдвех}}$ схема будет представлять неинвертирующий УПТ с коэффициентом усиления по формуле (5.41). Поэтому можно найти требуемое входное напряжение $U_{\text{сдвех}}$, поделив необходимый выходной сигнал $E_{\text{сдв}}$ на коэффициент усиления УПТ (полагая резисторы ОС уже известными – см. расчет в п. 5.5.5):

$$U_{\text{сдвех}} = E_{\text{сдв}} / (1 + R_{\text{ОС}} / R_{\text{Г}}). \quad (6.40)$$

С другой стороны, напряжение $U_{\text{сдвех}}$ равно выходному напряжению резистивного делителя (без учета входного тока)

$$U_{\text{сдвех}} = E_1 \cdot R_{C2} / (R_{C1} + R_{C2}). \quad (6.41)$$

Приравнявая правые части соотношений и, считая для уменьшения токовой аддитивной погрешности (см. п. 6.3.3) $R_{C2} \approx R_{\text{Г}}$, можно найти R_{C1} .

6.6.2. Учет смещения сигнала при обработке

может производиться различными способами в зависимости от ситуации и имеющихся системных возможностей. Например, при чисто аналоговой обработке можно использовать последовательное потенциальное вычитание напряжений (рис. 6.33). Но при этом сопротивление нагрузки не привязано к общей точке, что не всегда удобно.

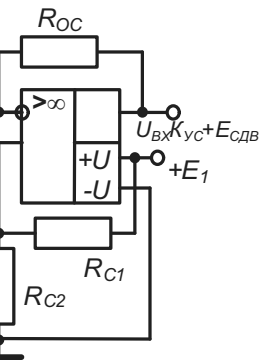


Рис. 6.32

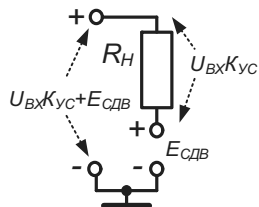


Рис. 6.33

Чтобы получить в этой схеме разностный сигнал относительно «земли» можно применить «летающий конденсатор» (см. п. 15.4.4).

Если усиленный сигнал поступает на АЦП для преобразования в цифровой код, то устранение смещения можно осуществить програм-

мно, вычитая из результата АЦП соответствующий константный код. Но при этом нельзя учесть изменения напряжения сдвига, происходящие, например, при изменении температуры или перенастройке аппаратуры. Поэтому на практике чаще используют схему рис. 6.34 и два входа многоканального АЦП, чтобы оцифровать оба сигнала и затем в программе вычесть их.

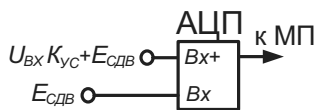


Рис. 6.34

Более удобно для этой цели использовать АЦП с так называемыми дифференциальными входами (рис. 6.34), которые оцифровывают разность напряжений между входами. При этом не нужно в программе предусматривать вычитание и, кроме того, повышается точность вычисления разности.

Проще всего использовать однополярное питание в усилителях переменного напряжения, в которых имеется выходной разделительный конденсатор, который не пропускает постоянное напряжение $E_{сдв}$ и «автоматически» выделяет информативную переменную составляющую.

6.6.3. Инвертирующий УНЧ с однополярным питанием

В качестве примера рассмотрим усилитель переменного напряжения на основе инвертирующего УПТ (рис. 6.35), считая для простоты, что нулевой уровень ($U_{Вых0}$) последнего достаточно мал. Поэтому для данного случая можно задаться значением $E_{сдв} = E_1/2$, смещающим постоянную составляющую $U_{Вых}$ на середину напряжения питания.

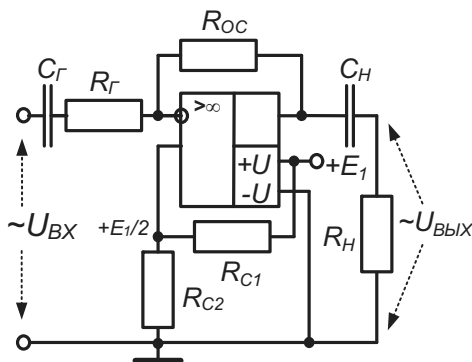


Рис. 6.35

Для этого подадим, как и ранее (см. п. 6.6.1) сигнал $U_{сдввх}$ на неинвертирующий вход ИОУ. Но величина его будет не такой, как в п. 6.6.1.

Дело в том, что в схеме существует 100% ОС по постоянному напряжению (конденсатор $C_Г$ не пропускает постоянный ток), и коэффициент усиления постоянного напряжения $U_{сдввх}$ равен поэтому единице.

Отсюда для данной схемы

$$U_{\text{сдвех}} = E_{\text{сдв}} = E_1 / 2. \quad (6.42)$$

Чтобы поделить напряжение питания пополам, сопротивления делителя должны быть одинаковыми, то есть $R_{c1} = R_{c2}$.

Далее, чтобы уравнивать пути протекания входных токов ИОУ и уменьшить токовую составляющую нулевого уровня (см. п. 5.7.3), нужно, чтобы их параллельное соединение $R_{c1} \parallel R_{c2}$ было равно R_{oc} . Из этого условия выбираем $R_{c1} = R_{c2} = 2R_{oc}$.

Все остальные расчеты подобны приведенным в п. 6.5.4.

6.6.4. Параметры ИОУ при однополярном питании

Принципиальных проблем при питании ИОУ от одного источника не возникает, поскольку ИОУ усиливает разностный сигнал между входами, и потенциалы входов могут отсчитываться относительно любой точки. Недаром у ИОУ нет вывода «общая точка», в просторечии «земли». Но для определенности такую точку отсчета («локальную землю», $E_{\text{сдв}}$) можно создать искусственно между потенциалами однополярного питания. Но при этом и выходной сигнал нужно также вычислять относительно созданной точки отсчета (см. п. 6.6.2).

Для одиночного ИОУ потенциал «локальной земли» можно получить на выходе делителя, например, из одинаковых резисторов, подключенных к однополярному источнику питания E_p , как в п. 6.6.3. В этом случае для ИОУ, как бы создается двуполярное питание $\pm 0,5E_p$, и нужно помнить, что не все ИОУ могут работать при пониженном питании. Поэтому следует проверить в справочнике минимальный уровень питания выбранного типа ИОУ или принимать дополнительные меры по увеличению напряжения питания за счет импульсного инвертора полярности (см. п. 6.6.5).

При пониженном напряжении питания обязательно нужно учитывать, что некоторые параметры ИОУ ухудшаются и выходят за нормируемые границы. Например, при малых напряжениях питания напряжение смещения увеличивается, а коэффициент усиления уменьшается.

Другим обстоятельством, которое нужно помнить при использовании однополярного режима, является то, что не все ИОУ могут работать в режиме *Rail-To-Rail* («от питания до питания»), и максимальное выходное напряжение для обычных ИОУ (см. п. 5.4.4) обычно мень-

ше напряжения питания. Поэтому необходимо правильно выбирать коэффициент усиления, чтобы при любом значении $U_{вх}$ напряжение $U_{вых}$ не выходило за пределы рабочей зоны и не было искажений выходного сигнала (см. рис. 6.36).

Аналогичные требования предъявляются и к величине входного синфазного сигнала. Необходимо следить за тем, чтобы потенциалы входов обычных ИОУ относительно общей точки обязательно были внутри однополярного питания.

Дело в том, что входная характеристика биполярного транзистора имеет, как известно, существенную нелинейность (см. рис. 6.17). И поэтому при однополярном питании есть ограничения на величину минимального и максимального входного сигнала.

Для пояснения на рис. 6.37 приведена иллюстративная схема входного дифференциально-каскада ИОУ при однополярном питании, из которой видно, что величина $U_{вх}$ должна быть больше хотя бы напряжения $U_{отп}$ входного транзистора (обычно на 1..2 В). Иначе базовый ток транзистора будет равен нулю, и малые приращения входного сигнала усиливаться не будут.

А с другой стороны, величина $U_{вх}$ должна быть меньше (на 1..2 В) напряжения питания $+E_1$, чтобы напряжение на коллекторном переходе $U_{БК}$ оставалось обратным, что необходимо для работы транзистора в усилительном режиме.

В ИОУ с полным размахом выходного напряжения («от питания до питания»), предназначенных для работы в однополярном низковольтном режиме запрещенная зона для входного и выходного сигналов значительно снижена (до десятков мВ) за счет использования полевых транзисторов и специальной схемотехники входных и выходных каскадов [2].

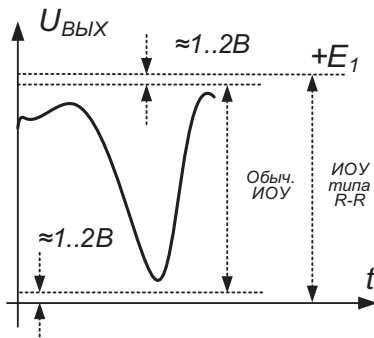


Рис. 6.36

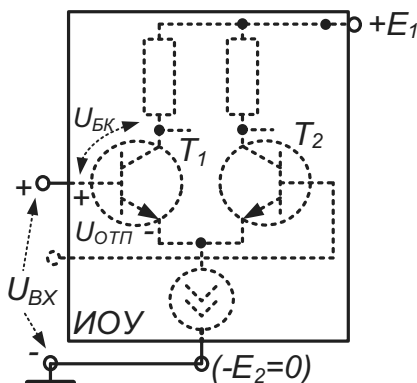


Рис. 6.37

6.6.5. Получение отрицательного напряжения из импульсов

При невозможности использования низковольтных ИОУ с полным размахом выходного напряжения в некоторых случаях для обеспечения работоспособности обычных ИОУ приходится применять простейший импульсный инвертор полярности (рис. 6.38) в качестве источника отрицательного напряжения $-E_2$. Прямоугольные импульсы для него можно получить на одном из выводов микропроцессора МП, при помощи программного генератора.

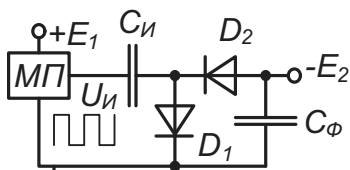


Рис. 6.38

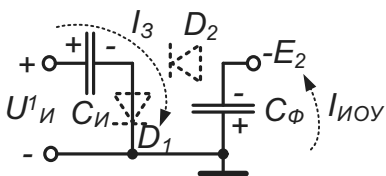


Рис. 6.39

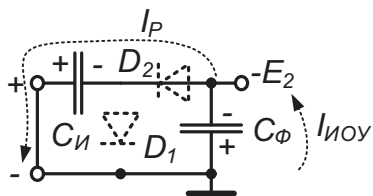


Рис. 6.40

В момент, когда $U_{и} = U_{и}^1$ происходит заряд конденсатора $C_{и}$ до $U_{и}^1$ (полярность указана на рис. 6.39) током I_3 через диод D_1 . В другой полупериод (рис. 6.40), когда $U_{и} = U_{и}^0 \approx 0$, левая положительная обкладка конденсатора $C_{и}$ оказывается подключенной к общей точке, а правая (отрицательная) через диод D_2 — к фильтрующему конденсатору C_{ϕ} , который заряжается током I_p и запоминает отрицательное напряжение и хранит его затем почти неизменным.

Следует отметить, что при определении величины E_2 в установившемся режиме нужно также учитывать падения напряжений на диодах и перепад сигнала $U_{и}$:

$$|E_2| = U_{и}^1 - 2U_{пр} - U_{и}^0 \approx E_1 - 2U_{пр} - U_{и}^0 \quad (6.43)$$

Поэтому реально величина $|E_2|$ оказывается на 2...3 В меньше напряжения E_1 , но для большинства ИОУ и такого небольшого отрицательного напряжения бывает достаточным для нормальной работы.

При расчете конденсаторов пользуются формулой, известной из школьного курса физики, которая определяет емкость, как отношение приращений заряда и напряжения:

$$C = dQ / dU = \Delta Q / \Delta U = \Delta t I / \Delta U. \quad (6.44)$$

В нашем случае для конденсатора C_ϕ в качестве ΔU будем использовать амплитуду пульсаций $U_{\text{пм}}$, которая задана. В качестве тока I – ток, потребляемый ИОУ от источника питания (находим из справочника), а под Δt будем понимать для (худшего случая) длительность периода импульсов $T_{\text{и}}$. Тогда из (6.46) можно получить расчетную формулу

$$C_\phi = T_{\text{и}} I_{\text{иоу}} / U_{\text{пм}} \quad (6.45)$$

Емкость конденсатора $C_{\text{и}}$ обычно выбирают равной или большей величины C_ϕ .

Пусть для примера $I_{\text{иоу}} = 2 \text{ мА}$, $U_{\text{пм}} = 0,1 \text{ В}$, $T_{\text{и}} = 0,1 \text{ мс}$, тогда $C_\phi = C_{\text{и}} = 10^{-4} \text{ 2 } 10^{-3} / 0,1 = 2 \text{ мкФ}$

§6.7. Генератор синусоидальных колебаний

Важную роль в микропроцессорных системах играют тактовые генераторы, обеспечивающие синхронизацию и совместную работу множества цифровых устройств.

6.7.1. Общие сведения

Не вдаваясь в подробную классификацию, отметим, что на практике генераторы различают в основном по форме сигнала, причем чаще всего используются синусоидальные и импульсные. В свою очередь все они делятся на низкочастотные и высокочастотные, автоколебательные и ждущие.

Принцип действия генераторов может быть основан на использовании нелинейных элементов с участком отрицательного сопротивления (применяют на очень высоких частотах), усилителей с положительной обратной связью (см. далее), гистерезисных устройств (см. §17.5) и прямого цифрового синтеза (см. п. 14.2.3).

Для задания частоты могут применяться резистивно-емкостные цепи (см. п. 6.7.3), индуктивные и кварцевые (см. §6.8).

Интересно отметить, что в те времена, когда электроники не было, миллисекундные импульсы получали при помощи маятника, который, качаясь, замыкал контакты. А микросекундные импульсы «генерировались» пулей из винтовки, которая в полете замыкала ряд контактов на своем пути.

6.7.2. Условия возникновения генерации в усилительной схеме

Ранее уже отмечалось (см. п. 5.1.6), что в схеме усилителя с ОС могут при определенных условиях возникнуть колебания. Эти условия обычно называют фазовым и амплитудным.

Фазовое условие требует, чтобы сдвиг фаз при прохождении сигнала по замкнутому усилительному кольцу (рис. 6.41) был кратен 360 градусам:

$$\varphi = \varphi_{\text{УС}} + \varphi_{\text{ОС}} = 0^\circ, 360^\circ, \dots \quad (6.46)$$

Иными словами, чтобы в схеме была положительная ОС, коэффициент усиления для которой был найден ранее (см. п. 5.1.3) и описывается уравнением:

$$K_{\text{ПОС}} = \frac{U_{\text{ВЫХ}}}{U_{\text{ВХ}}} = \frac{K}{1 - K\beta}. \quad (6.47)$$

Из уравнения можно заметить, что при $K\beta \rightarrow 1$, коэффициент $K_{\text{ПОС}}$ стремится к бесконечности. В реальной схеме такой большой коэффициент не достижим, поскольку возникает генерация.

Отсюда следует *амплитудное условие* – коэффициент усиления в генераторном кольце должен быть не меньше единицы, то есть должно выполняться неравенство

$$K\beta \geq 1. \quad (6.48)$$

Нарушение любого из двух условий (6.46) и (6.48) приводит к пропаданию генерации.

6.7.3. Схема генератора Вина

Рассмотрим особенности генератора синусоиды на примере схемы, которая называется мостовым генератором Вина (рис. 6.42) и состоит из неинвертирующего усилителя УС и звена обратной связи ОС.

Сдвиг фаз неинвертирующего усилителя равен нулю.

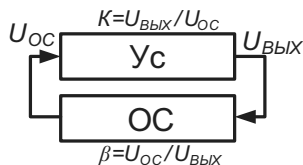


Рис. 6.41

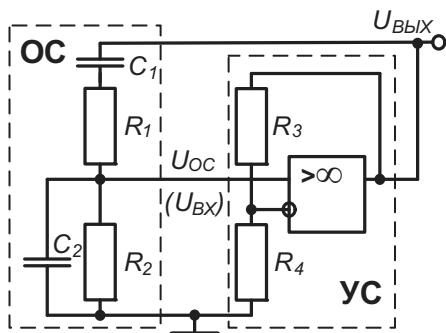


Рис. 6.42

Поэтому согласно фазовому условию генерация может возникнуть на частоте, на которой звено ОС тоже будет вносить нулевой сдвиг.

Цепь ОС имеет два конденсатора, причем C_1 (как дифференцирующий) создает опережающий фазовый сдвиг, а C_2 (как интегрирующий) – отстающий. На какой-то частоте фазовые сдвиги емкостей взаимно компенсируются и результирующий сдвиг оказывается равным нулю. Покажем это.

6.7.4. Коэффициент передачи звена обратной связи

Для анализа частотно-фазовых свойств звена ОС запишем напряжение U_{OC} , зависящее от выходного $U_{ВЫХ}$ в векторной форме:

$$U_{OC} = \frac{U_{ВЫХ} \cdot \frac{R_2 \cdot 1/j\omega C_2}{R_2 + 1/j\omega C_2}}{R_1 + \frac{1}{j\omega C_1} + \frac{R_2 \cdot 1/j\omega C_2}{R_2 + 1/j\omega C_2}} = \frac{U_{ВЫХ} \cdot \frac{R_2}{j\omega C_2}}{\left(R_1 + \frac{1}{j\omega C_1}\right) \left(R_2 + \frac{1}{j\omega C_2}\right) + \frac{R_2}{j\omega C_2}}. \quad (6.49)$$

После некоторых преобразований можно получить напряжение обратной связи в следующем виде:

$$U_{OC} = \frac{U_{ВЫХ}}{1 + \frac{R_1}{R_2} + \frac{C_2}{C_1} + j \left(\omega C_2 R_1 - \frac{1}{\omega C_1 R_2} \right)}. \quad (6.50)$$

И на основе этого выражения найти комплексный коэффициент передачи звена ОС, как отношение напряжения U_{OC} к выходному $U_{ВЫХ}$:

$$\beta = \frac{1}{1 + \frac{R_1}{R_2} + \frac{C_2}{C_1} + j \left(\omega C_2 R_1 - \frac{1}{\omega C_1 R_2} \right)}. \quad (6.51)$$

6.7.5. Нахождение частоты генерации

Из формулы (6.51) видно, что нулевой фазовый сдвиг звена ОС, необходимый для возникновения генерации, будет в том случае, когда выражение в круглых скобках будет равно нулю:

$$\omega C_2 R_1 - 1 / \omega C_1 R_2 = 0. \quad (6.52)$$

Это уравнение позволяет найти частоту, на которой выполняется данное фазовое условие:

$$\omega_0 = 1 / \sqrt{R_1 C_1 R_2 C_2}, \quad (6.53)$$

где $\omega_0 = 2\pi f_0$ – угловая частота, которую иногда называют частотой квазирезонанса.

Дело в том, что резонанса, как явления резкого увеличения или уменьшения амплитуды на определенной частоте, здесь нет. Просто на этой частоте выполняется одно из условий возникновения генерации.

Поэтому выражение (6.53) для коэффициента передачи звена ОС на частоте квазирезонанса можно переписать в упрощенном виде:

$$\beta_0 = 1 / (1 + R_1 / R_2 + C_1 / C_2). \quad (6.54)$$

6.7.6. Нахождение требуемого коэффициента усиления

В генераторе должно выполняться и второе, амплитудное условие (6.48), которое можно переписать для частоты квазирезонанса, подставляя значение β_0 из (6.54),

$$K / (1 + R_1 / R_2 + C_1 / C_2) = 1. \quad (6.55)$$

Принимая далее для простоты расчетов $R_1 = R_2$, $C_1 = C_2$, можно записать

$$K / (1 + 1 + 1) = 1 \quad (6.56)$$

и найти для рассматриваемого случая требуемый коэффициент усиления усилителя $K = 3$. Как следует из описания неинвертирующего УПТ (см. п. 5.5.2), такой коэффициент обеспечивается при $R_4 = 2R_3$.

Если коэффициент будет меньше трех, то генерация не возникнет. Если коэффициент будет больше требуемого, то выходное напряжение генератора будет большим, выйдет в режим насыщения (см. рис. 6.27) и будет искажено.

Поэтому схемы генераторов, как правило, имеют автоматическую регулировку усиления (АРУ) на основе нелинейного элемента, рассмотрение которой выходит за рамки пособия.

Отметим также, что цепь ОС может иметь RC элементы с другой структурой. Высокочастотные генераторы выполняют, как правило, на основе LC-цепей, но в вычислительной технике чаще используют кварцевые генераторы (см. далее).

§6.8. Кварцевые генераторы

Кварцевыми называют генераторы, в которых частоту задает кварцевый резонатор или в просторечии кварц. Его замечательной особенностью является очень высокая стабильность час-

тоты. Считается, что это один из самых точных элементов, используемых в электронике.

6.8.1. Свойства кварцевого резонатора

Кварцевый резонатор представляет собой природный или искусственный кварцевый кристалл, имеющий два электрода и обозначаемый на принципиальной схеме в виде УГО, приведенного на рис. 6.43.



Рис. 6.43

В пластинке кварца может возникать пьезоэффект – взаимное преобразование электрического поля в механическую деформацию, которое подобно превращениям электрической и магнитной энергии в LC -контуре.

При приложении переменного напряжения между электродами пьезокристалл начинает вибрировать. Причем амплитуды вибрации и тока через кристалл становятся максимальными (рис. 6.44), когда внешняя частота совпадает с механической резонансной частотой самой пластины. То есть кварц, как и колебательный LC -контур, обладает ярко выраженными резонансными свойствами.

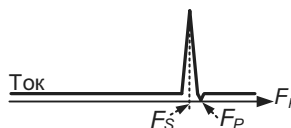


Рис. 6.44

Характеристика резонатора может иметь и другие менее заметные экстремумы на более высоких частотах, которые называют гармониками кварца.

Обычно говорят, что кварц хорошо пропускает сигналы основной резонансной частоты и почти не пропускает все остальные. Но это правильно отчасти. Дело в том, что кварц имеет две несколько различающиеся (примерно на 1%) резонансные частоты, параллельного резонанса F_P и последовательного F_S , причем $F_P > F_S$. Схемы генераторов могут использовать оба типа резонанса (см. п.п. 6.8.2 и 6.8.3).

Главной особенностью кварца является то, что его частотные свойства определяются весьма стабильными геометрическими размерами пластины, которые почти не изменяются при изменении температуры и со временем.

При анализе генераторов и в моделирующих программах обычно используют эквивалентную схему, приведенную на рис. 6.45, где элементы L_1 и C_1 отражают индуктивность и емкость подвижной массы пластины, сопротивление R_1 учи-

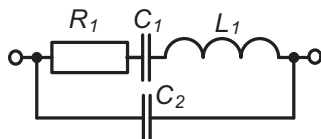


Рис. 6.45

тывает потери энергии при колебаниях, емкость C_0 – наличие паразитной электростатической связи между выводами.

Следует также отметить, что элементы L_1 и C_1 определяют последовательный резонанс, а свойства параллельного резонанса определяют все элементы эквивалентной схемы. Поэтому паразитные емкости монтажа могут несколько сместить резонансную частоту F_p . Иногда, наоборот, к кварцу подключают дополнительные конденсаторы для небольшого изменения частоты.

Вторым после частоты резонанса важным параметром кварца является добротность ($\approx 20..200$ тысяч), которая показывает во сколько раз энергия, запасенная в пластине, больше энергии потерь на колебания. Чем больше добротность, тем выше и уже резонансный экстремум.

Примечание: *От добротности зависит время запуска (разгона) генератора. Это приходится учитывать в микропроцессорных системах при выборе длительности импульса сброса (см. п. 7.1.5).*

Различают природные кристаллы кварца и искусственные (пьезо-керамические). Природные резонаторы имеют большую добротность и стабильность, однако пьезокерамические – дешевле.

Чем выше частота кварца, тем тоньше и хрупче его пластина. Чтобы увеличить механическую прочность и устойчивость к ударам были разработаны для высокочастотных применений (более 30 МГц) специализированные резонаторы, работающие на 3 гармонике. Они маркируются в «МГц», а не в «кГц», в отличие от кварцев, работающих на основной гармонике.

В вычислительных устройствах могут использоваться две основные схемы кварцевых генераторов, каждая из которых имеет свои достоинства и недостатки.

6.8.2. Кварцевый генератор на неинвертирующем усилителе

Для работы на высоких частотах необходим соответствующий усилитель. В простейшем варианте можно использовать логические элементы, имеющие рабочие частоты до десятков и даже сотен МГц, которые значительно дешевле высокочастотных ИОУ и имеют несколько инверторов в одном корпусе микросхемы.

Схема генератора, приведенная на рис. 6.46, реализована на двух инверторах, выполняющих функции неинвертирующего усилителя,

поскольку инверторы сдвигают фазу дважды по 180 градусов, то есть на 360 градусов.

Конденсатор C_P пропускает полностью переменную составляющую сигнала и является разделительным (см. п. 6.5.1).

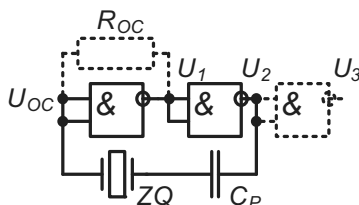


Рис. 6.46

В данном генераторе используется режим последовательного резонанса кварца, который характеризуется минимальным сопротивлением и нулевым фазовым сдвигом. Поэтому суммарные сдвиги усилителя (360°) и резонатора (0°) обеспечивают выполнение *фазового* условия (6.46) на частоте F_S .

Однако с выполнением *амплитудного* условия (6.48) в данном случае имеются определенные трудности. Дело в том, что логический элемент предназначен для работы с цифровыми уровнями сигналов, то есть рабочими для него являются участки «А» и «Б» передаточной характеристики (рис. 6.47). Коэффициенты усиления на этих участках (отношение приращений выходного напряжения к входному) из-за очень малого $\Delta U_{ВЫХ}$ оказываются ничтожными.

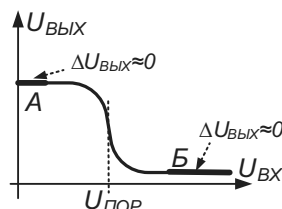


Рис. 6.47

Поэтому для выполнения амплитудного условия необходимо вывести инвертор на крутой участок характеристики вблизи напряжения $U_{ПОР}$. Для этого включают резистор R_{OC} , показанный пунктиром, который подает на вход часть выходного напряжения. В этом случае работающий на крутом участке инвертор может обеспечить коэффициент $K \approx 50..100$ в зависимости от типа микросхемы.

Чтобы резистор R_{OC} пропускал большой входной ток $I_{ВХ}^0$ инвертора (см. п. 2.4.3), его сопротивление выбирают примерно равным 400 Ом для ТТЛ элементов и в тысячу раз больше для КМОПТЛ.

Иногда подобный резистор используют и во втором инверторе, чтобы увеличить надежность запуска генератора, но при этом между инверторами нужен второй разделительный конденсатор. И схема оказывается перегруженной внешними элементами.

Достоинством рассмотренной схемы генератора на основе неинвертирующего усилителя является работа кварца на частоте последовательного резонанса F_S , при которой на частоту генерации не влияют паразитные и подстроечные емкости. Кроме того, использование ло-

гических элементов позволяет получать прямоугольный сигнал ТТЛ уровня, для чего обычно используют третий инвертор, показанный на рис. 6.47 пунктиром.

Недостатки – при излишне большом коэффициенте усиления генератор может возбуждаться на механических гармониках кварца (на частоте в 3 или 5 раз большей расчетной). При этом система может оказаться неработоспособной.

Кроме того на стабильность частоты генерации несколько влияют напряжение питания и саморазогрев кварца. Дело в том, что в схеме, работающей на частоте последовательного резонанса, протекает достаточно большой ток (см. рис. 6.44) и если его не ограничивать, то кварц может разогреваться и стабильность его частоты ухудшается. В этом смысле генератор на частоте параллельного резонанса (см. далее), характеризуемого минимумом тока и соответственно меньшей выделяемой мощностью, является более стабильным.

6.8.3. Кварцевый генератор на инвертирующем усилителе

В данной схеме (рис. 6.48) для того, чтобы выполнялось фазовое условие (6.46), необходимо дополнительно к 180-градусному сдвигу инвертирующего усилителя добавить с помощью цепи обратной связи сдвиг еще на 180 градусов.

Для этого совместно с кварцем, который в режиме параллельного резонанса эквивалентен LC-контуре (см. рис. 6.45), включают конденсаторы C_1 и C_2 . В результате образуется генераторное кольцо, в котором могут возникнуть колебания на частоте F_p .

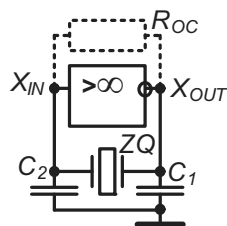


Рис. 6.48

При выборе величин емкостей C_1 и C_2 нужно иметь в виду, что искусственные (керамические) кварцы требуют несколько больших значений (30...50 пФ). Иногда с этой особенностью сталкиваются на практике при отладке, когда система начинает работать, если коснуться вывода генератора шнуром осциллографа, который вносит свою емкость в цепь ОС, и генератор возбуждается.

Отметим, что в данной схеме более заметно влияние паразитных емкостей и инерционных свойств усилителя. Однако кварц, работающий с минимальными токами на частоте F_p (см. рис. 6.45), не разогревается и в целом обеспечивается булышая, чем в генераторе с

неинвертирующим усилителем, общая стабильность частоты и независимость ее от напряжения питания.

Поэтому генератор с инвертирующим усилителем является основой для встроенных генераторов, причем заботы о выполнении условий генерации берут на себя производители микросхем. А пользователю нужно только точно следовать рекомендациям, изложенным в технической документации.

Полезно знать, что в некоторых процессорах имеются конфигурационные биты для выбора высокочастотного или низкочастотного кварца, а также подключения внутренней (и нестабильной) RC-цепочки вместо внешнего кварца.

Инвертирующие усилители внутри процессоров (рис. 6.49), обычно имеют выходы входа X_{IN} и выхода X_{OUT} для подключения внешнего кварца и конденсаторов.

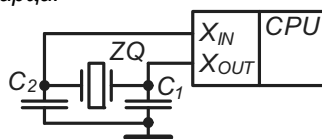


Рис. 6.49

Отметим, что выходной сигнал подобных схем является практически синусоидальным и имеет значительно больший уровень, чем входной. Поэтому с вывода X_{OUT} при необходимости можно снимать сигнал для своих нужд. А на вход X_{IN} можно в отсутствие кварца подавать тактирующий сигнал от другого генератора, например, от кварцевого модуля, содержащего и резонатор, и усилитель.

7 ВНУТРЕННИЕ РЕСУРСЫ ОВМ СЕМЕЙСТВА x51

В этой главе приводятся основные сведения для первоначального знакомства с функциональными возможностями ОВМ, которые, не подменяя справочных, дают опорный материал для использования в дальнейшем при изучении процессоров других семейств. Чтобы более наглядно иллюстрировать процессы, происходящие внутри ОВМ, принципы действия ее внутренних узлов будут поясняться при помощи средств языка ассемблера. В дальнейшем будем придерживаться в основном рекомендаций, изложенных в ОСТ 110342.2-87 “Язык ассемблера АСМ51”.

Имена переменных и процедур, вводимые пользователем в примерах программных фрагментов, будут начинаться с префикса “m”, чтобы отличать их от стандартных имен спецрегистров, операторов и директив, которые для большей заметности будут набраны жирным шрифтом.

§7.1. Общее описание ОВМ x51

7.1.1. Внутренняя структура ОВМ

Под ОВМ обычно понимается функционально законченная БИС, имеющая в своем составе (рис. 7.1) все необходимые узлы для самостоятельной работы по программе с целью обслуживания внешних устройств.

Как правило, в ОВМ выделяют две части, основную (процессорное ядро) и дополнительную (набор вспомогательных устройств).

7.1.2. Процессорное ядро

Основой ОВМ x51 являются:

- 8-разрядный процессор, обеспечивающий вычисления и управление по программе;

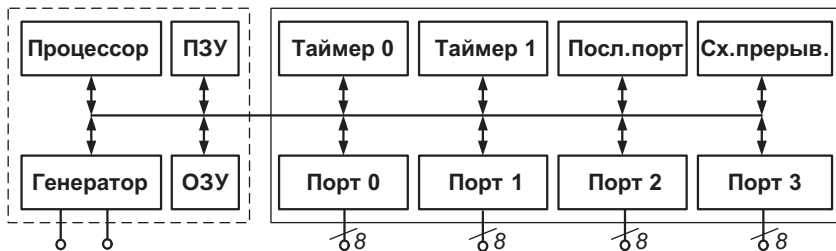


Рис. 7.1

- память команд, в виде постоянного запоминающего устройства (ПЗУ), где хранится программа;
- память данных, которую часто для простоты называют оперативным запоминающим устройством (ОЗУ);
- тактовый генератор (с внешним кварцем), обеспечивающий внутреннюю синхронизацию.

Раздельная память команд и данных (гарвардская архитектура) применяется в большинстве ОВМ, поскольку позволяет при модернизации процессора увеличивать размер памяти без опасности наложения адресных пространств ПЗУ и ОЗУ друг на друга.

Процессор взаимодействует со своим обрамлением через внутреннюю системную магистраль, но обычно этот обмен скрыт от пользователя.

7.1.3. Внутренняя периферия

Важной частью ОВМ является набор так называемых внутренних периферийных устройств, предназначенных для взаимодействия с внешними объектами и аппаратной реализации некоторых функций – счета и измерения времени, цифрового и аналогового ввода-вывода, организации последовательной связи и т. д. Следует отметить, что набор этих устройств для разных семейств различен, но в любом случае имеются регистры, называемые портами, для внешнего ввода-вывода цифровых сигналов.

В стандартном варианте ОВМ x51 имеет 4 порта, два таймера-счетчика, последовательный порт и схему прерываний, особенности которых будут рассмотрены далее.

В каждом из подобных устройств имеется несколько регистров, организующих их работу и называемых специальными регистрами (*SFR – Special Function Registers*), в отличие регистров общего назначения (РОН), необходимых для вычислений. Все спецрегист-

ры имеют стандартные символические имена, которые понимаются инструментальными программами и используются программистами. Например, регистры четырех портов обозначаются как **P0**, **P1**, **P2**, **P3**, регистр управления для последовательного порта имеет имя **SCON** (**S**erial **C**ONTrol). Другие стандартные имена спецрегистров будут рассмотрены по ходу пояснения.

Отметим только, что в состав внутренней периферии более сложных ОВМ [31] дополнительно входят энергонезависимая память данных, АЦП, ЦАП, цифровые часы, интерфейс (например, USB) и другие устройства, необходимые микропроцессорной системе для обслуживания внешних объектов. Эти дополнительные устройства имеют свой набор и имена спецрегистров.

7.1.4. Условное графическое обозначение

выполнено, как обычно, в виде прямоугольника (рис. 7.2), в центральном поле которого пишется знак процессора – **CPU**. Как известно, обозначения внешних *входов* должны размещаться слева, а *выходов* – справа. Однако это правило неприменимо для портов **P0..P3**, поскольку биты портов могут служить как для ввода, так и для вывода сигналов. Поэтому внешние выводы портов рисуют и слева, и справа.

Каждый из 4 портов имеет 8 линий, то есть всего в распоряжении разработчика имеется 32 внешних вывода, которые можно использо-

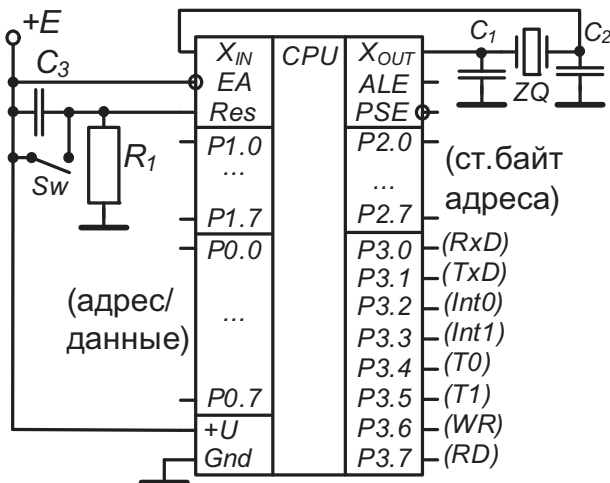


Рис. 7.2

вать независимо друг от друга, побайтно или побитно, для ввода или вывода сигналов.

Выводы *отдельных битов* портов нумеруются числами от 0 до 7 через точку после имени порта, например, **P1.5**. Особенности реализации выходных цепей портов приведены в §7.2.

Замечательной особенностью семейства x51 является наличие команд управления отдельными битами портов или переменных (см. п. 7.10.6).

Общее число внешних выводов, включая общий вывод и питание, в стандартном варианте равно 40. Но существуют множество разновидностей ОВМ семейства x51, отличающихся размером внутренней памяти, наличием дополнительных устройств (АЦП, интерфейсов). Такие сложные кристаллы могут иметь значительно большее число внешних выводов и соответствующую цену. Для простых применений выпускают дешевые микросхемы с усеченным набором внутренней периферии и малым числом внешних выводов (см. п. 7.2.3).

Важно помнить, что многие биты портов ОВМ имеют иные (альтернативные) назначения, которые указаны на рис. 7.2 в скобках и могут быть при необходимости запрограммированы (см. п.п. 7.1.6 и 7.1.7).

Кроме 32 линий портов для общения с внешними устройствами, ОВМ имеет несколько служебных выводов, назначение которых рассмотрено далее.

7.1.5. Служебные выводы общего назначения

Обеспечивают работоспособность ОВМ и служат для подключения внешних элементов для генератора и цепи сброса.

X_{IN} , X_{OUT} – вход и выход внутреннего генератора для подключения внешнего кварцевого резонатора, задающего тактовую частоту работы ОВМ.

Частоту кварца выбирает пользователь, исходя из условий технического задания. Например, на предельных рабочих частотах (от 12 до 50 МГц для разных кристаллов) достигается максимальная производительность, но существенно увеличивается энергопотребление. Переносные устройства, питающиеся от батареек, для снижения потребляемой мощности используют пониженные частоты от 1 МГц и менее. Иногда, если позволяет процессор, применяют даже кварц от цифровых часов 32 кГц. Очевидно, что при этом производительность системы снижается.

Следует отметить, что кварц обязательно обрамляется внешними конденсаторами C_1 и C_2 , обеспечивающими фазовое условие возникновения генерации (см. п. 6.8.3). Величина этих конденсаторов указывается производителем в технической документации и находится в пределах от 15 до 50 пФ. Причем меньшая цифра относится к природным кварцам, а большая - к искусственным (керамическим).

В некоторых ОВМ можно программно включить генератор с внутренней RC цепочкой, задающей фиксированную частоту, например 1 МГц. Но стабильность тактовой частоты в этом случае невысока.

RES (от **RESet** – сбросить) – вход, который служит для установки ОВМ в начальное состояние. При этом большинство битов спецрегистров сбрасываются в ноль, а биты портов устанавливаются в единичное состояние (ориентируются на ввод, см. п. 7.2.2). В программный счетчик записывается нулевой код и выполнение программы после окончания импульса сброса начинается поэтому с нулевой ячейки (см. п. 7.3.1).

Единичный сигнал сброса должен иметь длительность, большую времени разгона генератора. Время разгона (старта) генератора зависит от частоты и добротности кварца (см. п. 6.8.1). Для природных кварцев это время равно примерно 10000 тактовых периодов. Например, на частоте 12 МГц время разгона – 1,2 мс, на 1 МГц – 10 мс. Для страховки длительность импульса сброса делают большим 20 мс.

В простейшем случае импульс сброса получают от кнопки Sw или при помощи дифференцирующей RC цепочки (см. п. 1.2.3), как показано на рис. 7.2. При подаче напряжения питания на схему разряженный ранее конденсатор C_3 представляет в первом приближении короткую. Поэтому на выводе сброса возникает единичный сигнал, который по мере заряда конденсатора спадает до нуля. Учитывая, что большой точности для импульса сброса не требуется, берут постоянную времени с запасом, например, $R_1 C_3 \approx 20$ мс. Поэтому, если $R_1 = 10$ кОм, то $C_3 = 20 \cdot 10^{-3} / 10 \cdot 10^3 = 2$ мкФ.

Отметим, что в некоторых микросхемах есть внутренний твердотельный резистор между входом и общей точкой, поэтому в этом случае внешний резистор R_1 в цепи сброса можно исключить.

Для дальнейшего уменьшения внешних элементов внутри некоторых ОВМ иногда размещают и сам формирователь импульса сброса, который в ряде случаев делают в виде так называемого сторожевого таймера (Watchdog Timer) [31]. Это специализированное устройство сброса предохраняет программу от «зависаний» в тех случаях, когда некому перезапустить программу.

При нормальной работе сторожевой таймер периодически обнуляется программой, чтобы сигнал сброса был пассивным. При нештатной работе программы таких обнулений нет, и сторожевой таймер, спустя некоторое время, вырабатывает сигнал сброса, возвращая программу в исходное состояние.

7.1.6. Выводы для обслуживания внешней памяти

В то время, когда создавалось семейство x51, технология изготовления внутренней памяти была недостаточно развита. Поэтому базовая архитектура предусматривала подключение внешней памяти, для чего пришлось пожертвовать двумя портами **P0** и **P2**, которые были задействованы для обслуживания внешних ПЗУ и ОЗУ. Например, порт **P0** в этом режиме используется для мультиплексной передачи младшего байта адреса – данных, а **P2** – для передачи старшего байта адреса. Подробности здесь не рассматриваются, их можно найти в §14.6. Ниже указаны минимальные сведения, касающиеся назначения внешних выводов для обслуживания внешней памяти.

EA (External Address – внешний адрес) – *инверсный вход*, сигнал на этом выводе определяет, какое ПЗУ будет использоваться в работе, внешнее или внутреннее (см. п. 14.6.4). В большинстве случаев используется внутренняя память команд и этот инверсный сигнал пассивен и равен 1.

ALE (Address Latch Enable – «защелка» адреса разрешена) – *выход*, на котором формируется тактовый импульс для синхронизации внешних устройств, служит чаще всего для фиксации во внешнем регистре младшего байта адреса, выводимого через порт **P0** (см. п. 14.6.1).

PSE (Program Storage Enable) – *инверсный выход*, на котором образуется синхросигнал чтения из внешней памяти команд. Он возникает тогда, когда происходит выборка из внешнего ПЗУ, определяя момент приема кода очередной команды через порт **P0** (см. п. 14.6.2).

Отметим, что выходы синхросигналов чтения-записи (RD и WR) для внешнего ОЗУ используют при необходимости биты порта P3 (см. п. 7.1.7).

Порт P1 в базовой конфигурации альтернативных функций не имеет.

У кристаллов, не предназначенных для подключения внешней памяти, сигналы EA, ALE и PSE отсутствуют.

Кроме того, отметим, что при занесении двоичных кодов во внутреннее ПЗУ (программировании кристалла, см. п. 7.3.3) прак-

тически все выводы также меняют свое назначение. Но эти вопросы специфичны и здесь не рассматриваются.

7.1.7. Альтернативные функции порта P3

Чтобы не увеличивать числа внешних выводов ОВМ, некоторые из битов порта **P3** могут использоваться различной внутренней аппаратурой, то есть являются многофункциональными.

Перечень альтернативных названий битов порта **P3**, которые внутренняя периферия может после программной активации использовать для общения с внешними устройствами, дан на рис. 7.2 в скобках и приведен ниже.

RxD (Receive Data) – принимаемые данные) – *вход* приемника последовательного порта (см. п. 12.6.3).

TxD (Transmitte Data) – передаваемые данные) – *выход* передатчика последовательного порта (см. п. 12.6.1).

INT0, INT1 (INTerrupt) – прерывание) – *входы* внешних запросов на прерывание работы процессора по программе для экстренного обслуживания внешних устройств. Механизм прерывания рассмотрен в п. 8.4.1.

T0, T1 – внешние *входы* для управления таймерами-счетчиками (см. §8.6).

RD (Read) и **WR (WRite)** – *выходы* инверсных синхросигналов чтения и записи, которые формируются в момент, когда процессор обращается к внешнему ОЗУ, производя чтение или запись (см. п. 14.6.3).

В некоторых ОВМ имеется возможность программно назначать внешний вывод порта для обслуживания внутренней периферии.

§7.2. Особенности выходных цепей портов

При подключении внешних схем к ОВМ нужно помнить, что выходные каскады портов в x51 имеют схемотехнику двух типов: классическую двунаправленную (с двумя ключами и третьим состоянием) и квазидвунаправленную (с одним ключом и внутренним резистором нагрузки).

7.2.1. Выходные цепи порта P0

Порт **P0** имеет двунаправленную структуру с третьим состоянием (см. п. 2.3.7), свойственную практически всем микропроцессорным

магистралям для обмена информацией с множеством абонентов. В первом приближении выходной каскад, обслуживающий отдельный вывод порта **P0**, можно представить в виде двух противофазно работающих ключей (рис. 7.3), которые управляются командами процессора.

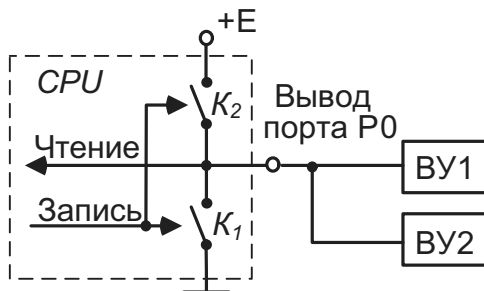


Рис. 7.3

При выводе информации один из ключей проводит ток, другой – не проводит, обеспечивая на внешнем выводе сигналы «0» или «1», например, для внешнего устройства ВУ1. При вводе информации оба ключа находятся в непроводящем (третьем) состоянии, а данные на выводе под управлением не показанного на схеме синхросигнала могут устанавливаться внешним устройством ВУ2.

Такая структура порта позволяет подключать к ОВМ внешнюю память, и в этом режиме порт **P0** выполняет функции мультиплексированной шины адреса/данных. Отметим также, что большую роль в организации обмена через порт **P0** играют синхросигналы чтения-записи *PSE*, *RD* и *WR* (см. §14.6), разрешающие ввод или вывод *байтовой* информации.

*Очень важно отметить, что в том случае, когда внешняя память отсутствует, ключ K_2 порта P0 всегда находится в непроводящем состоянии, работает только ключ K_1 , то есть двухключевой каскад превращается в каскад с открытым стоком. Эту особенность нужно обязательно учитывать при выводе информации (см. п. 7.2.4 и 7.2.5) и включать **внешний** «подтягивающий» резистор (см. далее).*

7.2.2. Выходные цепи портов **P1**, **P2**, **P3**

Эти порты имеют так называемую квазидвунаправленную структуру (рис. 7.4), основанную на использовании однотранзисторной выходной цепи с внутренним транзисторным ключом K_1 и резистором R_B , который часто называют «подтягивающим» (английский синоним *Pull-up*).

Подобная конфигурация выходной цепи позволяет организовать ввод или вывод отдельно по каждой из линий без использования разрешающих синхросигналов направления.

При работе линии порта на вывод информация (в виде «0» или «1») записывается процессором во внутренний триггер порта и при помощи буферного ключа K_1 устанавливается на внешнем выводе. Таким образом, активная роль принадлежит ключу K_1 , и внешние сигналы на вывод, вообще говоря, подавать не рекомендуется (хотя и не возбраняется).

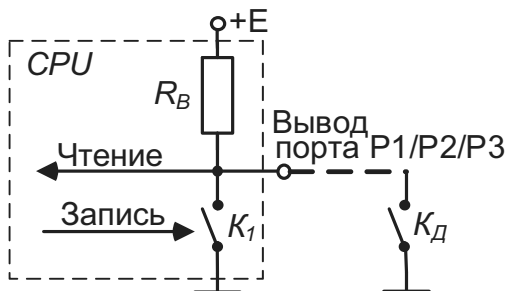


Рис. 7.4

Информацию с этого же вывода можно также прочитать при помощи процессора. Однако, если ранее триггер был установлен в «0», то ключ K_1 будет проводить. Поэтому процессор в этом случае будет принимать нулевой сигнал независимо от состояния внешнего датчика (ключа K_D). Поэтому, чтобы настроить (запрограммировать) определенный бит порта на ввод, нужно предварительно установить его в «1». Тогда сигнал от ключевого или логического датчика будет правильно восприниматься процессором.

Примечания:

1. Внутренний резистор $R_B \approx 40$ кОм в моменты записывания ключа K_1 переключается на короткое время не показанным на схеме ключом, чтобы форсировать установку единичного сигнала путем ускоренного заряда паразитной емкости нагрузки.
2. Для уменьшения потребляемого тока внутренний резистор R_B отсоединяется от питания, если напряжение на выводе становится меньше примерно 1,5 В. (см. п. 7.2.5).
3. В современных ОВМ тип выходного каскада можно программировать, выбирая двухключевой выход, квазидвухнаправленный вход/выход с одним ключом (с внутренним сопротивлением или без) или только вход [31].

7.2.3. Особенности маловыводного варианта x51

В простых микропроцессорных системах не все ресурсы и внешние выводы («ножки») стандартного варианта ОВМ оказываются задействованы. В таких случаях можно использовать так называемые «маломощные» кристаллы с несколькими ограниченными функциональными возможностями. Особенности их рассмотрим на примере

ОВМ *AT89C2051* фирмы *Atmel*, УГО которой приведено на рис. 7.5.

Ядро и состав такой ОВМ почти полностью соответствуют стандартной конфигурации. Отличие заключается только в том, что исключены служебные выводы и два порта (**P0**, **P2**), за счет чего число внешних выводов уменьшено до 20. Очевидно, что при этом теряется возможность подключения внешней памяти и перестают работать некоторые команды по ее обслуживанию.

Но вся внутренняя периферия семейства *x51*, как и прежде, остается доступной через альтернативные выводы (показаны в скобках) порта **P3**. И, более того, дополнительно в состав данного кристалла введен внутренний *аналоговый компаратор* (см. §17.1), который весьма полезен на практике для многих применений, например, для реализации простейшего АЦП (см. п. 18.4.2).

Для внешнего подключения компаратора (рис. 7.6) задействованы квазидвунаправленные выводы **P1.0** (прямой вход *In+*) и **P1.1** (инверсный вход *In-*), ключи которых в данной микросхеме лишены внутренних подтягивающих резисторов. Если компаратор используется, то эти разряды должны иметь единичное значение (внутренние ключи не проводят), иначе компаратор не будет правильно работать.

Выход внутреннего компаратора скрыт от пользователя, поскольку подключен к не имеющему внешнего вывода биту **P3.6**. Неудобств при этом не возникает, потому что указанный бит обычно читают и анализируют программно.

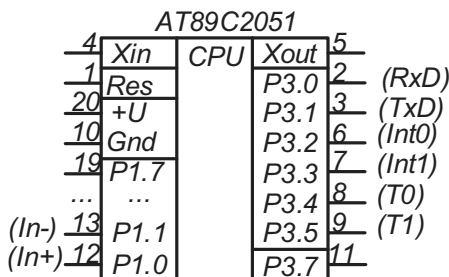


Рис. 7.5

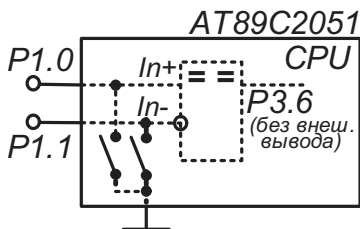


Рис. 7.6

7.2.4. Подключение нагрузки к выводу порта

Нагрузочная способность внешних выводов ОВМ зависит от модификации. В стандартном варианте считается, что к выводу порта

можно подсоединить один ТТЛ вход, то есть $I_{\text{МАКС}}^0 \approx I_{\text{ВХ}}^0 \approx -1,6 \text{ мА}$ и $I_{\text{МАКС}}^1 \approx I_{\text{ВХ}}^1 \approx 0,1 \text{ мА}$.

Напомним, что в соответствии с несимметричной входной характеристикой ТТЛ элемента ток $I_{\text{МАКС}}^0$ втекает в выходной зажим, $I_{\text{МАКС}}^1$ вытекает из него.

Более современные кристаллы имеют увеличенную нагрузочную способность. Например, маловыводная микросхема АТ89С2051 допускает выходной ток до 20 мА.

Некоторые микросхемы могут иметь специализированные выходы с еще большим током для прямого управления, например, светодиодными индикаторами. Но общий ток всех портов ОВМ обычно ограничивается на уровне 60...80 мА.

При подключении нескольких входов внешних микросхем к одному выходу ОВМ вычисляют обычно суммы входных токов $I_{\text{ВХ}}^0$ и $I_{\text{ВХ}}^1$, которые должны быть меньше соответствующих максимальных выходных токов ОВМ – $I_{\text{МАКС}}^0$ и $I_{\text{МАКС}}^1$.

Нередко к внешнему выводу ОВМ подключают светодиод для индикации. В этом случае нужно обязательно включать последовательно внешний резистор $R_{\text{ОГР}}$ (см. п. 3.1.2) для ограничения выходного тока на уровне, меньшем максимального для выходной цепи ($I_{\text{МАКС}}^0$ или $I_{\text{МАКС}}^1$).

Известны два способа подключения светодиода. В первом из них (рис. 7.7 и 7.9) свечение диода происходит при единичном выходном сигнале, а во втором (рис. 7.8 и 7.10) – при нулевом. Однако схема рис. 7.7 для бита порта $P0$ оказывается **нерабочей**, поскольку ключ K_2 функционирует только при обслуживании внешней памяти (см. п. 7.2.1). В этом случае следует использовать схему рис. 7.9 с внешним резистором $R_{\text{ОГР}}$.

Величину резистора для схем рис. 7.8 и 7.10 выбирают исходя из заданного тока светодиода (см. п. 3.1.2), но ограничивают снизу, используя неравенство



Рис. 7.7

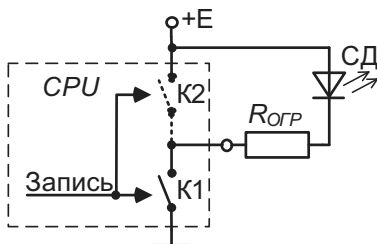


Рис. 7.8

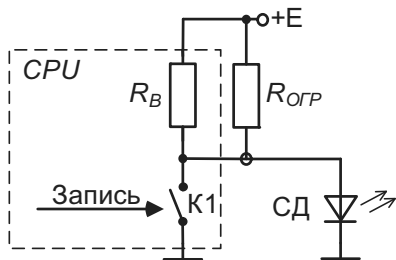


Рис. 7.9

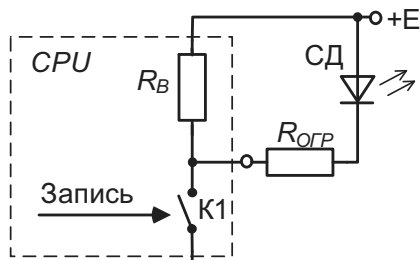


Рис. 7.10

$$R_{OГР} \geq U_{R_{OГР}} / I_{\max} = (E - U_{Вых}^0 - U_{СДпр}) / I_{\max}^0, \quad (7.1)$$

а для схемы рис. 7.9:

$$R_{OГР} \geq U_{R_{OГР}} / I_{\max} = (E - U_{Вых}^0) / I_{\max}^0. \quad (7.2)$$

Кроме светодиодов к выводам ОВМ бывает нужно подсоединять и другую более низкоомную нагрузку, для которой нагрузочной способности ОВМ может не хватить. В таких случаях применяют ключевые усилители на основе внешних микросхем или транзисторов (см. главу 3).

7.2.5. Внешний ключ на *n-p-n* транзисторе

Как и в случае со светодиодом, при подключении внешнего транзистора нужно учитывать структуру выходного каскада порта ОВМ. При использовании порта *P0* не следует надеяться на двухключевой выходной каскад, который образуется только при обслуживании внешней памяти. Таким образом, схема на рис. 7.11 оказывается *нерабочей*, так как ключ K_2 не проводит, и нет единичного сигнала, необходимого для насыщения транзистора. Надо использовать внешний резистор $R_{ББ}$, как на рис. 7.12.

В квазидунаправленной схеме (рис. 7.12) на первый взгляд подобной проблемы нет, благодаря наличию внутреннего резистора, ко-

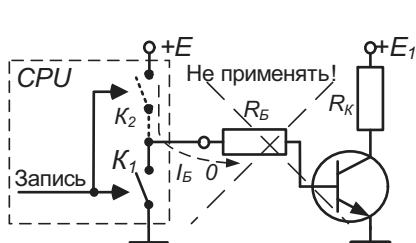


Рис. 7.11

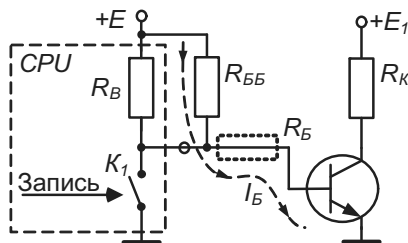


Рис. 7.12

торый можно использовать для задания насыщающего базового тока транзистора (внешний резистор R_B практически не нужен).

Но данная схема все равно оказывается ограниченно работоспособной. Дело не только в том, что внутренний резистор обеспечивает малый базовый ток ($\approx 0,1 \text{ mA}$), но и в том, что единичный уровень напряжения на выводе порта становится малым и равным напряжению на прямосмещенном переходе $U_{БЭ}$. А при таком напряжении внутренний резистор отсоединяется от питания (см. *примечание 2* к п. 7.2.2), базовый ток внешнего транзистора исчезает и ключ становится неуправляемым.

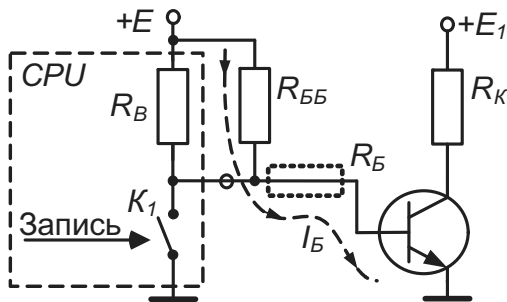


Рис. 7.12

Поэтому рекомендуется подключить параллельно внутреннему резистору R_B внешний $R_{ББ} \approx 4..10 \text{ k}\Omega$ (рис. 7.12), выбираемый из условия насыщения транзистора, но не менее

$$R_{ББ} \geq (E - U_{ВЫХ}^0) / I_{МАКС}^0 \quad (7.3)$$

Напряжение питания ключа E_1 может отличаться от напряжения питания OBM и иметь значение до сотен В.

Другим способом восстановления управляемости внешнего ключа является использование составного транзистора T_C (рис. 7.13), напряжение на базе которого равно сумме напряжений на эмиттерных переходах отдельных транзисторов. Эта сумма превышает внутренний порог, и твердотельный подтягивающий резистор R_B в такой схеме не отсоединяется от питания. Кроме того, составной транзистор имеет большой коэффициент усиления B_C , равный произведению коэффициентов составляющих транзисторов, что позволяет коммутировать ток на-

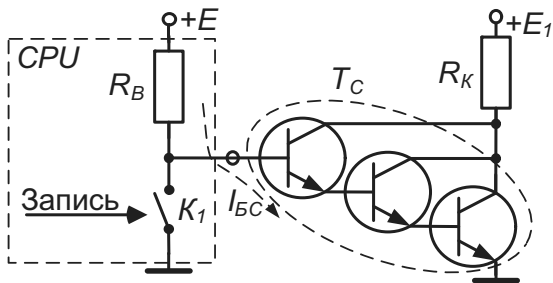


Рис. 7.13

грузки, который в B_c раз больше малого тока внутреннего резистора ($\approx 0,1 \text{ мА}$).

Выходные транзисторы составного работают на границе насыщения и имеют поэтому увеличенное напряжение в проводящем состоянии (см. п. 3.3.1). Для устранения этого недостатка обычно используют двухкаскадный ключ (см. п. 3.3.3). Для увеличения управляющего тока внешнего ключа можно изменить тип транзистора (см. далее), чтобы использовать для его насыщения нулевой сигнал с большим выходным током $I_{\text{МАКС}}^0$.

7.2.6. Внешний ключ на p - n - p транзисторе

Внешний транзистор в подобной схеме (рис. 7.14) проводит не при единице на выводе порта, а при нуле (см. п. 3.2.4). Поэтому базовый ток, задаваемый резистором R_B , может достигать значения $I_{\text{МАКС}}^0$, которое в квазидвухнаправленной выходной цепи ОВМ на порядок больше величины $I_{\text{МАКС}}^1$. Поэтому ток в нагрузке p - n - p транзистора будет больше, чем n - p - n , при одинаковом усилении транзисторов.

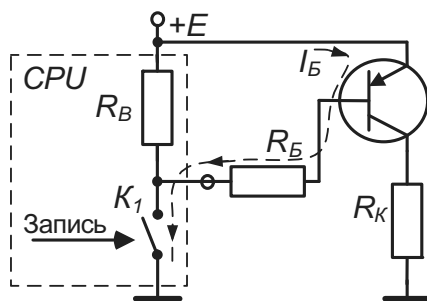


Рис. 7.14

Некоторым недостатком этой схемы является то, что напряжение на нагрузке не превышает напряжение питания ОВМ.

Другие схемы ключевых усилителей мощности, в том числе и с гальваническим разделением входа и выхода, рассмотрены в главах 3 и 4.

§7.3. Работа ОВМ и структура памяти

7.3.1. Общие сведения

Как известно, последовательность выполнения команд в программе определяет программный счетчик процессора **PC** (**Program Counter**), он хранит адрес ячейки памяти команд (ПЗУ), где записан код очередной команды. Последовательное выполнение программы может быть изменено при выполнении команд перехода или при вызове подпрограмм. При этом в **PC** заносится новый адрес (см. §7.11).

Работа ВМ состоит, по сути, в бесконечном циклическом выполнении одних и тех же действий:

- извлечение кода из ПЗУ по адресу указанному в **РС**;
- расшифровка кода команды и подготовка к ее выполнению;
- инкремент содержимого **РС**;
- выполнение команды.

Реальный цикл процессора, конечно же, сложнее. В нем осуществляются проверки сигналов прерывания, извлечение, если нужно, дополнительных кодов команды или данных и другие операции.

Важной характеристикой цикла особенно при обслуживании реальных объектов является время выполнения команды. Большинство команд семейства **x51** выполняется за один так называемый машинный цикл, а некоторые – за два.

Внутренний бесконечный цикл процессора x51 имеет 12 этапов (стадий), поэтому время машинного цикла в 12 раз больше периода кварца. Например, при частоте кварца 12 МГц период цикла равен 1 мкс. Современные клоны x51 имеют меньшее число стадий и обычно работают в несколько раз быстрее.

7.3.2. Структура памяти команд (ПЗУ)

Память команд в большинстве случаев резидентна, то есть расположена внутри кристалла и имеет в зависимости от типа микросхемы размеры от 1 до 64 кбайт. Внешнее ПЗУ включается достаточно редко, например, в сложных или отладочных системах. В настоящее время для внутренней памяти команд чаще всего используется флэш-память (см. п. 14.1.4), двоичные коды в которую заносятся пользователем.

Но в любом случае в начальной зоне ПЗУ выделены особые ячейки, которые не рекомендуется использовать по-иному.

00h – для начального пуска программы,

*03h – для внешнего прерывания **Int0**,*

*0Bh – для прерывания по переполнению таймера **T0**,*

*13h – для внешнего прерывания **Int1**,*

*1Bh – для прерывания по переполнению таймера **T1**,*

23h – для прерывания по последовательному порту.

Запоминать в точности эти адреса не стоит, поскольку они входят в состав обязательной части текста любой программы (см. п. 7.12.4) и копируются из документации на ОВМ.

В некоторых ОВМ могут быть и другие выделенные служебные зоны ПЗУ. Например, сектор для процедуры загрузчика, который, как в «настоящем» компьютере, может загрузить внешнюю программу в память программ, например, через последовательный интерфейс (см. п. 7.3.3).

В этих особых ячейках программист записывает команды перехода (типа *Go To*) на начало программы и процедуры прерывания (см. п. 7.12.4). Зона особых ячеек ПЗУ в ОВМ с расширенным набором внутренней периферии имеет соответственно больший размер.

Область за пределами начальной зоны используется по усмотрению программиста для хранения кодов программы и таблиц констант (см. п. 7.12.5 и п. 7.12.8).

Для резервирования ячеек в определенном месте ПЗУ используются указания (псевдокоманды) инструментальной программе в виде так называемых *директив* ассемблера (см. п. 7.3.5).

7.3.3. Способы занесения кодов во внутреннее ПЗУ

Двоичные коды команд на заключительном этапе разработки должны быть занесены во внутреннюю память ОВМ. При этом могут использоваться различные методы.

В свое время чаще всего применялся внешний программатор, в который помещали кристалл ОВМ. Но подобная технология была не очень пригодна для модификации программ, поскольку приходилось устанавливать ОВМ в контактную розетку, которая снижала надежность системы.

Поэтому было разработано широко применяемое в настоящее время внутрисистемное программирование *ISP (In-System Programmable)*, когда для занесения двоичных кодов программы в запаянную на плате ОВМ используется специальный последовательный интерфейс, управляющий встроенным внутрь ОВМ программатором.

Развитием метода *ISP* стало так называемое самопрограммирование через штатный последовательный порт. В этом случае в специальную загрузочную зону ПЗУ предварительно и однажды, например, методом *ISP*, заносятся коды простой программы загрузчика. В дальнейшем пользователь может при необходимости активировать загрузчик для приема кодов новой программы от компьютера через встроенный последовательный интерфейс и сохранения их

в флэш-памяти. После занесения кодов загрузчик отключается, и штатный последовательный порт может использоваться уже загруженной программой пользователя.

При частой модификации программ флэш-память, как известно, деградирует (см. п. 14.1.5), что может нарушить работу системы. Поэтому для случаев с многократной перезаписью кодов программы стали использовать ОВМ с памятью команд в виде ОЗУ, внутреннего или внешнего (см. п. 14.6.4). Ячейки ОЗУ не теряют своей работоспособности при любом числе циклов записи. Загрузчик при этом, как и прежде, размещен во внутреннем ПЗУ.

7.3.4. Структура памяти данных

ОВМ в упрощенном виде представляется программисту набором различных 8-разрядных регистров или ячеек памяти, которые имеют сквозную нумерацию и размещены в памяти данных. Внутренняя память данных размером 256 байтов в стандартном варианте делится на две части (рис. 7.15) – область основного ОЗУ (для хранения пользовательских данных) и область регистров специального назначения SFR, управляющих внутренней аппаратурой ОВМ. Все 256 байтов нумеруются по порядку. И порядковый номер байта (ячейки) используется в машинных кодах программы, записанной в ПЗУ.

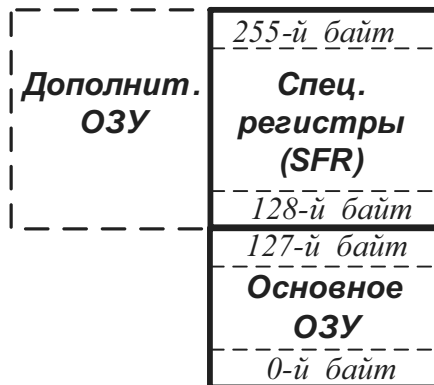


Рис. 7.15

Программист не обязан помнить эти номера, поскольку спецрегистры, как уже отмечалось, имеют стандартные имена, а для своих переменных программист обычно использует символические («говорящие») имена ячеек, например, mCount или mTemp, а конкретный номер этой ячейки в код команды уже подставляет программа ассемблера.

Некоторым семействам ОВМ стандартного пространства адресов спецрегистров (от 128 до 255) оказывается недостаточно, и в них используются страничные SFR, номер страницы которых задается битами одного из спецрегистров нулевой страницы.

Для увеличения памяти данных в некоторых модификациях ОВМ встраивается дополнительное ОЗУ с номерами такими же, как и спецрегистры, и более. В этом случае ячейки дополнительного ОЗУ имеют такую же адресацию, как и спецрегистры. Поэтому при их обслуживании используются различные механизмы вычисления адреса.

При операциях со спецрегистрами используется *прямая* адресация – адрес прямо указан в теле команды (см. п. 7.6.3), а при операциях с дополнительным ОЗУ – *косвенная*, когда адрес хранится в регистре-указателе (см. п. 7.6.4).

7.3.5. Навигация по ячейкам памяти

Когда программист пишет программу, в тексте он может давать указания ассемблеру, используя псевдокоманды (директивы) ассемблера. Эти директивы не будут превращаться в машинные коды, они служат для общего управления трансляцией текста программы в двоичные коды.

Чтобы директива отличалась от остального текста, она в большинстве случаев должна начинаться с точки или другого служебного знака.

Разные программы ассемблеров могут иметь разные обозначения для сходных директив. Но есть и общие черты. Например, для пояснения сути команд или отдельных фрагментов в текст программы включают комментарии, которые начинаются символом “;” и не учитываются программой ассемблера при кодировании. Знак комментария действует до конца строки.

Например, для того чтобы разместить программу или таблицу констант в памяти команд (кодовом сегменте ПЗУ), используют директиву

.CODE (или .CSEG)

которая указывает ассемблеру, что следующие строчки программы относятся к ПЗУ. Если программист переходит к работе с памятью данных (в сегменте данных ОЗУ), то необходимо использовать директиву

.DATA (или .DSEG)

Для точного указания ассемблеру номера (адреса) заданной ячейки в ПЗУ или ОЗУ используют директиву

.ORG *mn*

которая устанавливает внутренний указатель адреса в заданное значение *mn*.

Если этого не сделать, то программа ассемблера будет заполнять ячейки памяти по порядку, например, с нулевого значения.

Программист при написании текста программы может помечать некоторые ячейки памяти ПЗУ или ОЗУ, давая им осмысленные имена, которые называют *адресными метками*, которые должны заканчиваться двоеточием (см. п. 7.11.1).

*Другие директивы будут рассмотрены по ходу пояснений. Например, резервирование выбранных ячеек ОЗУ (при помощи директивы **DS**) описано в п. 7.5.2. Директива **DB** для размещения констант в ПЗУ и общие особенности ассемблерных программ описаны в §7.12.*

§7.4. Спецрегистры ОВМ x51

*В области спецрегистров **SFR** размещены не только регистры внутренней периферии, но другие регистры аппаратуры, тесно связанные с работой процессора - аккумулятор, регистр признаков и регистры-указатели, которые часто считаются неотъемлемой частью процессора, но для упрощения кодировки команд все они расположены в едином адресном пространстве **SFR**.*

Большинство регистров в соответствии с разрядностью семейства x51 являются 8-разрядными. Однако есть и несколько 16-разрядных – программный счетчик, указатель данных и два таймерных регистра-счетчика.

7.4.1. Аккумулятор

Само название регистра указывает, что в нем накапливается (образуется) результат операции. Аккумулятор, который имеет обозначение **A** или **Асс**, участвует почти во всех видах арифметических и логических операций в качестве места размещения одного из операндов.

*Отметим также, что ОВМ x51 имеет еще один арифметический регистр **B**, используемый при делении и умножении (см. п.п. 7.9.5 и 7.9.6).*

7.4.2. Регистр признаков

Хранит в своих триггерах так называемое слово состояния программы **PSW** (**P**rogram **S**tatus **W**ord), представляющее собой набор битов

(признаков, флажков) с некоторыми «обобщенными» результатами операций, которые могут использоваться по ходу работы программой или самим процессором. Расположение признаков в байте запоминать не нужно, потому что все биты регистра **PSW** доступны программисту по отдельности и имеют свои имена.

*Признак переноса **C** (Carry)* является одним из наиболее используемых признаков. Он устанавливается в «1», если при выполнении операции результат в аккумуляторе оказывается больше $2^8=255$. Говорят, что 8-разрядная сетка переполняется, то есть признак переноса является как бы 9 битом аккумулятора.

Нужно помнить, что не все команды при переполнении результата устанавливают признак переноса, и игнорирование этой особенности делает программу неработоспособной (см. п.п. 7.9.2 и 7.9.3).

Кроме того, бит **C** имеет и самостоятельное применение, иногда он используется в качестве 1-битового операнда в арифметических и логических командах (см. п.п. 7.9.2, 7.9.3, 7.10.4, 7.10.6).

*Вспомогательный перенос **AC** (Auxiliary Carry)* используется гораздо реже, при выполнении команды десятичной коррекции аккумулятора (см. п. 7.9.4), сигнализируя о переносе из младшей тетрады аккумулятора в старшую (переполнение 4 разрядов).

*Признак арифметического переполнения **OV** (Overflow)* используется в основном в целочисленных вычислениях со знаком, когда 8-й бит аккумулятора кодирует знак числа и разрядная сетка сужается до 7 битов. Поэтому **OV** = 1, если арифметический результат становится большим $2^7=127$. Кроме того, бит **OV** используется в командах умножения-деления, как индикатор особых случаев – деления на ноль и двухбайтного результата при умножении (см. п. 7.9.5 и 7.9.6).

Перечисленные признаки фиксируются (запоминаются) в триггерах регистра **PSW**. Причем процессор образует их (устанавливает или сбрасывает), как некий обобщенный результат выполненной арифметической или логической операции. Ранее запомненные признаки (со значениями «0» или «1») могут анализироваться другими программными фрагментами (если последующие команды не модифицируют признаки). То есть фиксируемые признаки могут передаваться между процедурами в качестве битового результата.

Кроме того, указанные признаки могут устанавливаться и сбрасываться программно.

7.4.3. «Нефиксируемые» признаки результата

Некоторые признаки контролируются в каждом машинном цикле по состоянию аккумулятора, то есть по сути, не хранятся в триггерах. К ним относятся признаки четности и нулевого результата.

*Признак паритета (четности) **P*** вычисляется как сумма по модулю 2 (см. п. 11.10.3) всех 8 разрядов аккумулятора. И его можно только читать, потому что формируется он только аппаратно.

Этот бит, являющийся индикатором нечетного количества единиц в аккумуляторе, используется иногда для контроля сохранности байта при удаленных пересылках через последовательный порт.

*Признак нулевого результата **Z (Zero)*** вообще не входит в перечень битов регистра **PSW**, что является отличительной чертой семейства **x51**. Признак **Z** (как и **P**) контролируется в каждом машинном цикле («непрерывно») по состоянию аккумулятора.

*С одной стороны, это приводит к определенному ограничению для программ – не «фиксируемое» в **PSW** значение **Z** нельзя передавать в качестве булевой переменной между фрагментами программы.*

Но с другой стороны признак нуля формируется не только после арифметико-логических команд, но и после команд пересылок, что позволяет в некоторых случаях упростить тестирование операнда на ноль.

Признак **Z** широко используется в условных командах передачи управления (см. п. 7.11.4).

7.4.4. Особые биты регистра признаков

Другой особенностью регистра **PSW** является то, что в нем размещены несколько битов, не относящихся к результатам операций. Они управляются только программистом.

Главными из них являются *биты-указатели **RS0** и **RS1*** (от **Registers Selector**) номера рабочего банка **РОН **RB0..RB3**** (см. п. 7.5.1) Эти два бита сбрасываются-устанавливаются программой и позволяют при необходимости выделить некоторым процедурам свой набор **РОН**, не доступный другим.

7.4.5. Регистры указатели

Основным из них является 16-разрядный *программный счетчик (указатель команд) **PC***, определяющий последовательность выполнения

команд программы. Его содержимое (от 0 до $2^{16}-1=65535$) не только определяет место расположения команды, но может указывать и массивы констант. Программный счетчик доступен побайтно по именам **PCH** и **PCL** (буква *H* от *High*, *L* от *Low*). Подробности использования **PC** – при изучении системы команд (см. §9.5).

Указатель данных DPTR (Data Pointer) – 16-разрядный, в него заносят адрес переменной или массива (со значением от 0 до 65535) для косвенной адресации данных во внутренней или внешней памяти (см. п. 7.8.3). Программист может использовать 16-битный указатель **DPTR** и побайтно в виде старшего **DPH** и младшего **DPL** байтов.

Указатель стека SP (Stack Pointer) имеет 8 разрядов, поскольку он ориентирован на использование в качестве стека только внутреннего ОЗУ. Особенности стека в п. 7.5.1, команды – п. 7.8.3.

Примечание. В качестве регистров указателей могут также использоваться два 8-разрядных регистра общего назначения, **R0** и **R1**, расположенных вне зоны **SFR** (см.п. 7.6.4).

7.4.6. Спецрегистры внутренней периферии (SFR)

Позволяют программисту управлять внутренней аппаратурой, обеспечивая необходимый режим и работу по программе.

Четыре цифровых регистра (порта) по 8 разрядов в каждом используются для вывода или ввода внешних логических сигналов и в текстах программ обозначаются именами **P0...P3**.

Два 16-разрядных таймера-счетчика, служащих для счета внутренних или внешних импульсов, в текстах программ используются побайтно, причем таймер 0 имеет имена **TH0** для старшего и **TL0** для младшего байта, а таймер 1 – **TH1** и **TL1**. Кроме того, таймеры имеют общие регистры **TCON (Timer CONTROL)** – для управления работой таймеров и **TMOD (Timer MODE)** – для задания режимов работы.

Последовательный порт, состоящий из приемника и передатчика, имеет два буферных регистра для приема и передачи с общим именем **SBUF (Serial BUFFER)** и регистр **SCON (Serial CONTROL)** для управления.

Схема прерываний обслуживается регистрами **IE (Interrupt Enable)** – для разрешения или запрета прерываний и **IP (Interrupt Priority)** – для управления приоритетом прерываний.

Форматы и побитный состав перечисленных спецрегистров и особенности их использования приведены в главе 10.

§7.5. Память данных – регистры и операнды

7.5.1. Структура основного ОЗУ

Эта область от 0 до 127 байта, которую иногда называют *пользовательской*, делится на четыре зоны (рис. 7.16).

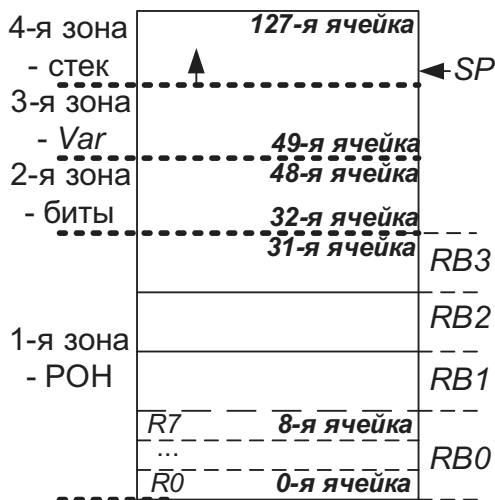


Рис. 7.16

Первая зона (ячейки от 0 до 31) содержит используемые системой команд регистры общего назначения POH. Всего таких регистров восемь, они имеют стандартные имена от **R0** до **R7**. Чтобы программные процедуры (например, при прерывании) имели защиту POH от модификаций, в ОЗУ организованы 4 отдельных так называемых банка POH с именами **RB0..RB3**. Какой из 4 банков является рабочим, определяется битами-селекторами **RS0** и **RS1** (см. п. 7.4.4).

Вторая зона (ячейки от 32 до 48) содержит область, где биты 16 байтов пронумерованы от 0 до 127. И, используя этот номер в символическом виде (*mFlagReady*, например), программист может проводить операции с отдельными битами. Эту зону иногда даже называют битовым (булевым) сегментом памяти (**BSEG**).

Номера битов от 128 до 255 используются для адресации битов в спецрегистрах.

Третья зона ОЗУ (область переменных) определяется пользователем, который резервирует место для своих переменных (см. п. 7.5.2).

Начало этой зоны и ее размер задаются программистом. Например, если битовых переменных в программе мало, то можно использовать незанятую часть битовой зоны и даже разместить переменные на месте незадействованных банков РОН.

Четвертая зона ОЗУ – область стека. Стек – это специально выделенная зона ОЗУ типа **LIFO (Last Input First Output** – последний вошел, первый вышел), используемая процессором для хранения адресов возврата при вызове подпрограмм или обслуживании прерываний. В стеке также можно хранить данные. Иллюстративным аналогом доступа к элементам стека является доступ к стопке книг, когда можно брать их сверху в порядке обратном заполнению.

Элементы стека не привязаны жестко к номерам ячеек, и их положение определяется при помощи ранее рассмотренного 8-битного указателя стека **SP**.

Особенностью стека x51 является то, что он заполняется в сторону увеличения адреса ОЗУ. Программист может изменить начало стека, и должен следить за размерами стека, чтобы он не вышел за пределы внутреннего ОЗУ.

*Если не задать начальное значение **SP**, то при сбросе стек будет размещен сразу после первого банка РОН и есть опасность того, что стек при своем расширении наложится на переменные пользователя, искажая их.*

7.5.2. Определение операндов пользователя

Резервирование места для операндов пользователя осуществляется директивой **DS (Define Size)**, которая указывает ассемблеру размер именованной области данных в байтах (переменных различного формата, массивов данных и пр.). Ассемблер назначает символическим именам порядковые адреса, начинающиеся с номера, указанного директивой **ORG**. Впоследствии эти номера используются в двоичных кодах команд, заносимых в ПЗУ.

Например, в фрагменте, приведенном ниже, директива **ORG** указывает начальный адрес (32), который присваивается переменной для хранения флажков по имени *mFlags* размером в два байта. Следующая байтовая переменная *mCountFaz* будет иметь порядковый номер больше на 2 и т. д.

.ORG 32	: шестнадцатиричное значение 20h
<i>mFlags</i> DS 2	; <i>mFlags</i> будет кодироваться номером 32
<i>mCountFaz</i> DS 1	; <i>mCountFaz</i> – 34
<i>mTemp</i> DS 1	; <i>mTemp</i> – 35

mBuflndik DS 4 ; mBuflndik – 36
mBeginStack DS 1 ; mBeginStack – 40

Особо отметим последнюю переменную, которая в определенной мере является фиктивной. Она не предназначена для хранения данных и нужна только для того, чтобы получить адрес начала области стека для инициализации указателя **SP** (см. п. 7.8.5).

Программист может изменить последовательное распределение адресов переменных, добавив там, где нужно, дополнительную директиву **ORG** с требуемым номером ячейки.

Отметим также, что доступ к элементам многобайтных переменных может осуществляться указанием имени переменной и смещения. Например, имя mBuflndik+1 указывает на ячейку с адресом 36+1, а mBuflndik+3 – на ячейку с адресом 36+3.

Проследить за назначением номеров отдельным операндам и кодировкой команд можно в файле протокола процесса ассемблирования, который генерируется программой ассемблера и имеет расширение .LST

7.5.3. Использование зарегистрированных имен

При именовании операндов можно использовать уже известные (зарегистрированные ассемблером) имена (стандартные или введенные пользователем) при помощи директивы **REG**. Например

mTemp1 REG R7
mPortIndik REG P0
mBitOut REG P1.3
mFlagReady REG mFlags.0

Как и прежде (см. п. 7.1.3), через точку нумеруются биты соответствующих регистров.

§7.6. Методы адресации операндов

Методы адресации задают процессору механизм вычисления адреса для определения месторасположения данных.

7.6.1. Непосредственная адресация

В этом случае операнд находится непосредственно в теле команды (в ПЗУ). Обычно это константы, которые нужно занести в регистры или ячейки памяти. Изменить их в ходе вычислений нельзя. Операнды с непосредственной адресацией отмечают значком **#** (ре-

шеткой), который ставится впереди числовой или символической константы.

В ассемблерных командах можно задавать шестнадцатиричные константы (**#0Dh**), двоичные (**#010011b**), десятичные (**#255**), символьные (**#'a'**).

Программа ассемблера сама производит нужные преобразования систем счисления, символов, имен и кодирует в теле команды второй байт непосредственного двоичного операнда.

*Настоятельно рекомендуется использовать символические имена для задания констант в программе, предварительно определив их при помощи макроопределения **EQU** (от **EQUal**), например, **myConst EQU 100**.*

В этом случае при модификации константы, используемой в нескольких местах программы, поправку вносят только в определение константы, что позволяет избежать ошибок при модификации программных модулей.

7.6.2. Регистровая адресация

Когда операнды находятся в **регистрах** общего назначения, говорят о регистровом типе адресации.

Следует отметить, что для ОВМ понятие оперативного регистра несколько иное, чем для процессоров без внутренней памяти. Процессор ОВМ имеет, по сути, в ОЗУ, размещенном во внутренней памяти, множество оперативных регистров. Поэтому для ОВМ **x51** под регистровыми операндами понимают только аккумулятор **A** и восемь регистров **R0..R7**.

При регистровой адресации код команды получается компактным, поскольку адрес РОН занимает только 3 бита, а код команды умещается в 8 битов.

В свое время, когда размер внутренней памяти был невелик, программисты старались применять именно регистровую адресацию. Однако тексты программ с регистровыми обезличенными переменными плохо читаются и нуждаются в дополнительном комментарии, поскольку имена регистров не несут смыслового указания на характер переменной.

*Поэтому в настоящее время, когда размер памяти в большинстве ОВМ стал больше, предпочтительнее использовать **прямоадресуемые** ячейки ОЗУ с «говорящими» идентификаторами или присваивать свои имена РОН. Тексты программ при этом получаются более наглядными.*

Однако отметим, что регистровая адресация широко используется компиляторами языков высокого уровня (см. п. 9.2.3) и в библиотечных функциях.

7.6.3. Прямая адресация

Используется при обработке одиночных переменных. В теле команды *прямо* указывается номер (от 0 до 255) ячейки памяти или бита, где хранится операнд. Как уже отмечалось, программист не имеет дела с числовыми адресами, он использует стандартные или свои символические имена ячеек и битов, такие как **TCON**, **TL1**, **mCountFaz**, **mBitOut** и др.

А программа ассемблера на основе текста программы сама присваивает численные значения именованным операндам и записывает их во втором или третьем байте двоичного кода команды. Поэтому прямоадресуемые операнды несколько уступают регистровым операндам по экономичности кода. Однако для спецрегистров прямая адресация (по стандартному имени) является единственной.

Следует также отметить, что на компактность кода в настоящее время внимания почти не обращают, потому что в большинстве случаев памяти хватает. А именованные переменные легче читаются и не требуют дополнительных комментариев.

7.6.4. Косвенная адресация

Адрес ячейки памяти находится в одном из регистров для косвенной адресации, в качестве которых допускается использовать 3 регистра: **R0**, **R1** и **DPTR**. Последний из них 16-разрядный и может адресовать и внешнюю память в пределах от 0 до 64 кбайт.

Чтобы пометить, что регистр содержит не сам операнд, а адрес операнда, перед регистром косвенной адресации ставится знак **@** (так называемое коммерческое **at** – при). Например:

@R0, @R1, @DPTR, @A+DPTR.

Последний пример использует более сложную разновидность косвенной адресации – по сумме содержимого аккумулятора **A** и регистра указателя **DPTR**. Она обычно используется для извлечения элемента массива, например, при табличном преобразовании (см. п. 11.5.3). В этом случае индекс элемента указывают в **A**, а базовый адрес массива заносят предварительно в **DPTR**.

*Неявную форму косвенной адресации по указателю **SP** имеют стековые команды (см. п. 7.8.4)*

Ячейки дополнительного ОЗУ, имеющие адреса, совпадающие с адресами спецрегистров (см. п. 7.3.4), доступны только при использовании косвенной адресации.

Соответственно для общения со спецрегистрами применяется только прямая адресация.

7.6.5. Относительная адресация

Адрес операнда вычисляется *относительно* программного счетчика **РС**. Этот вид адресации в основном используется в командах перехода (условных и безусловных), когда адрес перехода вычисляется как сумма программного счетчика и знакового смещения, указанного в теле команды.

Это очень удобно при перемещении частей программы. Когда ассемблер кодирует команду перехода, он автоматически вычисляет смещение и включает его в тело команды. Теперь этот фрагмент с относительной адресацией можно *перемещать* по всей памяти без дополнительной перекодировки, он будет работоспособным, так как в нем нет абсолютных адресов.

Кроме того, относительная адресация может использоваться при извлечении константного элемента из таблицы в ПЗУ. Операнд при этом записывается в следующем виде:

@A+PC,

и адрес образуется, как сумма содержимого аккумулятора **A** и счетчика **PC** (см. п.п. 7.8.3 и 11.5.3).

§7.7. Особенности ассемблерных команд

В следующих параграфах описан перечень основных команд для пояснения функционирования ОВМ и ее узлов. Далее будем придерживаться в основном рекомендаций, изложенных в ОСТ 110342.2-87 “Язык ассемблера АСМ51”.

7.7.1. Мнемокоды команд

Команды ОВМ, записываемые в ПЗУ, должны представлять собой двоичные (машинные) коды. Однако, программист работает с более удобными мнемоническими (ассемблерными) обозначениями команд, запомнить которые легче, так как они основаны на общепринятых в микропроцессорной технике сокращениях английских слов и несут смысловое указание на выполняемую командой функцию.

Набор таких мнемочкодов для процессора образует язык ассемблера семейства. Каждой команде ассемблера однозначно соответствует двоичный машинный код размером в один, два или три байта. Тогда как одному оператору языка программирования высокого уровня соответствует множество машинных кодов (см. п. 9.8.3).

Заметим, что производительность кода ассемблерной программы всегда выше производительности кода после компилятора языка высокого уровня, который проще для программиста при реализации сложных алгоритмов. Поэтому на практике часто используют комбинацию ассемблера (для быстрых процедур) и языка высокого уровня, например, Си (см. п. 9.2.5).

В пособии рассмотрены только основные черты языка ассемблера, чтобы более наглядно иллюстрировать процессы, происходящие внутри ОВМ.

На конечном этапе разработки тексты программ транслируют при помощи специальных компьютерных программ (кросс-ассемблеров или других инструментальных средств) в двоичные коды, которые заносятся затем во внутреннее ПЗУ (см. п. 7.3.3).

7.7.2. Особенности записи мнемочкода команды

Перед тем как рассматривать смысл и обозначения команд, укажем некоторые общие особенности, присущие ассемблерным командам в «интеловской нотации».

Кроме мнемонического обозначения большинство команд имеют два операнда и условно записываются в виде, представленном на рис. 7.17.

МнемоКоманда Оп1, Оп2

← - - - -

Рис. 7.17

Приемником информации всегда является операнд, записанный в теле команды первым (Оп1), то есть движение данных при обработке происходит справа налево (по иллюстративной стрелке), как при вычислении алгебраических выражений.

Например, для выражения $Y = Y + Z$ сумма значений Y и Z заносится в Y .

Нужно отметить, что существуют ограничения на типы операндов в команде. Одни из них естественные, например, в качестве Оп1 нельзя использовать константу, так как она не может изменяться. Но большая часть ограничений введена разработчиками семейства x51 из-за того, что для кодировки мнемочкодов команд использован байтовый формат, то есть числа от 0 до

255, и для команд с некоторыми комбинациями операндов не хватает кодов. В таких случаях говорят, что система команд не полностью симметрична по отношению к различным операндам. Даже в более сложных процессорах бывают определенные ограничения на использование в одной команде некоторых методов адресации.

Возможные виды записи операндов:

- с непосредственной адресацией – **#mConst**;
- с регистровой адресацией – **A, R0..R7**;
- с прямой адресацией – все имена спецрегистров и переменные пользователя, например, **mVar, mBit**;
- с косвенной адресацией – **@R0, @R1, @DP, @A+DPTR, @A+PC**;

Весь набор команд обычно подразделяют на группы, объединяющие функционально схожие и родственные команды, как-то: команды пересылок, арифметические, логические, передачи управления.

§7.8. Команды пересылки данных

7.8.1. Общий вид команд пересылки

Команды пересылки (копирования) данных чаще всего имеют мнемоническое сокращение, основанное на английском слове **move** (двигать, перемещать):

Mov Op1, Op2 ; Op1=Op2

В ОВМ x51 большинство пересылок происходит с участием аккумулятора, то есть один из операндов (или первый или второй) чаще всего **A**. Но есть и исключения, которые обычно не запоминают, поскольку при написании программ используют шпаргалку, например, в виде текстового файла с табличным перечнем команд.

Различные сочетания методов адресации при пересылках с участием аккумулятора легче усвоить из нижеследующего перечня однотипных команд.

7.8.2. Регистровые пересылки

между аккумулятором **A** и регистрами **R0..R7** являются самыми простыми:

Mov A, Ri ; **A = Ri**
Mov Ri, A ; **Ri = A**

где $i = 0..7$, то есть всего таких команд 16. Все они кодируются компактно, одним байтом.

*Нужно иметь в виду, что регистровые команды работают с одним из 4 регистровых банков **RB0..RB3** (см. п. 7.5.1), номер которого указан в битах **RS0, RS1** регистра признаков **PSW** (см. п. 7.4.4). При начальном сбросе эти биты обнуляются и устанавливается банк **RB0**.*

*Переключение банков осуществляется путем соответствующей установки или сброса битов **RS0, RS1** командами, которые рассмотрены далее (см. п. 7.10.6).*

Одной из форм пересылок с использованием регистровой адресации является команда обмена, когда аккумулятор и указанный регистр меняются своим содержимым. Чтобы подчеркнуть особенность такой взаимной пересылки в обоих направлениях, используется другой мнемокод (от слова **eXCHange** – обмен):

Xch A, Ri ; **Reg = A, A = Ri, Ri = Reg**

где **Reg** – вспомогательный (скрытый от программ) регистр.

К командам обмена обычно относят и команду перестановки содержимого тетрад в аккумуляторе:

Swap A

Младшая тетрада помещается на место старшей, а старшая – на место младшей.

7.8.3. Пересылки с косвенно-регистровой адресацией

Эти команды реализуются при помощи регистров-указателей, в которых задается адрес операнда.

Напомним, что в том случае, когда в регистре хранится адрес операнда, перед регистром ставят знак @. Кроме того, как и прежде, информация копируется из правого операнда в левый, то есть $Op1 = Op2$.

В первую очередь такими регистрами-указателями с байтовым форматом адреса (значения от 0 до 255) являются **R0, R1**, которые в дальнейшем будут обозначаться **Rk**, где $k = 0$ или 1.

Отметим, что, как и ранее, в пересылках участвует аккумулятор и всего имеется восемь таких команд, четыре для обслуживания *внутреннего ОЗУ*:

Mov A, @Rk ; A = @Rk
Mov @Rk, A ; @Rk = A

и четыре для *внешнего* ОЗУ (буква *X* в мнемокоде от *eXternal* – внешний):

MovX A, @Rk ; A = @Rk
MovX @Rk, A ; @Rk = A

Причем в последних четырех случаях содержимое регистра Rk выводится в качестве адреса через порт P0 и сопровождается внешним сигналом ALE (см. п.п. 14.6.1 и 14.7.1).

Затем в зависимости от направления передачи формируется сигнал RD (или WR) и внешнее ОЗУ устанавливает (или принимает) через порт P0 данные для (или из) аккумулятора (см. п.п. 14.6.3 и 14.7.1).

Можно использовать косвенную адресацию и в команде обмена содержимого аккумулятора и ячейки внутреннего ОЗУ:

Xch A, @Rk

Полезной командой в ряде случаев является команда, производящая обмен *только младших* тетрад аккумулятора и ячейки памяти, не имеющая других типов адресации:

XchD A, @Rk

Буква *D* в мнемокоде – от слова декада, которая кодируется, как известно, 4-мя битами (тетрадой).

Регистр-указатель данных **DPTR** является 16-разрядным и может работать с адресами от 0 до $2^{16}-1=65535$, что излишне для внутренней памяти, но удобно для внешней. Поэтому две команды пересылки в/из аккумулятора используют *косвенную адресацию* по содержимому указателя данных **DPTR** для адресации *внешней* памяти данных (см. п. 14.6.3):

MovX A, @DPTR
MovX @DPTR, A

Буква X в мнемокоде команд напоминает о том, что адресуется внешняя память и генерируются внешние сигналы.

7.8.4. Пересылки с прямой адресацией

по имени (номеру) ячейки памяти. Под номером будем понимать байтовую переменную пользователя или стандартное имя спецрегистра и обозначать далее обобщенным именем *mVarOrSFR*.

Пересылки с участием аккумулятора:

Mov A, mVarOrSFR	; A = mVarOrSFR
Mov mVarOrSFR, A	; mVarOrSFR = A
Xch A, mVarOrSFR	

Пересылки с регистрово-прямой адресацией:

Mov Ri, mVarOrSFR	; Ri = mVarOrSFR
Mov mVarOrSFR, Ri	; mVarOrSFR = Ri

Пересылки только с прямой адресацией:

Mov mVarOrSFR1, mVarOrSFR2	; mVarOrSFR1 = mVarOrSFR2

Пересылки с косвенно-прямой адресацией:

Mov mVarOrSFR, @Rk	; mVarOrSFR = @Rk
Mov @Rk, mVarOrSFR	; @Rk = mVarOrSFR
Push mVarOrSFR	; SP+1, @SP = mVarOrSFR
Pop mVarOrSFR	; mVarOrSFR = @SP, SP-1

В последних двух командах регистр-указатель **SP** не указан явно, он подразумевается в специальном мнемокоде команды (**Push** – занести операнд в стек, **Pop** – извлечь из стека в операнд). Кроме того, нужно обратить внимание на автоматический прединкремент-постдекремент содержимого **SP**.

7.8.5. Пересылка (загрузка) констант

в регистры и ячейки внутреннего ОЗУ осуществляется с использованием *непосредственной адресации* данных для операнда **Op2**, когда байтовая константа записана непосредственно в теле команды (в ПЗУ). В качестве обобщенного имени байтовой константы в дальнейших пояснениях будем использовать **mConst**.

Значение константы байтового формата записывается на месте правого операнда со значком **#**. Левый операнд при этом может использовать регистровую, косвенную или прямую адресации:

Mov A, #mConst	; A = #mConst
Mov Ri, #mConst	; Ri = #mConst
Mov mVarOrSFR, #mConst	; mVarOrSFR = #mConst
Mov @Rk, #mConst	; @Rk = #mConst

Примечание. В некоторых случаях значок **#** ставится не перед константой, а перед именем переменной, например, при инициализации указателя стека (см. п. 7.12.5):

Mov SP, #mBeginStack	; Заносится адрес ячейки
Mov SP, mBeginStack	; Заносится содержимое ячейки

*Первая команда копирует в регистр **SP** адрес переменной для инициализации. Во второй (ошибочной) команде из фиктивной переменной копируется ее ненужное содержимое.*

Засылка единой 16-разрядной константы **mConst16** (от 0 до 65535) возможна только в **DPTR**:

Mov DPTR, #mConst16

По этой команде осуществляется засылка в регистр **DPH** старшего байта константы (**#mConst16 / 256**), а в регистр **DPL** младшего байта (остаток от указанного деления).

7.8.6. Загрузка констант из ПЗУ с адресацией по сумме регистров

Используется при извлечении табличных констант из ПЗУ по индексу (номеру) элемента таблицы (массива). Значение индекса предварительно заносится в аккумулятор. Адрес в этом случае вычисляется как сумма индекса и базового адреса таблицы, который хранится в регистре **DPTR** или отсчитывается относительно программного счетчика **PC**. Например, по команде

MovC A, @A+DPTR

содержимое **A** и **DPTR** суммируются, определяя адрес в ПЗУ. После чего значение табличного элемента по вычисленному адресу копируется в аккумулятор (см. п. 11.5.3).

*Буква **C** в мнемокоде команд указывает, что адресуется память кодов команд.*

В другой подобной команде адрес вычисляется как сумма содержимых **A** и **PC**:

MovC A, @A+PC

что позволяет хранить константы вблизи с выполняемым программным кодом (см. п. 11.5.3). Директива **DB (Define Byte)** для размещения именованных массивов констант в ПЗУ рассмотрена в п. 7.12.8.

§7.9. Арифметические команды

7.9.1. Общие особенности

Результат арифметических операций чаще всего образуется в аккумуляторе, в котором должен быть размещен и один из операндов.

После вычисления результата также фиксируются обобщенные признаки результата **C**, **AC**, **OV** (см. п. 7.4.2). Кроме того формируются и не запоминаемые признаки **Z** и **P** (см. п. 7.4.3).

Следует также отметить, что признак переноса **C** не только устанавливается при переполнении разрядной сетки аккумулятора, но и может участвовать в качестве полноправного *однобитового слагаемого* в командах сложения-вычитания. Необходимость в этом возникает при обработке многобайтных чисел, когда перенос (или заем) из младшего байта передается следующей команде для суммирования со старшим байтом (см. п. 8.5.2).

В однооперандных командах типа инкремента-декремента результат заносится в указанный в мнемокоде операнд. Но *признаки результата* (и главное признак переполнения) в подобных командах *не формируются*.

7.9.2. Команды сложения (ADD)

Имеется несколько разновидностей этой команды.

Сложение без учета переноса – операнды суммируются и результат заносится в аккумулятор:

Add A, #mConst	; A = A + #mConst
Add A, Ri	; A = A + Ri
Add A, @Rk	; A = A + @Rk
Add A, mVarOrSFR	; A = A + mVarOrSFR

Сложение с последующим добавлением бита C (ADD with Carry) к младшему биту результата:

AddC A, #mConst	; A = A + #mConst + C
AddC A, Ri	; A = A + Ri + C
AddC A, @Rk	; A = A + @Rk + C
AddC A, mVarOrSFR	; A = A + mVarOrSFR + C

Инкремент (INCRement) осуществляет добавление единицы к содержимому операнда. Отметим, что все инкрементируемые операнды за исключением **DPTR** являются 8-разрядными:

Inc A	; A = A + 1
Inc Ri	; Ri = Ri + 1
Inc @Rk	; @Rk = @Rk + 1
Inc myVarOrSFR	; myVarOrSFR = myVarOrSFR + 1
Inc DPTR	; DPTR = DPTR + 1 (16 разрядов!)

Отметим две особенности команд инкремента – результат образуется в указанном операнде, а не в аккумуляторе, и признак переноса *не устанавливается* при переполнении.

7.9.3. Команды вычитания

Имеются две разновидности – собственно вычитание с учетом бита заема (роль которого выполняет бит переноса **C**) и декремент операнда.

*Вычитание с учетом заема (**SUB**traction with **Bur**ry):*

SubB A, #mConst	; A = A - C - #mConst
SubB A, Ri	; A = A - C - Ri
SubB A, @Rk	; A = A - C - @Rk
SubB A, mVarOrSFR	; A = A - C - mVarOrSFR

Важно отметить, что перед выполнением вычитания первого байта многобайтного или однобайтного числа признак переноса должен быть программно *обнулен*!

*Команда декремента (**DEC**rement)* вычитает единицу от содержимого операнда и не имеет, в отличие от команды инкремента, команды 16-разрядного декремента **DPTR**, то есть применима только для 8-разрядных операндов:

Dec A	; A = A - 1
Dec Ri	; Ri = Ri - 1
Dec mVarOrSFR	; mVarOrSFR = mVarOrSFR - 1
Dec @Rk	; @Rk = @Rk - 1

Необходимо обратить внимание на то, что в командах декремента (как в командах инкремента) признак переноса при переполнении не устанавливается. Больших неудобств эта особенность не создает, поскольку при организации циклов и в других подобных случаях можно использовать специальные двоенные команды (см. п. 7.11.5), но забывать об этом не следует.

7.9.4. Команда десятичной коррекции аккумулятора

Также относится к арифметическим командам и записывается в виде:

DA A

В мнемокоде команды **DA** указана её функция (**D**ecimal-**a**djust for **A**ddition – десятичная коррекция для сложения). Эта команда превращает двоичное 8-разрядное число в аккумуляторе *после сложения* в два двоично-кодированных числа (**BCD** – **B**inary **C**oded **D**ecimal), каждое со значениями от 0 до 9. Если корректируемое число больше 99, то устанавливается признак переноса.

Такой формат BCD чисел, когда в байте размещается сразу две декады (по 4 бита), называют упакованным, в отличие от неупакованного, когда декада кодируется целым байтом.

Отметим также, что при сложении двух BCD чисел максимальный результат не выходит за рамки значения 199 ($99 + 99 + \text{перенос}$), причем единица сотен будет кодироваться переносом.

Команда коррекции работает *только после команды сложения* и не применима для других команд (вычитания, пересылок и др.). Если все-таки нужно провести десятичное преобразование, например, перед сложением первого байта многобайтного числа, то используют фиктивное сложение с нулем и затем ставят команду коррекции.

7.9.5. Команда умножения

Работает с двумя операндами, которые могут размещаться только в двух арифметических регистрах **A** и **B** и нигде больше. Это подчеркивается мнемокодом команды, где буквы операндов написаны слитно:

Mul AB ; $A \cdot B = B$ – старший байт, **A** – младший байт

Результат 8-разрядного умножения также заносится в указанные регистры, причем регистр **B** является продолжением старших битов аккумулятора. Поэтому признак **C**, так называемый 9 бит аккумулятора, в этой команде не работает и всегда сбрасывается.

Индикатором переполнения аккумулятора становится признак арифметического переноса **OV**, который устанавливается, если результат получается большим 255, и означает, что регистр **B** – ненулевой.

7.9.6. Команда деления

Аналогично работает только с двумя арифметическими регистрами:

Div AB ; $A/B = A$ – результат, **B** – остаток

Результат деления образуется в аккумуляторе, а остаток в регистре **B**.

Особую роль, выполняет признак **OV**, он устанавливается при делении на значение $B = 0$, что позволяет программе выделить и при необходимости обработать этот особый случай.

§7.10. Логические команды

*Особенностью логических команд является то, что признаки результата, кроме признака **Z**, не формируются, поскольку в боль-*

шинстве случаев параллельно выполняются 8 отдельных операций над одноименными битами.

7.10.1. Команды логического умножения

Выполняют операцию логического умножения «И» (**ANd Logical**) над отдельными одноименными битами операндов. Битовые результаты этих восьми операций сохраняются в соответствующих разрядах аккумулятора:

ANL A, #mConst
ANL A, Ri
ANL A, @Rk,
ANL A, mVarOrSFR

Но есть две команды, в которых результат образуется в любой, прямо указанной (именованной) ячейке памяти внутреннего ОЗУ или **SFR**:

ANL mVarOrSFR, A
ANL mVarOrSFR, #mConst

Аналогичные исключения есть и в других логических командах.

7.10.2. Команды логического сложения

Реализуют операцию логического сложения «ИЛИ» (**OR Logical**) над отдельными битами

ORL A, #mConst
ORL A, Ri
ORL A, @Rk
ORL A, mVarOrSFR
ORL mVarOrSFR, A
ORL mVarOrSFR, #mConst

В четырех первых командах результат сохраняется в аккумуляторе, а в двух последних – в именованной ячейке памяти.

7.10.3. Команды «исключающего ИЛИ»

Реализуют сложение по модулю 2 (см. п. 11.9.1) для каждого из 8 разрядов двух операндов. Эта логическая операция имеет название «исключающего ИЛИ» и соответствующий мнемокод команды (от **eXclusive oR Logical**):

XRL A, #mConst
XRL A, Ri

XRL A, @Rk
XRL A, mVarOrSFR
XRL mVarOrSFR, A
XRL mVarOrSFR, #mConst

Относительно сохранения результата операции эти команды подобны предыдущим.

7.10.4. Команды ротации (сдвига)

работают только с одним операндом – аккумулятором, все биты которого *одновременно* сдвигаются влево или вправо, причем бит, выходящий за пределы разрядной сетки, вдвигается на освободившееся место с другой стороны и копируется одновременно в признак переноса *C*. Такой кольцевой сдвиг аккумулятора имеет две разновидности – ротацию влево (*Rotation Left*) и ротацию вправо: (*Rotation Right*)

RL A ; $A.n+1 = A.n$ (для $n=0..6$), $A.0 = A.7$, $C = A.7$
RR A ; $A.n = A.n+1$ (для $n=0..6$), $A.7 = A.0$, $C = A.0$

Еще в двух командах ротации (влево и вправо) признак переноса включается в кольцо, как 9-й бит аккумулятора, что отражается буквой **C** в мнемокоде:

RLC A ; $A.n+1 = A.n$ (для $n=0..6$), $A.0 = C$, $C = A.7$
RRC A ; $A.n = A.n+1$ (для $n=0..6$), $A.7 = C$, $C = A.0$

Особенности команд ротации иллюстрируются на рис. 7.18.

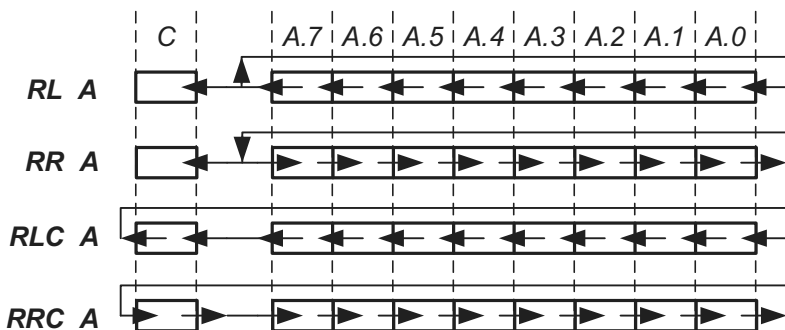


Рис. 7.18

7.10.5. Команды очистки и инвертирования аккумулятора

Также являются однооперандными. Мнемокоды команд основаны на сокращении английских слов **CLear** и **ComPLement**:

Clr A ; **A = 0**
Cpl A ; **A = $\bar{A}$**

Применив последовательно эти две команды, можно установить все биты аккумулятора в единичное состояние.

Отметим, что для очистки и инвертирования остальных операндов эти команды не применимы и приходится применять для обнуления, например, занесение нулевой константы

Mov mVarOrSFR, #0

а для инвертирования – команду «исключающего ИЛИ».

7.10.6. Команды побитовой обработки

Являются очень полезными при «мелких» действиях по обслуживанию внешних объектов и управлении внутренней периферией, когда приходится манипулировать отдельными битами.

Перечень таких битовых команд достаточно многочислен и включает пересылки, логические «И» и «ИЛИ», сброс, установку и инвертирование битов. Иногда, даже преувеличенно говорят о наличии битового процессора в семействе **x51**, тем более что этот «процессор» имеет своеобразный «аккумулятор» битового результата, в качестве которого выступает бит переноса **C**.

Пересылки, как и ранее, осуществляются из правого операнда в левый:

Mov C, mBitVarOrBitSFR ; **C = mBitVarOrBitSFR**
Mov mBitVarOrBitSFR, C ; **mBitVarOrBitSFR = C**

Логические операции над одиночными битами имеют мнемокоды, аналогичные байтовым:

ANL C, mBitVarOrBitSFR ; **C = C & mBitVarOrBitSFR**
ORL C, mBitVarOrBitSFR ; **C = C | mBitVarOrBitSFR**

Для простоты не приведены редко используемые логические команды с инверсными значениями именованных битов.

*Команды очистки (**CLear**), установки (**SET Bit**) и инвертирования (**ComPLement**) одиночных битов являются однооперандными и наиболее часто используемыми на практике:*

Clr C	; C = 0
Clr mBitVarOrBitSFR	; mBitVarOrBitSFR= 0
SetB C	; C = 1
SetB mBitVarOrBitSFR	; mBitVarOrBitSFR= 1
Cpl C	; C = Not(C)
Cpl mBitVarOrSFR	; mBitVarOrBitSFR= Not(mBitVarOrBitSFR)

Заметим, что очень удобные на практике команды установки-сброса-инвертирования отдельных битов применимы только для битовых переменных пользователя, расположенных в ячейках специальной зоны ОЗУ (см. п. 7.5.1), и в SFR для побитнодоступных спецрегистров. Многие регистры (и даже некоторые ОВМ в целом) не имеют возможности работы с отдельными битами. В этом случае нужно использовать байтовые логические команды, в которых операции над битами производятся одновременно в 8 разрядах (см. §8.3).

§7.11. Команды передачи управления

7.11.1. Общие сведения

При помощи команд передачи управления организуют ветвления и переходы к различным фрагментам программы. Результат действия таких команд проявляется в изменении содержимого программного счетчика **PC**, который, как известно, определяет последовательность выполнения программы.

Различают два типа передачи управления: *команды перехода* (с мнемоникой, основанной на слове **JuMP** – прыгать) и *команды вызова подпрограмм* (с мнемокодом **CALL** – вызывать).

При выполнении команды перехода на новый адрес старый, записанный ранее в программном счетчике, теряется безвозвратно.

При вызове подпрограммы командой или аппаратурой (через механизм прерываний) содержимое программного счетчика, указывающее на следующую перед передачей управления команду, сохраняется в стеке как адрес возврата. Последней командой в подпрограмме должна быть команда возврата (см. п. 7.11.6), по которой запомненный ранее адрес из стека заносится в счетчик **PC**, и возобновляется последовательная работа по программе.

Прежде чем рассматривать команды передачи управления, отметим, что в тексте программы используются чаще всего символические, а не числовые адреса. Чтобы указать точку для перехода, записывают в нужном месте программы символическое имя

адреса (метку), обязательно заканчивающееся знаком «двоеточие», по которому программа ассемблера определяет, что имени нужно поставить в соответствие адрес. Примеры задания адресных меток:

*myLabel1:
myLevel:
myReadPort:*

В некоторых ассемблерах двоеточие можно опускать, если имя метки начинается с первой позиции строки. Но в таких ассемблерах нужно помнить, что любой мнемокод в первой позиции (даже мнемокоманда) будет трактоваться ассемблером как метка.

7.11.2. Разновидности команд перехода

Переходы делятся на *безусловные* и *условные*. В безусловных командах переход на указанный адрес осуществляется всегда, а в условных зависит от выполнения условия в команде.

Кроме того переходы бывают *абсолютными* или *относительными*. В первом случае в программный счетчик заносится адрес, записанный в теле (коде) команды. Во втором – адрес перехода *вычисляется относительно* текущего содержимого **РС**, например, добавляется знаковое смещение, указанное в коде команды.

Кроме того, адрес перехода может кодироваться 8-ю, 11-ю или 16-ю битами, что определяет максимальную длину перехода («прыжка»). В соответствии с указанными размерами кодировки различают:

- *короткие* (на ± 127 байтов – могут измениться 8 битов **РС**);
- *внутрисегментные* (в пределах сегмента размером 2048 байт – могут измениться 11 битов **РС**);
- *длинные* переходы по всему адресному пространству OBM 64К (могут измениться все 16 битов **РС**).

7.11.3. Безусловные переходы

Самое первое место в командах перехода традиционно принадлежит так называемой *пустой* команде

NOP

не производящей никаких действий (**No OPeration**), кроме увеличения содержимого программного счетчика для извлечения кода следующей команды.

Эта команда может оказаться полезной при создании коротких пауз в работе программы (см. п. 9.8.2) или при наладке аппаратуры.

Короткий безусловный переход (Short JuMP):

SJMP mLabelShort

является *относительным*, что позволяет кодировать команду двумя байтами. В первом байте размещается код мнемокоманды, а во втором – разность между текущим адресом и адресом метки *mLabelShort*, которая должна быть расположена вблизи команды перехода, не далее ± 127 байтов, и ее поэтому называют короткой.

При выполнении команды к текущему содержимому счетчика **PC** добавляется второй байт, который интерпретируется как 7-разрядное число со знаком. В результате алгебраического сложения в программном счетчике образуется полный 16-разрядный адрес метки.

Двоичные фрагменты (например, библиотечные процедуры) с подобными относительными командами перехода могут размещаться без пересчета кодов адресов в любых местах памяти программ.

Длинный безусловный переход (Long JuMP) (в пределах 64к):

LJMP mLabelLong

является *абсолютным* (не вычисляемым), поскольку в программный счетчик **PC** заносится конкретный двухбайтовый адрес длинной метки *myLabelLong*. Двоичные коды команд этого типа при перемещении двоичных фрагментов программ *должны быть пересчитаны*.

Промежуточной, в смысле перемещаемости, является команда так называемого *внутрисегментного абсолютного* перехода (внутри *текущего* сегмента размером 2048 байтов):

AJMP mLabel

С одной стороны, по своему механизму эта команда является абсолютной. Адрес не вычисляется, а просто копируются 11 битов кода команды в **PC**, то есть обеспечивается переход только в пределах сегмента размером $2^{11} = 2048$ байт.

С другой стороны, старшие 5 битов адреса при выполнении команды остаются неизменными, то есть управление не будет передано за пределы *текущего* сегмента памяти команд. Эта особенность позволяет перемещать двоичные фрагменты с такими командами, но не произвольно по адресному пространству, а выровненными на границу сегмента.

Чтобы сделать кодировку такого перехода более компактной (двухбайтной), разработчики ОВМ разместили старшие 3 бита в первом байте кода команды. Это привело к тому, что одной мнемокоманде соответствуют 8 (2^3) различных кодов. Но для

программиста эти 8 кодов обычно скрыты, поскольку кодировка осуществляется программой ассемблера.

Вычисляемый безусловный переход:

JMP @A+DPTR

осуществляет передачу управления по адресу, равному сумме аккумулятора и регистра-указателя **DPTR**. Он является длинным (по всей памяти размером 64к), и может быть использован для многовариантного выбора пути ветвления, например, подобно структуре *Switch* в языке Си.

Такой косвенный переход может быть по механизму и абсолютным и относительным (перемещаемым). Если в регистре DPTR записывается константа, то эту константу при перемещении нужно пересчитывать. Если в регистр DPTR программа предварительно копирует содержимое счетчика PC, то фрагменты с такими кодами можно перемещать без пересчета адреса.

7.11.4. Условные переходы

Предварительно осуществляют анализ отдельных признаков или прямо адресуемых битов, включая именованные биты ОЗУ и спец-регистров.

Если условие, записанное в мнемокоде команды, не выполняется, то программный счетчик работает в обычном инкрементном режиме и процессор переходит к следующей по порядку команде.

Если условие выполняется, то к содержимому **PC** добавляется знаковое смещение и происходит переход к новому фрагменту программы.

То есть условные переходы по механизму вычисления адреса являются *короткими и относительными*.

Ассемблерные условные переходы предназначены для реализации ветвления программы подобно оператору *IF* в языках высокого уровня, только условие ветвления должно быть предварительно подготовлено в битах регистра **PSW** или в других именованных битах.

Приведем перечень команд условного перехода с краткими комментариями.

JZ mLabelShort ; Переход, если Z=1 (содержимое **A** равно нулю)
JNZ mLabelShort ; Переход, если Z=0 (содержимое **A** не равно нулю)

Нужно помнить, что признак **Z** относится только к содержимому аккумулятора и не фиксируется, поэтому его нельзя передавать междупрограммными фрагментами.

JC *mLabelShort* ; переход, если перенос C=1
JNC *mLabelShort* ; переход, если перенос C=0

Более же универсальными и чаще применяемыми, но такими же, по сути, командами являются переходы по состоянию любого прямо указанного бита:

JB *mBitOrBitSFR, mLabelShort* ; переход, если бит =1
JNB *mBitOrBitSFR, mLabelShort* ; переход, если бит =0
JBC *mBitOrBitSFR, mLabelShort*

Последняя команда выполняет переход при единичном значении бита, а потом обнуляет бит.

Некоторые процессоры могут не иметь подобных команд анализа отдельных битов и приходится заменять их двумя командами – обнулением незначащих битов при помощи команды логического умножения (см. п. 7.10.1) и условным переходом по ненулевому результату.

Повторно нужно отметить, что условные переходы реализуются только в пределах ± 127 байтов. Если метка расположена дальше, то ассемблер делает соответствующее предупреждение.

Для организации длинных условных переходов необходимо пользоваться дополнительной командой длинного безусловного перехода, на месте короткой метки:

mLabelShort :
LJMP *mLabelLong*

7.11.5. Сдвоенные условные команды перехода

выполняют две операции как одну. Например, команды *декремента* содержимого операнда *и* *перехода* по ненулевому результату

DJNZ *Ri, mLabelShort*
DJNZ *mVar, mLabelShort*

эквивалентны последовательному действию двух команд **DEC** и **JNZ**.

Но особенность сдвоенных команд заключается в том, в них не используется признак нуля Z в явном виде, который непрерывно контролирует содержимое только аккумулятора.

Сдвоенные команды декремента широко используются для организации циклов упрощенно подобным оператору языка Си **do-while** (см. п. 9.8.5). Например, нижеприведенный фрагмент выполняет тело цикла 10 раз:

Mov mCount, #10
mLoop: ; здесь тело цикла
DJNZ mCount, mLoop

Другой тип вдвоенных команд – *сравнение* операндов (вычитание без сохранения результата) и *переход*, если операнды не равны:

CJNE A, #mConst, mLabelShort
CJNE Ri, #mConst, mLabelShort
CJNE @Rk, #mConst, mLabelShort
CJNE A, mVar, mLabelShort

Отметим, что три вдвоенных команды компарирования и перехода сравнивают операнды с константой, и только одна команда позволяет сравнивать содержимое аккумулятора и прямоадресуемой переменной.

7.11.6. Вызовы подпрограмм

могут быть программными или аппаратными (при помощи запросов аппаратуры на прерывание). В любом случае при переходе на подпрограмму значение программного счетчика (2 байта), указывающее на следующую команду автоматически сохраняется в стеке.

Как и переходы, вызовы подпрограмм могут быть длинными или внутрисегментными. Коротких нет.

Длинный вызов подпрограммы своему механизму является абсолютным, то есть модифицирует все 16 разрядов программного счетчика **PC** и имеет в мнемокоде первую букву **L** (от **Long**):

LCALL mLabelLong

Внутрисегментный вызов изменяет только 11 битов программного счетчика и поэтому допускается передвигать фрагменты с его кодами, выровненными на границу кодовых сегментов по 2 килобайта.

ACALL mLabelA

Первые буквы перечисленных команд **L** и **A** обычно опускают, ассемблер сам использует нужную команду. Но в перемещаемых модулях желательно прямо указывать внутрисегментный вызов, если это возможно.

Условные вызовы подпрограмм организуются при помощи команд условных переходов или косвенного (вычисляемого) перехода.

Последней командой подпрограммы обязательно является команда возврата

Ret ; ,

которая восстанавливает из стека прежнее значение программного счетчика (2 байта) для продолжения программы.

Для подпрограмм обработки прерывания необходимо использовать разновидность команды возврата с буквой *I* (от *Interrupt*) в конце:

RetI ; ,

действие которой почти такое же как команды ***Ret***. Она также извлекает из стека в программный счетчик 2 байта адреса, но в отличие от обычной команды возврата может сбрасывать некоторые аппаратные флажки системы прерываний (см. §8.4).

§7.12. Оформление ассемблерной программы

7.12.1. Структура ассемблерной программы

Как уже отмечалось, программист реализует требуемый алгоритм на основе набора мнемонических обозначений команд ОВМ, последовательность которых образует текст программы. Этот текст, кроме собственно команд и директив ассемблера, содержит символические имена констант, адресов и переменных.

На заключительной стадии разработки нужно от ассемблерного текста перейти к машинному отображению программы, то есть каждую мнемокоманду, константу, адрес перехода и переменную преобразовать в двоичный код. Этот процесс преобразования текста в последовательность двоичных кодов, которые заносятся во внутреннее ПЗУ, называют обычно трансляцией, и выполняется он при помощи специальных программ (кросс-ассемблеров) для соответствующего типа ОВМ.

Чтобы ассемблер правильно осуществлял трансляцию, текст программы должен записываться с учетом определенных правил, которые зависят от типа ассемблера. Но существуют и несколько общих соглашений, например, символический адрес (метка) должен заканчиваться двоеточием, а текст комментариев к фрагменту программы на каждой строке должен начинаться точкой с запятой.

Кроме команд и комментариев, в текст программы обычно включают псевдокоманды (директивы) для ассемблера, которые не преобразуются в двоичные коды, но помогают программисту руководить ходом трансляции.

Например, директивой **END** программист дает ассемблеру указание об окончании трансляции. Другие директивы по мере необходимости будут рассмотрены далее.

Более мощные кросс-средства предполагают наличие редактора внешних связей (LINK), позволяющего включать в программу модули, разработанные независимо друг от друга, и пересчет адресов перемещаемых программных модулей на абсолютные адреса.

Эти и другие особенности оформления текста программы иллюстрируются ниже на примере обобщенной структуры программы для ОВМ **x51**, которая должна иметь следующие текстовые блоки:

- определение констант (их в принципе можно размещать по тексту),
- резервирование места для переменных в памяти данных,
- заполнение особых ячеек (таблицы векторов прерываний) в ПЗУ,
- собственно текст программы и ее процедуры.

7.12.2. Определение констант и распределение внешних выводов

Директива **EQU** присваивает символическому имени числовое значение или значение вычисляемого выражения с заранее определенными аргументами, которая записывается в виде:

*Имя **EQU** константа или выражение*

Примеры присвоений:

<i>mKvoFaz EQU 6</i>	<i>; Количество фаз у двигателя</i>
<i>mFreq EQU 11059000</i>	<i>; Частота кварца</i>
<i>mTickT0 EQU mFreq / 12</i>	<i>; Частота обновления таймера</i>
<i>mIndFreq EQU 200</i>	<i>; Обновление индикатора 200 раз в секунду</i>

Распределение битов портов ввода-вывода и битов других регистров удобно проводить, используя специализированную директиву **REG**, которая устанавливает (см. п. 7.5.3) соответствие между указанным именем (регистра, переменной или бита) и стандартным именем регистра (или другим ранее введенным и известным программе именем).

*mBitSegmentA **REG** P1.0*
*mBitDogTimer **REG** P3.6*
*mBitKnopkaPravo **REG** P2.1*
*mPortIndik **REG** P1*

7.12.3. Резервирование переменных в памяти данных

Для работы с ячейками памяти данных (ОЗУ) нужно указать ассемблеру в тексте программы тип памяти и начало области пользователя при помощи директив **.DATA** и **.ORG** (см. п. 7.3.5) и объявить нужные переменные директивой **DS**, которая символическому имени присваивает конкретный порядковый адрес и резервирует определенное количество ячеек памяти (см. п. 7.5.2).

Например:

.DATA		; Объявляем тип памяти.
.ORG 20h		; Начинаем с области битовых ячеек (см. п. 7.5.1).
mFlags DS 1		; Резервируем 8 битов-флажков,
mFlagLevo REG myFlags.0		; именуем 0-й бит
mFlagFazaOn REG myFlags.2		; и другие биты...
mCountFaz DS 1		; Счетчики текущей фазы
mCountInt0 DS 1		; и числа прерываний Int0.
mBufIndik DS 4		; Буферы для индикации данных,
mBufT DS 6		; для передачи пакета данных,
mBufR DS 6		; для приема пакета данных.
mBeginStack DS 1		; Фиктивная ячейка – дальше стек

7.12.4. Заполнение особых ячеек в ПЗУ

Таблица векторов прерываний (см. п. 7.3.2) является неотъемлемой частью двоичного кода программы, своеобразным «игрушечным» диспетчером для внутренних устройств ОВМ. Эту таблицу желательно заполнять полностью, записывая команды перехода на прерывающие процедуры или команды возврата, если данное прерывание не используется:

.CODE		; Работаем с памятью команд.
.ORG 0		; При сбросе счетчик PC обнуляется и управление
Jmp mBeginProg		передается на начало программы
.ORG 3		; Ячейка для передачи управления
Jmp mProcInt0		; процедуре прерывание Int0.
.ORG 0Bh		; Ячейка для передачи управления
Jmp mProcT0		; процедуре прерывания по таймеру 0.
.ORG 13h		; Остальные
Retl		; прерывания
.ORG 1Bh		; в данном
Retl		; примере
.ORG 23h		; не
Retl		используются.

7.12.5. Текст основной части программы

желательно для наглядности представлять в виде *набора подпрограмм и макросов* (см. п. 7.12.6) с именами, несущими информацию о выполняемых функциях, и явно выделенным основным циклом алгоритма, например:

```

mBeginProg:
    Mov SP, #mBeginStack-1 ; Инициализация SP обязательна!
    Call mInitTimer         ; Инициализируем таймеры
    Call mInitInt           ; и систему прерываний.
mMacroInitVar mPar1, mPar2 ; Это макрос (см. п.7.12.6)
mLoopMain:
    ....                   ; Здесь основной цикл программы
    Jmp mLoopMain

```

В главном цикле программы лучше всего делать фоновую (неспешную) работу или вообще оставить пустым. Все действия, связанные с событиями или временем необходимо вынести в процедуры обработки прерываний. Но и перегружать систему прерываний не стоит, чтобы ни один из процессов не монополизировал вычисления.

7.12.6. Подпрограмма и макрос

Для лучшего понимания алгоритма желательно логически связанные фрагменты программы именовать и использовать в виде подпрограмм или макросов. Такое разбиение программы на отдельные части повышает читаемость ассемблерных текстов, приближая их к языкам высокого уровня.

Подпрограмма заносится в память в виде двоичного набора кодов команд однажды, но может вызываться программой много раз при помощи команды **Call**. Поэтому использование подпрограмм уменьшает размер используемой памяти команд.

Макрос представляет собой именованный фрагмент программы, обрамленный при помощи служебных слов **.macro** и **.endm**. Встретив имя макроса, ассемблер подставляет коды его команд в тело программы. Сколько раз встретился макрос, столько фрагментов будет подставлено. Поэтому макрос уступает подпрограмме в экономичности двоичного кода. Но при выполнении макроса процессор не делает лишнюю работу – не сохраняет в стеке адрес возврата и не извлекает его. Кроме того, при помощи макросов легко передавать аргументы.

Пример макроса для инициализации переменных константами *mArg1* и *mArg2*:

```
.macro mMacroInitVar mArg1, mArg2
    Mov mVar1, #mArg1
    Mov mVar2, #mArg2
.endm
```

Если внутри макроса используются адресные метки, то они предваряются специальным префиксом, например, \$. Следует также отметить, что некоторые ассемблеры некорректно обрабатывают макросы с комментариями на кириллице.

7.12.7. Тексты используемых процедур

Тексты процедур и макросов можно размещать в любом месте: либо до основной программы, либо после (если ассемблер это позволяет).

*Процедуры прерывания, вызываемые аппаратно, имеют несколько отличий от обычных подпрограмм, которые рассмотрены в п. 8.5.1. Здесь же мы ограничимся чисто внешним отличием – последней командой возврата (**Ret** или **Retl**).*

В качестве примера ниже приведено обрамление процедур прерывания, используемых в таблице особых ячеек ПЗУ, и обычных подпрограмм:

```
mProcInt0:
    .....
    Retl                                     ; Здесь текст процедуры.

mProcT0:
    .....
    Retl

mInitTimer:
    .....
    Ret
mInitInt:
    .....
    Ret
```

7.12.8. Размещение массивов констант в ПЗУ

может осуществляться с помощью директивы **DB** (**Define Byte**), которая помещает следующий за ней байт или последовательность байтов (в числовом или символическом представлении) в ячейки ПЗУ по порядку следования. Перед директивой **DB** обычно задают месторасположение массива констант при помощи директивы **ORG** и ставят адресную метку для именования адреса массива:

.ORG mSizeROM - 30 ; Разместим в конце ПЗУ
mLabel: **DB** 'Copyright SFMEI, 2011' ; имя правообладателя.

7.12.9. Общие советы

Начинать программирование желательно с общего описания работы программы в виде благих намерений по отдельным пунктам. Эти пункты можно записать в виде комментариев в тексте будущей программы. Допустимо использование операторов Паскаля, Си и математических функций. Затем пункты, которые понятно, как делать, записывать в виде реальных фрагментов программ.

Обязательно нужно структурировать программу, разбивая ее на модули, оформленные как подпрограммы или макросы. Их легче читать и отлаживать по отдельности. Кроме того, меньше будет проблем при групповой работе над большой программой.

Размер каждой процедуры, включая главную, должен иметь не более 20–25 строчек и помещаться на экране, чтобы для понимания ее сути не нужны были лишние перемещения по тексту.

В программах рекомендуется использовать «говорящие» имена переменных и адресных меток.

Внутренние метки процедур желательно начинать с той же буквы или нескольких букв, что и саму процедуру, добавляя только смысловую часть или цифровые индексы.

Не жалеть комментариев для описания общего смысла действий, а не отдельных операций.

Не использовать регистры **R0..R7** (за исключением регистров косвенной адресации **R0, R1**), требующие дополнительных напоминаний, что в них хранится. Вместе с тем при оформлении библиотечных функций использование РОН имеет определенный смысл, поскольку процедурам разного уровня можно выделять отдельные банки регистров (см. п. 7.5.1) для своих вычислений, не затрагивающие чужих регистров.

8 УПРАВЛЕНИЕ ВНУТРЕННЕЙ АППАРАТУРОЙ ОВМ x51

В этой главе описываются возможности, режимы работы и особенности управления основными внутренними узлами ОВМ – параллельными портами, таймерами, системой прерываний и последовательным портом. Изложенный материал может оказаться полезным при изучении принципа действия более сложных процессорных семейств.

§8.1. Вывод информации битовыми командами

Обслуживание внешних объектов осуществляется через порты ОВМ. Рассмотренные ниже манипуляции с отдельными битами и группами битов портов можно использовать при работе и с другими прямоадресуемыми переменными.

8.1.1. Особенности адресации битов

Существуют три способа представления битовой информации в ОВМ.

Первый из них используется в семействе x51, где выделены специализированные битовые регистры (см. п. 7.5.1) – 16 смежных байтовых ячеек в ОЗУ, начиная с адреса 32, и 16 спецрегистров, находящихся в области SFR. Все биты указанных ячеек последовательно пронумерованы от 0 до 255 и их номера (имена пользователя и стандартные имена битов) используются в специальных битовых командах в качестве второго байта машинного двоичного кода, обеспечивая доступ к указанному в коде биту.

Чтобы не путать имена байтов и битов семейства x51, рекомендуется в имена битовых переменных включать префикс mPin (Pin – штырек) для битов портов или mFlag для именованных битов ОЗУ.

В тех же ОВМ, где нет подобных команд, работающих с отдельными битами, приходится использовать команды с байтовым (или иным) форматом данных. При этом доступ к биту однозначно определяется его положением в байте. Отсюда следуют и другие способы адресации битов, привязанные уже не к определенным местам в ОЗУ, а к любой из переменных.

При *втором* способе адресации положение бита указывается номером бита в переменной (для байта от 0 до 7). Во избежание ошибок в этом случае желательно включать в имя бита префикс *mNomerBit* или иное смысловое указание.

Третий способ кодирования положения бита основан на указании веса или значимости бита. Для байта это будут двоично-взвешенные числа 1, 2, 4, 8, 16, 32, 64, 128. Имена битов в этом случае также должны отличаться от обычных, например, иметь префикс *mVesBit*.

Особенности использования битов в байтовых командах рассмотрены в §8.3.

Наличие указанных трех возможных способов адресации положения битов может привести к трудновывяемым ошибкам в программе, особенно при неправильном использовании стандартных имен битов спецрегистров, часть из которых не доступна битовым командам. Во избежание путаницы настоятельно рекомендуется изучать включаемые файлы инструментальных программ.

8.1.2. Установка, сброс и инвертирование отдельных битов

Стандартные имена битов портов пользователь для наглядности обычно переименовывает при помощи директивы **REG** (см. п. 7.5.3 и 7.12.2) в соответствии с именами сигналов своей схемы. Эти сигналы в дальнейшем можно устанавливать, сбрасывать или инвертировать в программе соответствующими битовыми командами, например:

```
SetB mPinCLK           ; Установка,
Clr RI                  ; сброс
Cpl mFlagReady          ; и инвертирование битов
```

Аналогичные команды можно использовать и для именованных битов в ОЗУ и спецбитов внутренней аппаратуры.

8.1.3. Вывод (копирование) групп битов

В некоторых случаях битовую информацию, вычисленную по ходу программы, необходимо копировать в произвольные биты одного или даже *разных* портов. Подобную операцию проводят обычно при помо-

щи команды пересылки бита в признак переноса (см. п. 7.10.6). Пусть для примера биты *разных* портов именованы, как *mPin0*, *mPin1*, *mPin2*, и нужно скопировать в них определенные биты аккумулятора:

```
Mov C, Acc.0
Mov mPin0, C
Mov C, Acc.1
Mov mPin1, C
Mov C, Acc.2
Mov mPin2, C
```

Биты аккумулятора в данном фрагменте могут быть и непоследовательными. Подобный способ применим не только для аккумулятора, но и для переменной, определенной в битовой области ОЗУ.

Копирование последовательно расположенных битов аккумулятора можно производить и через операцию сдвига (влево или вправо в зависимости от нужного порядка) в бит переноса (см. п. 7.10.4).

```
RLC A
Mov mPin0, C
RLC A
Mov mPin1, C ; и т. д.
```

Операции сдвига-копирования бита могут применяться в цикле, например, при использовании последовательного формата (см. далее).

8.1.4. Вывод битов в последовательном формате

Для экономии внешних выводов ОВМ информационные биты могут передаваться друг за другом последовательно во времени через один вывод порта.

Разновидностью подобной последовательной посылки является *асинхронная* (рис. 8.1), особенностью которой является обрамление информационных битов нулевым стартовым и единичным стоповым битом. Обычно такие передачи реализуют аппаратными средствами, используя внутренний последовательный порт (см. §8.8). Но в некоторых случаях приходится такую последовательность битов генерировать программно.

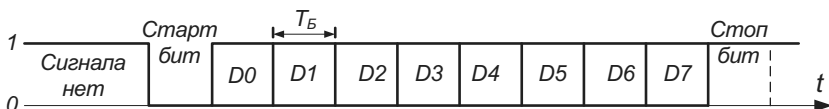


Рис. 8.1

В начале программа должна сформировать стартовый бит, сделать паузу длительностью $T_{\text{Б}}$, затем в цикле поочередно, начиная с младшего, передать с паузой 8 битов из аккумулятора и в заключение установить стоповый бит. Для организации цикла в нижеприведенной подпрограмме использована двоякая команда декремента и перехода по ненулевому результату (см. п. 7.11.5):

```

mSend:
  Clr mPinTxD                               ; Старм-бит.
  Call mDelayTb
  Mov mCount, #8                             ; Задаем количество битов.
myS1:                                       ; Цикл передачи битов.
  RRC A                                       ; Очередной бит – в перенос,
  Mov mPinTxD, C                             ; а затем в бит порта.
  Call mDelayTb
  DJNZ mCount, myS1                         ; Все ли биты передали?
  SetB mPinTxD                             ; Стоп-бит.
  Call mDelayTb
  Ret

```

Для реализации процедуры *mDelayTb*, определяющей период $T_{\text{Б}}$, можно использовать средства, описанные, например, в п. 8.7.4. Более подробно особенности программной задержки описаны в §9.8.

§8.2. Ввод информации битовыми командами

Биты портов, предназначенных для ввода информации, должны быть предварительно установлены в единицу или, как говорят, запрограммированы на ввод (см. п. 7.2.2). Обычно это делается автоматически после поступления импульса сброса. Но в некоторых случаях бит порта служит для двунаправленного информационного обмена. В подобных случаях перед чтением данных следует установить этот бит в единицу.

8.2.1. Ввод (чтение) одиночного бита

Самый простой способ чтения отдельного бита порта основан на использовании битовой команды пересылки в бит переноса **C** (см. п. 7.10.6):

```
Mov C, mPin.
```

После этого принятое значение можно переслать в другие биты, например, флажковой зоны ОЗУ:

Mov mFlagReady, C

или в именованные биты спецрегистров, например, в один из битов аккумулятора

Mov Acc.5, C.

8.2.2. Чтение и анализ бита

В принципе после чтения бита в перенос **C** можно воспользоваться командами анализа этого признака (**JC/JNC**). Но проще вообще обойтись без пересылки, используя условные команды перехода по состоянию бита порта (см. п. 7.11.4)

JB mPin, mMetka ; Переход, если бит порта установлен.

или

JNB mPin, mMetka ; Переход, если бит порта не установлен.

В этих командах операция анализа и условного перехода совмещена с операцией чтения бита.

В качестве примера рассмотрим фрагмент программы для анализа состояния перекидного ключа (рис. 8.2), определяющего один из трех возможных режимов работы программы и перехода на необходимую метку:

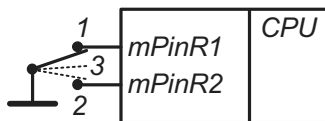


Рис. 8.2

JNB mPinR1, mRejim1 ; Если mPinR1=0, то переход на метку mRejim1,
JNB mPinR2, mRejim2 ; если mPinR2=0, то на метку mRejim2,
Jmp mRejim3 ; иначе на метку mRejim3

Условия, закодированные в мнемокодах команд (**NB** – «нет единичного бита»), проверяются последовательно в двух первых командах и при обнаружении активного нулевого состояния перекидного ключа осуществляется переход на соответствующую метку режима. Если перекидной ключ находится в среднем положении, то выполняется переход на метку третьего режима работы.

Вопросы, связанные с дребезгом контактов ключей, даны в п. 10.1.4.

Иногда подобный программный опрос состояния бита, проводимый время от времени, может запоздать. Поэтому при необходимости быстрого (аварийного) реагирования на состояние бита применяется механизм прерываний (см. §8.4)

8.2.3. Чтение групп битов из разных портов

В некоторых случаях битовая информация может быть расположена в произвольном порядке и даже в разных портах (что может быть обусловлено, например, разводкой печатной платы).

Предположим, что необходимо копировать три произвольных бита портов, именованных, как *mPin0*, *mPin1*, *mPin2*, в три младших бита аккумулятора. Будем пользоваться для копирования битовыми командами пересылки с участием бита переноса (см. п. 7.10.6):

```
Mov C, mPin0
Mov Acc.0, C
Mov C, mPin1
Mov Acc.1, C
Mov C, mPin2
Mov Acc.2, C
```

Пересылку из переноса в биты аккумулятора при объединении последовательных битов можно производить и операцией сдвига влево или вправо в зависимости от нужного порядка (см. п. 7.10.4).

```
Mov C, mPin2
RLC A
Mov C, mPin1
RLC A
Mov C, mPin0
RLC A
```

Указанные операции чтения-сдвига можно производить в цикле, например, при преобразовании последовательного формата данных в параллельный (см. далее).

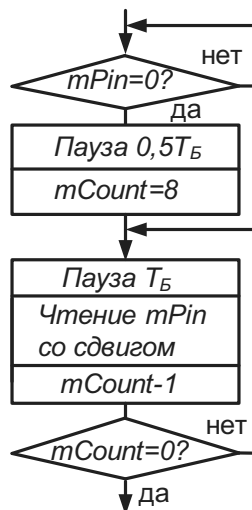


Рис. 8.3

8.2.4. Чтение последовательного формата битов

В некоторых случаях информация поступает по одному выводу порта последовательно во времени (см. п. 8.1.4). И нужно читать такой последовательный поток битов, преобразуя его в параллельный формат. Обычно для этой цели используется приемник последовательного порта (см. §8.8). Но в некоторых случаях он бывает занят другим связным процессом и приходится указанное преобразование реализовывать программным путем.

Программа (см. рис. 8.3) должна ожидать стартовый перепад, периодически опрашивая вывод порта *mPin*, затем сделать паузу, выйдя на середину периода передачи T_5 (см. рис. 8.1) и в цикле принять в аккумулятор последовательные информационные биты.

Данная процедура является иллюстративной, реальные подпрограммы используют механизм и выводы внешнего прерывания (см. §8.4), и поэтому текст реальной программы будет несколько отличаться от приведенной ниже, например отсутствием команды ожидания стартового перепада, на который реагирует система прерываний:

```

mReceiv:
    JB mPinRxD, mReceiv    ; Ждем стартовый бит.
    Call mDelay05Tb        ; Подстраиваемся под середину периода.
    Mov mCount, #8         ; Количество информационных битов.
mR1:
    ; Начало цикла.
    Call mDelayTb          ; Пауза, определяющая период приема.
    Mov C, mPinRxD         ; Чтение очередного бита
    RLC A                  ;          со сдвигом в аккумулятор.
    DJNZ mCount, mR1       ; Все биты приняты?
    Mov mBufR, A           ; Сохранение принятого байта.
    Ret

```

§8.3. Ввод-вывод информации байтовыми командами

Некоторые спецрегистры и внутренние переменные ОВМ x51 не имеют пронумерованных (именованных) битов, входящих в битовое адресное пространство от 0 до 255. В таких случаях говорят, что эти регистры не доступны побитно. Кроме того отметим, что некоторые ОВМ вообще не имеют специальных битовых команд. Поэтому в перечисленных случаях приходится для обработки битов использовать байтовые логические команды, в которых операции над битами проводятся одновременно в 8 разрядах.

Как и ранее, ниже основное внимание уделено операциям с портами, но все рассмотренные действия применимы и для других прямоадресуемых переменных.

8.3.1. Вывод (запись) байтовых констант и переменных

При сбросе ОВМ все биты портов устанавливаются в единичное исходное состояние (ориентируются на ввод). Для задания иной

комбинации битов в порт записывают требуемую константу, например:

Mov P1, #mByteConst.

Но по ходу программы может понадобиться изменение содержания портов в соответствии с некоторой вычисляемой байтовой переменной. В этом случае используют команды пересылки (копирования), в которых первый операнд (приемник информации) должен быть портом, а второй (источник информации) может быть регистром, прямо- или косвенноадресуемой ячейкой ОЗУ. Например:

Mov P1, A
Mov P3, R7
Mov P2, mByteVar
Mov P0, @R0

Однако во многих случаях требуется изменить не весь байт, а только часть его или даже отдельный бит.

8.3.2. Установка групп битов байтовыми командами

основана на операции логического «ИЛИ» байтовой переменной и константы, называемой маской. Для этого в маске на месте устанавливаемого бита нужно записать «1», а на месте сохраняемого – «0». Например, после выполнения команды

ORL P1, #00001111b

все биты младшей тетрады порта становятся единичными, а биты старшей тетрады, “замаскированные нулем”, останутся без изменения.

Примечание. Чтобы использовать символическое имя бита для образования константной маски, можно воспользоваться способностью ассемблера выполнять определенные операции над заданными числами, например, поразрядного сдвига единицы влево в зависимости от положения бита в байте.

Например, для задания константы с одной установленной единицей можно использовать выражение

(1 .SHL. mNomerBit),

где в качестве имени (номера) бита используется обобщенное имя *mNomerBit* со значением от 0 до 7 (см. п. 8.1.1).

Подобная возможность образования константной маски есть и в языке Си (см. п. 9.6.2), где сдвиг влево обозначается двумя знаками “меньше”, например $(1 \ll mNomerBit)$.

Если имя бита кодируется весом разряда (см. п. 8.1.1), то никакого дополнительного обрамления битовой константы не требуется.

*В качестве имени битов в подобных выражениях **нельзя** использовать имена битов, пронумерованных от 0 до 225, которые могут использоваться только в битовых командах. К ним относятся стандартные имена битов спецрегистров и имена, определенные в битовом сегменте ОЗУ.*

С учетом примечания для установки единицы в одном из битов можно использовать команду

ORL P1, #(1 .SHL. mNomerBit)

или

ORL P1, #mVesBit ;.

При операциях с несколькими битами символические маски образуют при помощи внутренней функции ассемблера – логического «ИЛИ» (**.OR.**), которая объединяет несколько единичных масок:

ORL P1, #((1 .SHL. mNomerBit1) .OR. (1 .SHL. mNomerBit2))

или

ORL P1, #(mVesBit1 .OR. mVesBit2).

Такая запись является несколько громоздкой по сравнению с числовой записью константы, но не нуждается в дополнительных комментариях и не требует модификации при изменении положения бита в байте, которое сказывается только при именовании бита. Для упрощения записи сложной символической маски ее можно определить в начале программы как именованную константу:

mMaska2 EQU (1 .SHL. mNomerBit1) .OR. (1 .SHL. mNomerBit2)

или

mMaska2 EQU mVesBit1 .OR. mVesBit2

8.3.3. Сброс групп битов байтовыми командами

осуществляют при помощи команды логического умножения «И» с константной маской. Только биты в маске должны быть *инверсными*, то есть ноль на месте сбрасываемого разряда, а единица – на месте сохраняемого.

Например, после выполнения команды

AnL P1, #00000111b

пять старших битов обнулятся, а три младших будут неизменными.

При символической записи инверсной константной маски используют ассемблерную логическую функцию инвертирования (**.NOT.**). Для примера запишем обнуления двух битов ранее определенной маски:

AnL P1, #.NOT. mMaska2

8.3.4. Инвертирование групп битов байтовыми командами

осуществляют при помощи команды «исключающего ИЛИ» с использованием *прямой* маски-константы («1» – на месте бита, который должен быть инвертирован, а «0» – сохранен). Например, команда

XRL P1, #(1 .SHL. mNomerBit)

производит инвертирование только одного бита *mNomerBit*.

8.3.5. Ввод (чтение) байтовой информации

При чтении всех битов порта можно пользоваться командами пересылки (копирования), в которых второй операнд (источник информации) должен быть портом, а первый (приемник информации) может быть регистром, прямо- или косвенноадресуемой ячейкой ОЗУ. Например:

Mov A, P1	; A = P1
Mov R7, P3	; R7 = P3
Mov mVar, P2	; mVar = P2
Mov @R0, P0	; @R0 = P0

8.3.6. Чтение групп битов байтовыми командами

При чтении нескольких битов можно пользоваться байтовой командой чтения всех битов порта и последующим обнулением незначащих битов при помощи логической операции «И» и *прямой* константной маски (см. п. 8.3.3), например:

Mov A, P1
AnL A, #((1 .SHL. mNomerBit1).OR.(1 .SHL. mNomerBit2))

Подобная последовательность байтовых команд широко используется в тех ОВМ, которые не имеют специальных битовых команд.

§8.4. Управление прерываниями

8.4.1. Общие сведения

При поступлении запроса на прерывание завершается выполнение текущей команды программы, в стеке сохраняется адрес следующей команды. Затем в программный счетчик заносится адрес особой ячейки в ПЗУ, выделенный данному источнику прерывания (см. п. 7.3.2). После чего из адресуемой ячейки извлекается код записанной там команды перехода к прерывающей процедуре (например, **Jmp mProclnt**). Эта процедура должна заканчиваться командой **Retf**, которая не только возвращает из стека адрес прерванной программы, но и может сбрасывать некоторые флажки запросов.

Биты запросов на прерывание можно анализировать и без привлечения механизма прерываний (см. п.п. 8.7.4, 8.10.1 и 8.10.3).

Управление системой прерываний обеспечиваются *побитно доступными* регистрами разрешения прерывания **IE (Interrupt Enable)** и приоритета **IP (Interrupt Priority)**. Биты в этих регистрах имеют стандартные имена и могут использоваться в программе без указания места размещения.

8.4.2. Регистр разрешения прерываний IE

Из восьми битов в регистре используется шесть, из них один бит **EA (Enable All)** является общим, его сброс запрещает все прерывания. Остальные пять описаны ниже.

Два бита **EX0, EX1 (Enable eXternal)** – разрешают прерывания от внешних сигналов на выводах **Int0 (P3.2)** и **Int1 (P3.3)**.

Два бита **ET0, ET1 (Enable Timer)** – разрешают прерывания по переполнению таймеров **T0** и **T1**.

Бит **ES (Enable Serial)** – разрешает прерывание от последовательного порта.

Перечисленные биты устанавливают и сбрасывают программно, то есть прерывания от любого источника могут быть независимо разрешены или запрещены, например:

SetB EX0	; Разрешить внешнее прерывание 0.
Clr EX1	; Запретить внешнее прерывание 1.
Clr EA	; Запретить все прерывания.

8.4.3. Регистр управления приоритетами IP

Каждому из пяти источнику прерываний в регистре **IP** соответствуют свои биты приоритета, стандартные имена которых похожи на имена битов разрешения прерываний. Только первая буква **E** в именах заменена на букву **P** (**Priority**) – **PX0**, **PX1**, **PT0**, **PT1**, **PS**.

Программист, пользуясь именами битов, может любому источнику присвоить один из уровней приоритета «0» или «1»:

SetB PT0	; Высокий приоритет таймера 0.
Clr PT1	; Низкий приоритет таймера 1.

Источник с высоким приоритетом может прервать программу обслуживания источника с более низким приоритетом. Если приоритет запроса одинаков или ниже, то прерывания нет, пока не закончится программа обслуживания.

*Если одновременно приходят запросы одного уровня приоритета, то первым обслуживается прерывание, имеющее меньший адрес особой ячейки (см. п. 7.3.2). То есть высший приоритет в подобной ситуации имеет внешнее прерывание **Int0**, а самый низший – прерывание последовательного порта.*

8.4.4. Управление типом внешних прерываний

Есть два типа внешнего прерывания – по нулевому уровню сигнала и по отрицательному перепаду.

В первом случае запрос на прерывание возникает, когда внешний сигнал **Int0** на выводе **P3.2** или **Int1 (P3.3)** становится *нулевым*, и после выполнения процедуры обслуживания прерывания внешняя аппаратура должна убрать этот ноль, чтобы не было повторного прерывания.

Более удобен на практике второй способ, когда прерывание вызывается при *отрицательном перепаде* внешнего сигнала. В этом случае запрос на прерывание фиксируется внутри процессора и его можно сбросить без внешней аппаратуры.

Биты управления типом внешних прерываний **IT0**, **IT1 (Interrupt Type)** упакованы вместе с таймерными битами в регистре **TCON**. Но про место размещения битов вспоминают редко, потому что регистр побитно доступен, биты именованы и их имена используются без упоминания имени спецрегистра.

Таким образом, тип внешнего прерывания задается программистом по простому правилу: если прерывание вызывается нулем, то бит типа должен быть равен нулю. Если перепадом, то – единице:

SetB IT0

; по отрицательному перепаду.

Clr IT0

; по нулевому уровню.

8.4.5. Запросы прерываний

Флажки запросов прерываний также поименованы и могут использоваться без упоминания *побитно доступных регистров*, где они размещены (четыре в регистре **TCON** и два в регистре **SCON**).

IE1, IE0 (Interrupt Edge) – биты запросов от *перепадов* внешних прерываний, которые при **IT** = 1 фиксируются при помощи внутренних триггеров. Эти биты *автоматически* устанавливаются при перепаде внешнего сигнала и сбрасываются после перехода на прерывающую программу.

TF0, TF1 (Timer overflow Flag) – флажки переполнения таймеров (17-ые разряды таймеров). Устанавливаются автоматически, сбрасываются программистом или аппаратурой системы прерываний.

TI (Transmitter Interrupt), RI (Receiver Interrupt) – запросы от передатчика и приемника последовательного порта, которые используют *общую особую ячейку* (вектор прерывания) в памяти команд.

Все рассмотренные флажки запросов можно проверять и программно, не привлекая механизм прерываний, например, используя условную команду перехода по состоянию бита.

JB RI, mPriem ; Если **RI** = 1, то обслуживается приемник!

При помощи такого анализа флажка готовности приемника программа определяет, какое из устройств последовательного порта (приемник или передатчик) нуждается в обслуживании (см. п. 8.10.2 и 8.10.5).

§8.5. Пример программы для внешнего прерывания

8.5.1. Организация внешнего прерывания

В программе нужно оформить три составляющие: задание вектора прерывания в особой ячейке ПЗУ, инициализация системы прерывания и собственно процедуру обслуживания.

Задание вектора внешнего прерывания осуществляется в рамках общего заполнения начальной зоны ПЗУ (см. п. 7.12.4).

При *инициализации* внешнего прерывания, которое производят в начале или по ходу программы, необходимо:

- установить тип прерывания (по перепаду или уровню);
- установить бит разрешения **EX0** или **EX1**;
- установить общее разрешение **EA**.

Например:

```

SetB ITO           ; Прерывание по перепаду
SetB EX0          ; Разрешить внешнее прерывание 0
SetB EA           ; Разрешить все разрешенные прерывания

```

При оформлении текста прерывающей процедуры нужно помнить два момента. *Первый* – процедура должна завершаться специальной командой возврата из прерывания **Retl**. *Второй* – модифицируемые процедурой прерывания регистры общего назначения (и аккумулятор) должны быть сохранены в стеке в начале процедуры и восстановлены из стека в конце процедуры с соблюдением обратной очередности при извлечении. Главное при этом, не забыть про регистр признаков **PSW**, который изменяется неявно для программиста.

8.5.2. Пример процедуры внешнего прерывания

Пусть наша подпрограмма должна считать количество импульсов на входе **Int0** (**P3.2**).

Примечание. Внешние импульсы можно считать и при помощи таймера (см. п. 8.6.2).

Для счета в программе можно использовать команду **инкремента**, но в семействе **x51** эта команда не устанавливает признака переноса и поэтому нельзя контролировать переполнение или организовать многобайтовый счет. Будем использовать команду сложения аккумулятора с константой, равной единице.

```

mProcInt0:
Push PSW          ; Сохраняем модифицируемые
Push Acc          ; регистры в стеке
Mov A, mFreq      ; Инкрементируем
Add A, #1        ; младший
Mov mFreq, A      ; байт
Mov A, mFreq+1    ; Добавляем к старшему байту
ADDC A, #0        ; перенос,
Mov mFreq+1, A    ; если он есть...
Pop Acc           ; Извлекаем регистры
Pop PSW          ; в обратном порядке
Retl

```

8.5.3. Использование процедуры прерывания

Предположим, что счет внешних импульсов нам нужен для измерения частоты сигнала, которая численно равна количеству импульсов на интервале в 1 секунду (см. п. 13.4.1). Пусть односекундный интервал отмеряет таймер, работающий в режиме прерывания по переполнению (см. п. 8.7.5), и его прерывающая программа устанавливает в начале каждого секундного интервала флаг *mFlag1s*.

Таким образом, будут независимо работать два процесса, счет импульсов и счет времени. И в главном цикле программы необходимо организовать их взаимодействие, разрешая и запрещая счет внешних импульсов по прерыванию:

```

mMetka1:
    Mov mFreq, #0           ; Обнуляем двухбайтовую переменную
    Mov mFreq+1, #0         ; для вычисления результата.
    JNB mFlag1s, mMetka1    ; Ждем флажок от таймера
    SetB EX0                ; Разрешаем счет импульсов и
    Clr mFlag1s             ; очищаем флажок
mMetka2:
    JNB mFlag1s, mMetka2    ; Ждем окончания 1 секунды.
    Clr EX0                 ; Запрещаем счет импульсов и
    Clr mFlag1s             ; очищаем флажок.
    Call mFreqToBufIndik    ; Результат – на индикацию.

```

§ 8.6. Регистры и возможности таймеров

Таймеров-счетчиков в стандартной конфигурации ОВМ x51 два, для программиста они представлены (см. п. 7.4.6) двумя 16-разрядными регистрами данных и двумя 8-разрядными регистрами, режима TMOD и управления TCON.

8.6.1. Регистр управления таймерами TCON

Этот регистр, несмотря на таймерное имя, обслуживает и контроллер прерываний, биты которого (*IT1*, *IT0*, *IE1*, *IE0*) для упрощения упакованы в младшей тетраде **TCON** (см. п. 8.4.4 и 8.4.5). Но про это обычно не вспоминают, поскольку регистр доступен побитно и месторасположение битов в битовых командах не указывается. А байтовые операции с регистром **TCON** использовать нет смысла.

Всего таймерных битов в регистре **TCON** четыре, два для пуска и два для индикации переполнения.

TR1, TR0 (Timer Run) – биты управления запуском таймеров. Устанавливаются и сбрасываются программно, например:

SetB TR0 ; Разрешить счет таймера 0
Clr TR1 ; Остановить счет таймера 1

TF1, TF0 (Timer overflow Flag) – флаги переполнения таймеров. Являются дополнительными битами – 9-ым для 8-разрядного счета и 17-ым для 16-разрядного.

Устанавливаются *автоматически* при переполнении сетки и служат в качестве сигнала для запроса прерывания по переполнению, то есть для аппаратного вызова подпрограмм обслуживания внешних устройств в моменты, определяемые по таймеру.

Сбрасываются *неявно* для программиста (аппаратно) после завершения подпрограммы прерывания командой **Retl**.

Эти флажки могут анализироваться и без привлечения механизма прерывания, например, командой:

JB TF0, mМетка

Но сбрасывать флаг переполнения **TF0** в этом случае должен уже сам программист (см. п. 8.7.4).

8.6.2. Формат регистра режима **TMOD**

Важно отметить и помнить, что регистр **TMOD** *побитовой адресации не имеет*, поэтому формат регистра (расположение битов в байте), приведенный на рис. 8.4, очень важен. Регистр имеет два набора одноименных битов для двух таймеров, упакованных в две тетрады.

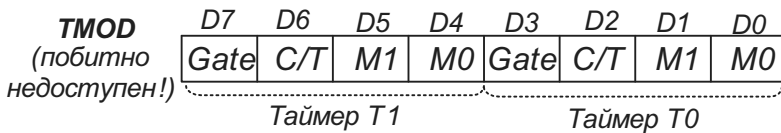


Рис. 8.4

Биты **M1** и **M0** (от **Mode**) являются главными, поскольку задают один из четырех возможных режимов работы таймеров.

- *режим 0* – 13-разрядный режим, имитирующий работу таймера в устаревшем семействе ОВМ **x48**,
- *режим 1* – 16-разрядный счет,
- *режим 2* – 8-разрядный счет с перезагрузкой,
- *режим 3* – 8-разрядный счет в обоих байтах таймера 0, таймер 1 остановлен.

Не останавливаясь на подробном описании режимов работы таймеров, которые можно найти в [17], отметим, что чаще всего используются режимы 1 и 2, особенности которых рассмотрены в п.п. 8.7.1 и 8.7.3.

Бит **C/T** (**Counter/Timer**) определяет тип входного сигнала для таймеров.

При **C/T** = 0 (режим таймера) происходит счет внутренних импульсов с частотой в 12 раз меньшей частоты кварца $F_{кв}$. Например, при $F_{кв} = 12 \text{ МГц}$, таймер инкрементируется с частотой 1 МГц, то есть увеличивается на единицу каждую *микросекунду* (далее *мкс*).

При **C/T** = 1 (режим счетчика) вход таймера подключается к выводу **P3.4 (T0)** или **P3.5 (T1)** для счета внешних импульсов, например, при измерении частоты внешнего сигнала (см. п. 10.4.5).

Бит **GATE** (ворота, клапан) служит для разрешения внешнего управления таймером и используется для измерения длительности внешнего импульса. Если **GATE** = 0, то состояние внешнего входа не влияет на работу таймера.

Если **GATE** = 1, то разрешено управление работой таймера внешним сигналом **P3.2** (для таймера 0) или **P3.3** (для таймера 1). В этом случае таймер считает, если внешний сигнал единичный и не считает, если нулевой. Таким образом, внешний сигнал превращается во внешний аналог бита пуска – **TR0** или **TR1**.

8.6.3. Инициализация (загрузка) регистра TMOD

Биты регистра **TMOD** приходится задавать байтовыми командами – засылкой байтовой константы или логическими операциями установки-сброса отдельных битов в байте (см. §8.3). Например, пусть таймер 0 должен работать в 16-битном режиме (режим с номером 1), а таймер 1 в 8-битном режиме для тактирования последовательного порта (режим с номером 2). Тогда тетрады (старшая для таймера 1 и младшая для таймера 0) будут численно равны номерам соответствующих режимов:

Mov TMOD, #00100001b

Поскольку режимы таймеров чаще всего задаются только однажды при инициализации, то с числовой записью константы приходится мириться. В тех случаях, когда режимы работы таймеров изменяют по ходу программы, более удобно пользоваться символическими именами битов, которым присвоены константы в виде веса двоичного разряда (см. п. 8.1.1). Однако на практике лучше пользоваться

двоичными комбинациями указанных битов в виде перечня режимов, например, для часто используемых режимов 1 и 2 (16-битного и 8-битного перезагружаемого):

```
mVesMode16bitForT0 EQU 1
mVesMode16bitForT1 EQU 16
mVesMode8bitReloadForT1 EQU 32
```

Тогда операция инициализации режимов таймера будет более наглядной:

```
Mov TMOD, # mVesMode16bitForT0 .OR. mVesMode8bitReloadForT1
```

Отметим, что будут установлены только необходимые биты режима M1 и M0, остальные будут обнулены, что чаще всего и нужно.

8.6.4. Загрузка регистров данных

Загрузка 16-битных регистров-счетчиков осуществляется при остановленном счете (бит **TR** = 0) и побайтно. Если загружаемый код уже вычислен и хранится в ячейках ОЗУ, то осуществляется простое копирование старшего и младшего байтов.

Например:

```
Mov TH0, mVarH
Mov TL0, mVarL
```

При загрузке 16-битной константы нужно разбить ее на составляющие байты и делает это программа ассемблера под управлением служебных знаков ">" (это аналог паскалевской функции *Hi* – старший байт слова) и "<" (это аналог *Lo* – младший байт слова):

```
Mov TH0, # > (mConst16) ; TH0 = mConst16 / 256
Mov TL0, # < (mConst16) : TL0 = остаток от деления на 256
```

8.6.5. Чтение регистров данных

При остановленном счете также используются две байтовые команды пересылки содержимого таймера:

```
Mov mVarH, TH0
Mov mVarL, TL0
```

Но иногда необходимо читать состояние счетчика на лету, без остановки счета. При этом обычно допустима ошибка в младшем разряде, так как есть вероятность чтения одного байта до инкремента таймера, а другого – после. Но опасно то, что в этот момент может произойти переполнение младшего регистра и соответственно изменение стар-

шего байта, то есть вес ошибки увеличится в 256 раз. Чтобы не допустить этого, нужно убедиться, что старший байт не изменился после чтения показаний [27]:

mMetkaRead16:

Mov A, TH0	; Чтение старшего байта
Mov mVarL, TL0	; Сохранение младшего байта
CJNE A, TH0, mMetkaRead16	; Повторное чтение и сравнение
	; старшего байта
Mov myVarH, A	; Сохранение старшего байта

Но следует иметь в виду, что в этом случае ошибка в чтении содержимого таймера увеличивается на несколько единиц из-за необходимости повторного чтения и сравнения.

В современных процессорах для безошибочного чтения двух байтов используется вспомогательный регистр хранения, который не участвует в счете и в который при чтении младшего байта копируется старший без ошибки. То есть на лету читаются по сути два байта – младший – по команде в указанный операнд, а старший – в теневой (непронумерованный) регистр хранения, из которого вторая команда чтения старшего байта прочитает верные данные [31].

§8.7. Счет времени при помощи таймеров

8.7.1. Общие сведения

При счете времени биты **С/Т** регистра **TMOD** должны быть обнулены, чтобы на вход регистров данных таймеров поступали внутренние импульсы (см. п. 8.6.2).

Ниже рассмотрены особенности наиболее часто используемых режимов 16-битного счета (код режима – 1) и 8-битного счета с автоматической перезагрузкой (код режима – 2).

В этих режимах на входы таймеров поступают импульсы с частотой в 12 раз меньше частоты кварца. При этом желательно (но не обязательно) использовать внешний кварц с частотой $F_{кв} = 12 \text{ МГц}$, тогда частота обновления таймера будет равна 1 МГц , то есть таймер будет считать *микросекунды*. Такая круглая и точная величина дискретности упрощает организацию временных пауз, задержек и системных цифровых часов.

Режим счета должен быть предварительно задан в битах **M1, M0** регистра **TMOD** (см. п. 8.6.2). Для конкретности предположим, что тай-

мер **T0** должен работать в режиме 1, а таймер **T1** – в режиме 2. В этом случае инициализация обоих таймеров осуществляется командой (см. п. 8.6.3):

Mov TMOD, # mVesMode16bitForT0 .OR. mVesMode8bitReloadForT1

При этом устанавливается необходимая комбинация битов режима, а все остальные биты, включая С/Т, будут обнулены.

8.7.2. 16-битный счет времени

рассмотрим на примере таймера **T0**. В режиме 1 регистры данных **TH0** и **TL0** таймера включены последовательно, и их двухбайтовое содержимое будет увеличиваться на единицу каждую микросекунду, то есть таймер будет выполнять роль счетчика микросекунд.

Содержимое этого счетчика, в принципе, можно читать с остановом счета или на лету. Но программа не будет успевать правильно читать и использовать полученный код, потому что время выполнения только одной команды программы равно или даже больше времени обновления таймера. Поэтому программе желательно работать с большими интервалами времени, например, в 1 миллисекунду и более.

Рассмотрим особенности формирования миллисекундных интервалов времени при помощи таймера. Для этого обратимся к рис. 8.5, на котором изображена числовая ось для возможных состояний таймера.

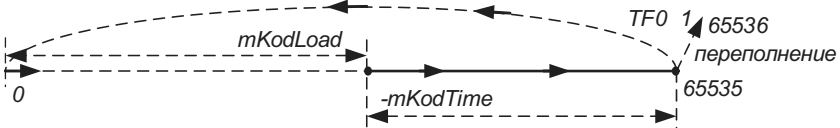


Рис. 8.5

При каждом внутреннем импульсе текущее состояние таймера увеличивается на единицу и приближается к максимально возможному, когда все разряды установлены в единицы, и равному $2^N - 1$, где N – используемая разрядность таймера-счетчика. Для 8 разрядов этот максимум равен 255, а для 16 – 65535.

Следующий импульс при максимальном коде (все единицы) приводит к переполнению таймера, все разряды его обнуляются, и устанавливается бит переполнения **TF0**, изменение которого является для процессора своеобразным «моментом истины», так как бит служит запросом для системы прерываний и его также можно анализировать программой.

Если использовать таймер в режиме свободного счета, когда происходит инкремент от 0 до максимума, то будет сформирован пропорциональный интервал времени – 256 микросекунд для 8-разрядного счета, и 65536 микросекунд – для 16-разрядного. Чтобы получить иные (более «круглые») значения, нужно после переполнения загрузить определенное, ненулевое значение кода в таймер.

Поскольку процессор обнаруживает изменения состояния таймера только в момент переполнения, необходимо вычислять начальный код загрузки таймера *назад по числовой оси* от момента установления бита **TF0**, вычитая число *mKodTime*, пропорциональное требуемому времени в микросекундах. Учитывая, что значимость бита переполнения равна 2^N , величину кода загрузки *mKodLoad* можно получить из выражения

$$mKodLoad = 2^N - mKodTime. \quad (8.1)$$

Но значение 2^N бита переполнения выходит за рамки разрядной сетки кодов для таймера и поэтому его обычно не учитывают, записывая выражение для кода загрузки в упрощенном виде:

$$mKodLoad = -mKodTime. \quad (8.2)$$

Если код времени задается в виде константы, то указанное преобразование отрицательного числа в обратный двоичный код загрузки можно произвести неявно с помощью средств ассемблера:

Mov TH0, # > (-mKodTime)	; Старший байт кода загрузки.
Mov TL0, # < (-mKodTime)	; Младший байт.
SetB TR0	; Пуск счета.

Если код времени вычисляется по ходу программы, то преобразование переменной в отрицательный код загрузки можно осуществить двумя способами. При первом способе от нуля отнимают саму двухбайтовую переменную, а при втором – инвертируют переменную и добавляют к ней единицу. Причем очень часто для простоты вычислений во втором способе ограничиваются только инвертированием, поскольку одна единица большой погрешности во временной интервал не вносит.

Поскольку двухбайтовое вычитание оказывается сложнее простого инвертирования, в большинстве практических программ используют упрощенный второй способ, например, в виде:

XRL mVarL, #255	; Инвертирование всех битов
XRL mVarH, #255 ;	для вычисления кода загрузки

и последующей пересылки в регистры таймера (см. п. 8.6.4).

Для повышения точности отсчета времени необходимо несколько корректировать код загрузки, учитывая запаздывание программы при анализе бита переполнения или при вызове процедуры обработки прерывания и загрузке начального кода.

8.7.3. 8-битный счет времени с перезагрузкой

Чтобы программа не тратила время на перезагрузку таймера при отсчете сравнительно малых интервалов времени, можно использовать 8-битный режим с автоматической перезагрузкой, который кодируется в регистре режима **TMOD** номером 2.

В этом режиме счет идет только в младшем регистре, а в старшем, постоянно хранится код загрузки. Поэтому при инициализации нужно только однажды записать константный код загрузки, отсчитанный назад по оси времени, например:

Mov TH1, #(-mByteTime)

или загрузить в старший регистр таймера вычисленную и проинвертированную переменную (см. п. 8.7.2).

При переполнении младшего регистра-счетчика устанавливается флаг переполнения, который используется программой или системой прерывания, и происходит *автоматическая загрузка* кода из старшего регистра.

Поэтому процесс счета времени после инициализации происходит без участия программы, что освобождает процессор от рутинной работы.

Кроме того, режим перезагружающегося счета таймера **T1** оказывается очень полезным для *тактирования последовательного порта*, поскольку позволяет изменять частоту приема-передачи в достаточно широких пределах (см. п. 8.9.2).

8.7.4. Отсчет времени без использования прерывания

Довольно часто в программах для синхронизации с внешними процессами используется функция задержки вычислений, подобная функции *Delay* в языке Паскаль. Такую процедуру можно реализовать, используя таймер, как генератор паузы длительностью 1 мс. Пусть код задержки со значениями от 1 до 255 предварительно задан в байтовой переменной *mKodDelay*. Нижеприведенная процедура будет отсчитывать в цикле (см. п. 7.11.5) при помощи таймера **T0**, работающего в режиме 16-битного счета, столько миллисекунд, сколько единиц было в переменной *mKodDelay*:

mKod1ms EQU 1000

mDelay:

Mov TH0, #>(-mKod1ms) ; Загрузка таймера на отсчет

Mov TL0, #<(-mKod1ms) ; времени 1 мс.

SetB TR0 ; Пуск таймера.

mD1:

JNB TF0, mD1 ; Ожидание переполнения (окончания 1 мс)

Clr TR0 ; Останов таймера

Clr TF0 ; и сброс бита переполнения.

DJNZ mKodDelay, mDelay ; Цикл закончен?

Ret ; Да, пауза завершена.

Отметим, что в подобных процедурах процессор не делает полезной работы, поэтому при отсчете больших интервалов времени и организации, например, цифровых часов (см. п. 10.4.7) желательно использовать механизм прерываний.

8.7.5. Отсчет времени с использованием прерываний

Прерывания по переполнению таймеров могут быть разрешены или запрещены командами установки-сброса соответствующих битов разрешения прерывания (см. п. 8.4.2). Ниже в качестве примера приведены эти команды для таймера **T0**:

SetB ET0 ; Разрешить прерывание по переполнению таймера T0

Clr ET0 ; Запретить прерывание по таймеру T0

Если прерывание запрещено, то процессор не реагирует на переполнение таймера и, как обычно, работает по программе.

Если прерывание разрешено, то при переполнении процессор приостанавливает работу по программе, сохраняет следующий адрес счетчика **PC** в стеке и переходит к обработке прерывающей программы. То есть записывает в программный счетчик номер особой ячейки ПЗУ, выделенной для таймера (см. п. 7.3.2), и извлекает оттуда команду перехода на прерывающую программу.

*Заметим, что в этом случае флаг переполнения **TF0** таймера обнуляется автоматически системой прерываний.*

Предположим, что ранее в таблице векторов прерываний была записана команда перехода на процедуру, обслуживающую переполнение таймера **T0**, а также задан режим 1 для таймера и разрешено прерывание по переполнению.

Оформление прерывающей процедуры для таймера в целом стандартное (см. п. 8.5.1) – модифицируемые регистры общего пользова-

ния должны быть в начале процедуры сохранены в стеке и в заключение восстановлены в обратном порядке. Последней командой должна быть команда возврата из прерывания **Retl**. Добавляется только фрагмент перезагрузки таймера и необходимые команды обработки данных:

```

mProcT0:
  Push PSW                ; Сохраняем модифицируемые
  Push Acc                 ;                      регистры в стеке
  Clr TR0                  ; Останавливаем таймер
  Mov TH0, #>(-mTik)       ; Перезагружаем таймер на новый
  Mov TL0, #<(-mTik)       ;                      отсчет времени
  SetB TR0                 ; Пуск таймера
  ; Здесь можно производить опрос датчиков, индикацию данных и пр.
  Inc mCount10ms          ; Считаем интервалы по 10 мс
  Mov A, mCount10ms       ; Проверяем прошла ли
  CJNZ A, #100, mExit      ;                      1 секунда?
  SetB mFlag1s            ; Да, устанавливаем флажок 1 с.
  ; Здесь можно организовать счет минут и часов (см. п. 10.4.7)...
mExit:
  Pop Acc                  ; Извлекаем регистры
  Pop PSW                 ;                      в обратном порядке
  Retl

```

В приведенной процедуре прерывания по переполнению таймера **T0** для примера организован счет десятимикросекундных интервалов времени, которые условно можно назвать «тиками» системных часов. Эти интервалы позволяют синхронизировать моменты обслуживания внутренней и внешней периферии ОВМ, опрос аналоговых и цифровых датчиков и другие действия.

Кроме того, системные «тики» позволяют организовать программные цифровые часы со счетом реального времени, например, в дополнительных переменных типа *mCount1s* и других. Все эти переменные можно анализировать в основной программе и выполнять необходимые действия в заданные моменты времени.

В некоторых случаях счет времени в процедуре прерывания не ведут, а ограничиваются установкой флажков-индикаторов в определенные моменты времени, например, *mFlag1s*, как бы делают временные отметки. После использования в основной программе эти флажки сбрасываются последним процессом, который их анализировал (см. п. 8.5.3).

При помощи таймеров можно не только формировать временные интервалы, но и измерять длительности и частоту внешних импульсов (см. п. 10.4.5).

§8.8. Управление последовательным портом

Асинхронный последовательный порт в настоящее время входит в состав большинства ОВМ и является универсальным устройством для обмена данными, что отражено в английском сокращении UART (Universal Asynchronous Receiver-Transmitter). Однако ОВМ x51 вдобавок имеет и синхронный режим, поэтому ее последовательный порт обычно обозначают, как USART.

8.8.1. Общие сведения

Последовательный порт представляет собой два отдельных устройства: приемник и передатчик (*Receiver-Transmitter*), выполненные на основе регистров сдвига. Приемник принимает последовательные во времени биты (см. п. 12.6.3) и преобразует их в параллельный формат в регистре данных. Передатчик выполняет обратное действие (см. п. 12.6.1) – данные параллельной формы передает в последовательном виде.

Последовательный порт работает в асинхронном режиме (рис. 8.1), то есть передатчик добавляет к последовательно передаваемым во времени информационным битам нулевой стартовый и единичный стоповый биты, а приемник это добавление автоматически удаляет, оставляя в параллельном формате только код данных. Отметим также, что информационные биты в последовательной посылке начинаются с младшего.

Входом приемника является вывод **RxD** – бит параллельного порта **P3.0**, выходом передатчика – **TxD** (бит **P3.1**). Если порт не используется, то указанные биты порта **P3** можно задействовать для ввода или вывода цифровых сигналов.

Частота работы приемника и передатчика одинакова и задается, как будет показано, при инициализации (см. §8.9). Для удвоения скорости передачи можно использовать бит удвоения в регистре управления мощностью **PCON** (см. п. 8.8.5).

Регистры данных приемника и передатчика имеют общее имя **SBUF (Serial BUFFer)**, но физически это разные регистры. Из регистра приемника можно только читать данные, а в регистр передатчика только записывать данные.

Кроме регистров данных **SBUF** последовательный порт имеет регистр управления **SCON (Serial CONtrol)**, биты которого описаны в п. 8.8.2 и 8.8.3.

Приемник значение каждого принимаемого бита определяет тремя выборками за период длительности бита и вычисляет «голосованием» два из трех (см. п. 11.10.5). В более современных кристаллах [31] число выборок может быть увеличено до 15, что позволяет эффективно выделять стартовый перепад и информационные биты в условиях помех. После приема последнего (стопового) бита приемник устанавливает флаг готовности (см. п. 8.8.2), после чего принятые данные можно читать в аккумулятор или любую переменную в ОЗУ:

Mov mBufR, SBUF ; Копирование кода в буфер для приема.

Передать байт можно следующей командой:

Mov SBUF, mBufT ; Передача кода из буфера или другой переменной.

Сразу же после выполнения этой команды байт копируется в регистр сдвига (см. п. 12.6.1), и на последовательном выходе **TxD** появляется стартовый бит. После опустошения регистра сдвига (окончания стопового бита) устанавливается бит готовности передатчика (см. п.п. 8.4.5 и 8.8.2).

8.8.2. Назначение битов регистра управления SCON

Регистр **SCON** побитно доступен, поэтому в программе можно использовать стандартные имена битов без указания места их размещения. Ниже приведен перечень этих битов.

REN (Receive Enable) – бит разрешения приема. Устанавливается программой при инициализации. Но иногда, например, чтобы не слушать свою передачу в общем канале связи, приемник отключают, программно сбрасывая этот бит.

RI (Receive Interrupt) – бит запроса прерывания от приемника, означающий, что принят новый байт. Устанавливается приемником аппаратно (автоматически), должен быть сброшен программистом (см. п. 8.10.1).

TI (Transmit Interrupt) – бит запроса прерывания от передатчика. Устанавливается автоматически, когда передатчик завершает последовательное преобразование (закончился стоп-бит), и можно записать следующие данные. Сбрасывается этот бит программистом (см. п. 8.10.3).

Бит **TB8 (Transmit Bit 8)** является необязательным 9 разрядом (**D8**) регистра данных **SBUF** передатчика. При необходимости он может включаться в асинхронную последовательную посылку в качестве

старшего бита данных, бита четности или бита указателя типа послы-ки (команда/данные, см. п. 8.8.4).

Передаваемый бит **TB8** в программе может быть установлен:

SetB TB8

или сброшен:

Clr TB8

или принять вычисленное значение из бита переноса:

Mov TB8, C.

Бит **RB8 (Receive Bit 8)** является аналогичным необязательным 9 разрядом (**D8**) регистра данных **SBUF** приемника.

Значение бита **RB8** определяется аппаратно при приеме 9-битного последовательного кода и может копироваться командой в бит переноса:

Mov C, RB8

или анализироваться программой

JB RB8, mLabel.

В специальном многоточечном режиме обмена (см. п. 8.8.4) бит **RB8** может служить индикатором принятой от главного устройства команды.

Биты **SM0, SM1, SM2 (Serial Mode)** – задают режимы работы последовательного порта. Их назначение и использование приведено далее.

8.8.3. Управление режимами последовательного порта

осуществляется тремя битами режима **SM0, SM1, SM2**, каждый из которых имеет свою основную функцию.

Например, при установке бита **SM1** последовательный порт подключается для тактирования к таймеру 1, и скорость можно изменять в широких пределах. Если бит **SM1** сброшен, то скорость будет фиксированная (см. п. 8.9.1).

С помощью бита **SM0** можно установить 8- или 9-битный режимы асинхронной послыки.

Таким образом, с помощью битов **SM0** и **SM1** можно закодировать четыре режима работы последовательного порта.

В режиме 0 (**SM0=0, SM1=0**) реализуется *синхронный* режим приема-передачи байтов. Передатчик работает в этом режиме, как гене-

ратор тактовых импульсов, то есть на выходе **TxD** при каждой передаче или приеме байта генерируются 8 инверсных синхроимпульсов с частотой $F_{KB}/12$.

Приемник работает (по входу данных **RxD**) подобно тактируемому регистру сдвига без стартового и стоповых битов, и может передавать или принимать синхронные данные. Передача байта начинается после записи его в регистр **SBUF**, а прием инициализируется сбросом бита готовности приема **RI** (при условии **REN** = 1). Временные диаграммы приема-передачи приведены на рис. 8.6.

*Следует отметить, что синхросигнал на выводе **TxD** – инверсный, то есть данные можно читать при активном нулевом уровне.*

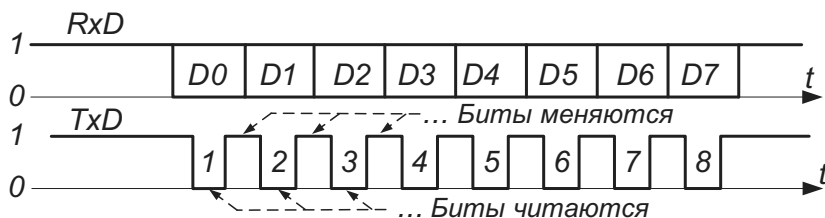


Рис. 8.6

Режим 1 (**SM0**=0, **SM1**=1) реализует 8-битную асинхронную посылку с частотой, задаваемой таймером 1 (см. п. 8.9.2).

Режим 2 (**SM0**=1, **SM1**=0) использует 9-битный асинхронный формат с фиксированной частотой $F_{KB}/64$ или $F_{KB}/32$ в зависимости от бита **SMOD** в регистре **PCON** (см. п. 8.8.5).

Режим 3 (**SM0**=1, **SM1**=1) реализует 9-битную асинхронную посылку (см. п. 8.8.4) с частотой, задаваемой таймером 1.

*На практике из-за обратной значимости битов **SM0** и **SM1** при кодировке режимов могут возникать трудности. Поэтому для облегчения использования этих битов рекомендуется переименовать их, чтобы новое имя напоминало о выполняемой битом функции. Например, бит **SM1**, управляющий подключением таймера **T1** для тактирования последовательного порта **PS** можно именовать, как*

mBitOnT1forPS_SM1 REG SM1

Причем во избежание путаницы в составном имени бита можно оставить исходное обозначение бита через знак подчеркивания.

*Аналогично бит режима **SM0** можно переименовать в соответствии с его выполняемой функцией подключения разряда **D8** в состав асинхронной посылки, например:*

mBitOnD8forPS_SM0 REG SM0

Тогда команды задания режимов последовательного порта можно будет записывать без комментариев:

SetB mBitOnT1forPS_SM1***Clr mBitOnD8forPS_SM0***

Бит **SM2** управляет особым режимом работы приемника – режимом для аппаратного различения в последовательном потоке кодов команд главной удаленной станции от кодов чужих данных (см. далее).

Если бит **SM2** не установлен, то принимаются все последовательные послышки, которые могут анализироваться и использоваться программой. Подобный режим работы является основным при обмене между двумя станциями, образующими так называемую двухточечную сетевую конфигурацию.

8.8.4. Режим аппаратного различения 9 бита при приеме

При организации удаленного взаимодействия нескольких микропроцессорных сетевых устройств (многоточечная сетевая конфигурация) обычно используется централизованный доступ к общему каналу связи под управлением главного устройства. Подчиненные только принимают команды от главного и выполняют их. При этом возникает задача различения команд и данных, для чего может быть использован 9 бит асинхронной послышки, который в команде должен быть установлен.

Поэтому, если принят бит **RB8=1**, то остальные 8 битов должны восприниматься программой как код команды от главного устройства, содержащий биты адреса подчиненного. Все подчиненные должны принимать команду и анализировать адресную часть послышки. Но только одно устройство, адрес которого указан в команде, должно выполнить ее. Остальные устройства игнорируют принятую команду.

Для упрощения реализации подобного режима обмена данными анализ бита **RB8** можно сделать не программным, а аппаратным (автоматическим). В этом режиме **SM2=1** и принимаются только команды, что сказывается только на установке бита готовности **RI** приемника.

Если принята асинхронная послышка с битом **RB8=0**, означающим, что приняты сетевые данные, то бит готовности **RI** не устанавливается, и принятые данные остаются поэтому невидимыми для программы.

Если принят бит **RB8=1**, это означает, что остальные биты являются кодом команды главного устройства, или отмечают начало сетево-

го пакета. Поэтому бит готовности **RI** устанавливается, и программы всех подчиненных устройств, реагируя на это, начинают анализ кода команды.

Но только одно устройство обнаруживает свой адрес в команде и начинает выполнять ее, остальные продолжают работу в пассивном режиме.

Устройство, получившее команду (например, принять), сбрасывает **SM2**, принимает – передаёт свой байт или еще несколько в обычном режиме, а затем, устанавливая бит **SM2**, переходит в режим ожидания следующей команды.

*Чтобы облегчить использование бита **SM2** в программах, его можно переименовать в соответствии с его выполняемой функцией подключения командного режима, например:*

mBitCmdMode_SM2 REG SM2

и осуществлять манипуляции с указанным битом обычными битовыми командами.

8.8.5. Бит последовательного порта в регистре управления мощностью **PCON**

Один из битов, управляющих работой последовательного порта, (бит **SMOD** – **Serial MODe**) не поместился в регистре **SCON**, и разработчикам ОВМ x51 пришлось упаковать его в совершенно другом регистре **PCON** (**Power CONtrol**), управляющем режимами энергосбережения.

Бит **SMOD** называют битом удвоения частоты тактирования последовательного порта, потому что при **SMOD** = 1 частота приема-передачи умножается на два.

Примечание: бит **SMOD** не работает в режиме 0.

Регистр **PCON** доступен только целым байтом, поэтому порядок битов в байте, который приведен на рис. 8.7, очень важен при программировании.

PCON	D7	D6	D5	D4	D3	D2	D1	D0
(побитно недоступен!)	SMOD	-	-	-	GF1	GF0	PD	IDL

Рис. 8.7

В частности, необходимый нам бит **SMOD** расположен в старшем разряде регистра **PCON**, то есть при кодировке бита **весом** его разряда имя бита **SMOD** определяется числом 2⁷:

mVesBitSMOD EQU 128

а команда для установки этого бита должна быть записана в виде

Mov PCON, #mVesBitSMOD

*Чтобы облегчить использование бита удвоения в программах, его можно именовать не **SMOD**, а в соответствии с его выполняемой функцией удвоения скорости, например:*

mVesBitBaudRateMul2 EQU 128

Другие биты регистра **PCON**, такие как бит понижения мощности **PD** (**Power Down**) и бит холостого хода **IDL** (**IDLe**) – в данном пособии не рассматриваются. Особенности их использования для экономии потребляемой мощности и отключения части внутренних узлов можно найти в литературе [27].

§8.9. Инициализация последовательного порта

8.9.1. Выбор режима тактирования последовательного порта

происходит с помощью бита **SM1**, который управляет подключением таймера **T1** к последовательному порту для тактирования. Два значения бита **SM1** определяют два режима тактирования, с фиксированной частотой и изменяемой при помощи таймера **T1**.

Если бит **SM1** = 0 (что соответствует режимам 0 и 2 последовательного порта), то скорость приема-передачи будет фиксированная и изменить ее программно практически нельзя. Например, в синхронном режиме 0 (**SM1** = 0, **SM0** = 0) тактовая частота зависит от частоты внешнего кварца:

$$F_{\text{Б}} = F_{\text{КВ}} / 12. \quad (8.3)$$

а период $T_{\text{Б}}$ поэтому неизменен и равен длительности машинного цикла.

В асинхронном режиме 2 (**SM0** = 1, **SM1** = 0), частота приема-передачи имеет только два возможных фиксированных значения.

Если бит **SMOD** = 0 (см. п. 8.8.5), то

$$F_{\text{Б}} = F_{\text{КВ}} / 32. \quad (8.4)$$

Если бит **SMOD** = 1, то

$$F_{\text{Б}} = F_{\text{КВ}} / 64. \quad (8.5)$$

Поэтому для задания скорости последовательного порта в режимах 0 и 2 необходимо просто установить номер режима битами **SM1** и **SM0**. Причем в режиме 2 можно удвоить скорость, установив бит **SMOD**.

Если бит **SM1** = 1 (*mBitOnT1forPS_SM1* = 1), то последовательный порт подключается для тактирования к таймеру **T1**, и скорость обмена можно изменять в широких пределах. Поэтому для задания скорости последовательного порта нужно не только установить номер режима битами **SM1** и **SM0**, но и задать режим работы таймера **T1**, и выбрать код перезагрузки, определяющий период тактирования $T_{\text{Б}}$ (см. далее).

8.9.2. Использование таймера T1 для тактирования последовательного порта

Возможно только в режиме с автоматической перезагрузкой начального кода (см. п. 8.7.3). В этом режиме после инициализации таймер работает без участия программы, как программируемый делитель частоты (см. п. 13.2.3), определяя интервал приема-передачи битов $T_{\text{Б}}$ (рис. 8.8).

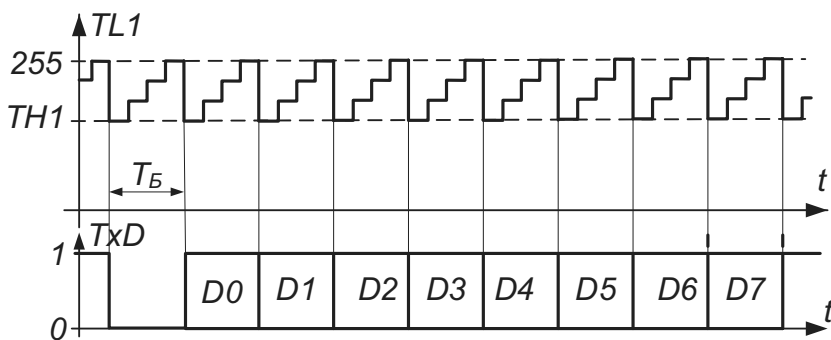


Рис. 8.8

Нетрудно видеть, что этот интервал легко программировать, изменяя начальный код загрузки в регистре **TH1**. Причем частоту приема-передачи ($F_{\text{Б}} = 1/T_{\text{Б}}$) следует выбирать из стандартного ряда частот, используемых в сетевых устройствах для совместимости (например, 1200 Гц, 2400 Гц и т. д.).

В фирменной документации и литературе (например, [17]) обычно приводятся формулы и таблицы значений кода начальной загрузки для регистра **TH1** в зависимости от частоты внешнего кварца и требуемой стандартной скорости.

Для упрощения использования в программах эти начальные коды загрузки можно записать в виде именованных констант для разных скоростей и частот кварца. Например, ниже приведены константы загрузки для кварца с частотой 11,059 МГц и стандартного ряда частот от 1200 Гц до 9600 Гц:

```
mTH1for9600kv11059 EQU 0FDh
mTH1for4800kv11059 EQU 0FAh
mTH1for2400kv11059 EQU 0F4h
mTH1for1200kv11059 EQU 0E8h
```

При инициализации таймера одну из этих констант загружают в регистр **TH1** (см. п. 8.9.4). Заметим, что выбранную частоту можно удвоить, если установить бит **SMOD** = 1.

8.9.3. Погрешность задания стандартной скорости

В некоторых случаях приходится применять кварцевые резонаторы с частотами, которые при делении таймером **T1** не дают точного значения одной из стандартных частот приема/передачи. Поэтому при этом возникает погрешность и опасность рассинхронизации удаленных сетевых устройств.

Из-за несовпадения частот приемника и удаленного передатчика происходит смещение момента приема бита (рис. 8.9), который должен быть близок к середине периода. Но приемник остается работоспособным, если смещение момента опроса последнего бита послышки (стоп-бита) не превысит половины периода, то есть 50% от T_b .

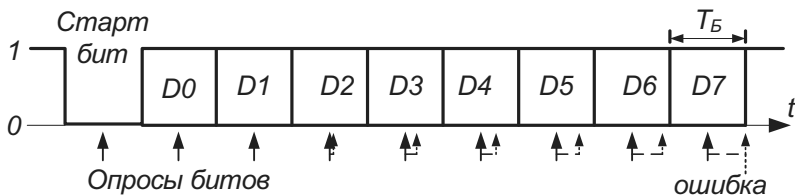


Рис. 8.9

Учитывая, что смещение момента опроса накапливается за время приема 10 периодов (старт-бит, 8 информационных и стоп-бит), можно найти относительную погрешность несовпадения для одного периода, которая будет в 10 раз меньше, то есть 5%. Для большей надежности последнюю цифру обычно уменьшают примерно вдвое.

Отсюда следует вывод о том, что на практике допустимо несовпадение частот приемника и передатчика в несколько процентов.

$$F_{SP} = \frac{2^{SMOD}}{32} \cdot \frac{F_{KB}}{12} \cdot (256 - TH1) \quad (8.6)$$

Для оценки данной погрешности вначале находят частоту работы последовательного порта по формуле [17]:

Затем находят относительную погрешность несовпадения с одной из частот F_B стандартного ряда:

$$\delta = (F_{SP} - F_B) / F_B \quad (8.7)$$

и сравнивают ее с допустимой.

8.9.4. Пример инициализации последовательного порта

Чтобы обеспечить правильное функционирование порта, нужно в самом начале программы выполнить следующие действия:

- задать режим таймера **T1**,
- в регистр **TH1** занести константу, определяющую скорость приема-передачи,
- задать режим работы последовательного порта,
- разрешить прерывания.

Для наглядности процесс инициализации оформляют обычно в виде именованных макроса или подпрограммы, как показано ниже:

myInitPS9600:

```

Mov TMOD, #(mVesMode16bitForT0 .OR. mVesMode8bitReloadForT1)
    ; Таймер T0 – 16-битный счет, T1 – 8-битный с перезагрузкой.
Mov TH1, #mTH1for9600kv11059 ; Задаем частоту 9600 Гц.
SetB TR1 ; и запускаем таймер T1.
Clr mBitOnD8forPS_SM0 ; Задаем 8 битный режим
SetB mBitOnT1forPS_SM1 ; и тактирование от T1.
    ; Mov PCon, #mVesBitBaudRateMul2 ; Удвоение частоты, если нужно.
SetB REN ; Разрешение приема.
SetB ES ; Разрешение прерывания от PS.
SetB EA ; Общее разрешение прерываний.
Ret

```

В основной программе эту процедуру вызывают командой **Call myInitPS9600**, почти не требующей комментариев.

§8.10. Процедуры для последовательного порта

8.10.1. Программное ожидание посылки

Если прерывание от последовательного порта не используется (запрещено), то перед чтением регистра приемника нужно в цикле ждать принятия новой порции данных, анализируя флажок **RI**. После этого прочитать данные и сбросить флажок во избежание повторного чтения. Ниже приведена процедура подобного программного приема:

```
mPriemW:
  JNB RI, mPriemW      ; Ждем готовность приемника
  Mov A, SBuf          ; Прием кода
  Clr RI               ; Сброс флажка
  Ret
```

Распознавание и обработка ошибок обычно осуществляется в программах более высокого уровня, управляющих процессом обмена в целом.

8.10.2. Прием по прерыванию

Последовательному порту для прерываний выделена, как известно, особая ячейка **23h** в начале ПЗУ (см. п. 7.3.2), в которую при оформлении программы нужно записать команду перехода на прерывающую процедуру:

```
.CODE
....
.ORG 23h      ; адрес прерывания для последовательного порта
  Jmp mIntPS
....
```

Однако этот адрес **23h** особой ячейки разделяют два отдельных устройства: передатчик и приемник. И вследствие этого процедура прерывания должна начинаться с анализа флажков запросов, чтобы определить, какое именно устройство последовательного порта нуждается в обслуживании. Поэтому в нижеприведенной подпрограмме можно выделить начальную команду-селектор и два фрагмента, для передачи и приема.

Передающая часть будет рассмотрена ниже. Здесь же мы ограничимся сбросом бита запроса передатчика, чтобы избежать повторного вызова прерывания, и установкой флажка-дублера готовности пере-

датчика (см. описание его назначения в п. 8.10.3) для программной передачи.

```

mIntPS1:
  JB RI, mPriemInt      ; Обслуживается только приемник!
  Clr TI                ; Сбрасываем, чтобы не было повторного вызова.
  ; SetB mFlagTReady    ; Можно установить свой индикатор,
                        ; дублирующий бит готовности TI.
  RetI

```

Фрагмент приема по своей структуре сходен с обычной процедурой прерывания: сохранение в стеке модифицируемых РОН, собственно прием (и обработка, если нужно), сброс флажка **RI** для исключения повторного вызова и в заключение восстановление РОН из стека.

```

mPriemInt:
  Push PSW              ; Сохранить регистры
  Push A
  Clr RI                ; Сброс запроса.
  ; Mov mBufR+2, mBufR+1 ; Байтовый сдвиг в буфере,
  ; Mov mBufR+1, mBufR   ; если принимается пакет,
  ; Inc mCountPaket      ; и инкремент счетчика.
  Mov mBufR, SBuf       ; Прием байта в буфер.
  ; Здесь первичная обработка, если нужна...
  Pop A                 ; Восстановить
  Pop PSW               ; регистры.
  RetI

```

8.10.3. Передача байта без использования прерываний

Передача обычно инициируется программой. Если есть данные, то вызывается процедура передачи, которая начинается с ожидания готовности передатчика.

Определенная тонкость есть при использовании приема по прерыванию, поскольку в этом случае бит готовности **TI** сбрасывается процедурой *mIntPS1* после селектирования, как видно из предыдущей подпрограммы. И в этом случае рекомендуется ввести и использовать свой флаг-дублер бита готовности, который устанавливается после сброса **TI**. Можно также запретить прерывание по последовательному порту на время передачи.

```

mSend:
  JNB TI, mSend         ; Ожидание готовности по TI
  ; JNB mFlagTReady, mSend ; или по флажку-дублеру
  Mov SBuf, mBufT       ; Передача кода

```


Clr TI	; Сброс TI или
; Clr mFlagTReady	; mFlagTReady
Ret	

Использование механизма прерывания при передаче пакета приведено в п. 8.10.5.

8.10.4. Передача с ожиданием окончания посылки

В некоторых случаях нужно дожидаться конца посылки, например, при использовании общего канала связи (типа *RS-485*) в полудуплексном режиме необходимо освободить канал, сбрасывая, например, бит *mOnTransmitter* управления передатчиком:

<i>mSendWait:</i>	
JNB TI, mSendWait	; Ждем готовности передатчика
Clr TI	; и очищаем флажок
SetB mOnTransmitter	; Подключаем передатчик к каналу
Mov SBuf, mBufT	; Передаем код из буфера.
<i>mSW:</i>	
JNB TI, mSW	; Ждем окончания посылки
Clr mOnTransmitter	; Отключаем передатчик.
Ret	

8.10.5. Передача пакета данных по прерыванию

При передаче нескольких байтов программа должна дожидаться, как в п. 8.10.4 окончания передачи данных бит за битом, то есть процессор в это время не выполняет полезной работы. Чтобы повысить производительность используют механизм прерываний, когда программа только загружает байт для передачи, не контролируя окончания.

Пусть нужно передать пакет, длина которого задана в переменной *mCountPak*, а байты подготовлены в буфере *mBufT*. Для инициализации передачи нужно задать ненулевое значение *mCountPak*.

<i>mInitPS2:</i>	
Push PSW	; Сохранение
Push A	; модифицируемых
Push R0	; регистров
JB RI, mPriemInt	; Прием или передача? Прием уже описан
Clr TI	; Обслуживаем передатчик
Mov A, mCountPak	; Все байты
JNZ mSendByte	; передали?
<i>mExitIntPS:</i>	; Да, выходим

Pop R0	<i>; Восстановление</i>
Pop A	<i>; модифицируемых</i>
Pop PSW	<i>; регистров</i>
Retl	<i>; Окончание процедуры</i>
<i>mSendByte:</i>	
Add A, #mBufT	<i>; Вычисляем адрес</i>
Mov R0, A	<i>; байта для передачи</i>
Mov SBuf, @R0	<i>; Передаем байт</i>
Dec mCountPak	<i>; Счет оставшихся байтов</i>
Jump mExitIntPS	

В этой программе окончание каждой посылки не отслеживается, и процессор только временами *прерывается* на засылку очередного байта в регистр данных передатчика.

Отметим, что передача идет в сторону уменьшения адреса буфера. Можно реализовать программу и по-другому, например, использовать две переменных – счетчик байтов для организации цикла, который декрементируется, и указатель передаваемого байта, который для вычисления адреса инкрементируется.

9 ОСОБЕННОСТИ ПРОГРАММИРОВАНИЯ ОВМ x51 НА ЯЗЫКЕ Си

Язык Си позволяет не только относительно быстро создавать для ОВМ достаточно сложные программы, но и существенно облегчает переход на другие процессорные семейства. Но всегда следует помнить, что ресурсы ОВМ в большинстве случаев ограничены, поэтому нужно минимизировать свои программы, использовать по возможности компактные байтовые переменные и ограничивать вложенные вызовы функций, чтобы уменьшить риск переполнения маленького стека.

Предполагается, что читатель знаком с базовыми средствами языка Си и его основными операторами [21] с учетом правил компилятора SDCC (ANSI стандарт с открытыми источниками, [32]).

§9.1. Директивы `#include` и `#define`

9.1.1. Общие сведения

Каждый язык программирования, как и монастырь, имеет свой устав. Программа на Си [21] несколько похожа по своей структуре на ассемблерную программу (см. п. 7.12.1). Например, она имеет фрагменты, где описываются константы и внешние сигналы, резервируются переменные, определяются используемые процедуры и главная часть программы пользователя. Но сначала отметим только некоторые общие особенности записи программ.

Основой для реализации алгоритмов являются операторы. В Си, как и в других языках программирования, имеется условный оператор **if**, операторы цикла **for**, **while**, выбора **switch** и другие.

Имена пользовательских констант, переменных и процедур, используемых в примерах, будут начинаться с префикса «т», чтобы отличать их от стандартных имен спецрегистров, операторов и директив, которые вдобавок, для отличия, набраны жирным шрифтом. При использовании описанных программных

фрагментов в своих программах имена с префиксом “m” нужно изменять в соответствии с личными предпочтениями.

После операторов в большинстве случаев ставится точка с запятой. Однако нужно помнить, что точка с запятой **не ставится** после директив препроцессора (начинающихся со знака #) и после фигурных закрывающих скобок.

Как всегда неотъемлемой принадлежностью программы являются комментарии, которые могут быть двух типов – многострочные (обрамленные служебными знаками /* и */) и однострочные (начинающиеся со знаков //). Рекомендуется [21] применять в программе однострочные комментарии, а многострочные использовать только в громоздком заголовке или при отладке для временного отключения фрагментов программы.

Язык Си с его особенностями и стандартными функциями, скрытыми во *включаемых* файлах, реализован для множества разных процессоров. Перечень таких файлов, используемых в программе, обязательно указывается в начале при помощи директивы **#include** с именем файла в *двойных кавычках* или *угловых скобках*:

```
#include "mNameFile"    // Файл находится в текущем каталоге.  
#include <mNameFile>    // Файл берется из каталога компилятора.
```

Содержимое включаемых файлов при предобработке дополняет текст самой программы.

Полезно помнить некоторые особенности записи директив:

- решетка # должна находиться в первой позиции строки;
- пробел после # допустим;
- точка с запятой после директивы не ставится;
- если файл не находится в текущем или стандартном каталоге, то должен быть указан полный путь к включаемому файлу.

9.1.2. Особенности включаемых файлов

Чаще всех используются **заголовочные** файлы, которые имеют имена с расширением **.h** (от **Header** – заголовок). Ведь в них описаны основные стандартные функции и определения языка. Они содержат прототипы функций, определения для функциональных констант, структуры, используемые компилятором, и другую полезную информацию.

Например, математические функции бывают доступны при использовании файла *math.h*, а функции системы прерываний – *signal.h* или *interrupt.h*.

При программировании бывает полезно ознакомиться с перечнем включаемых файлов для конкретного компилятора.

Отметим, что в заголовочных файлах действительно приведены только заголовки (объявления) функций, а реализация их скрыта в библиотечных файлах различного формата. Но в случае открытых исходных программ можно найти и тексты необходимых функций на Си, ознакомление с которыми очень полезно для начинающих программистов.

Кроме того, к заголовочным относятся файлы с описанием ресурсов процессора. Например, для процессора **AT89C51** фирмы **ATMEL** используется директива

#include <at89x51.h>

с файлом **at89x51.h**, в котором находятся основные определения стандартных имен для ОВМ **x51** и их порядковые номера в памяти данных. Для иных процессоров нужно применять соответствующие заголовочные файлы.

Кроме заголовочных в текст программы можно включать и любые другие файлы с процедурами и функциями, написанными на Си. Это позволяет многократно использовать свои (и чужие) наработки и значительно облегчает процесс программирования.

9.1.3. Директива препроцессора **#define**

записывается в следующем виде:

#define mName mTextC // Без точки с запятой.

Препроцессор, встретив в программе имя **mName**, заменяет его на текст **mTextC**, написанный по правилам языка Си. Поэтому эту директиву иногда называют директивой «подмены» текста. Кроме того, учитывая, что директива по смыслу похожа на ассемблерный макрос, то текст с именем **mName** иногда и называют макросом.

Эта директива служит для именования частей программы, а также для введения привычных имен для некоторых стандартных идентификаторов, определенных во включаемом файле. Но следует иметь в виду, что *ключевые слова самого языка Си переопределять нельзя.*

Для большей заметности и явного выделения текста макроса рекомендуется обрамлять его фигурными скобками:

#define mFazaOff { mFaza1 = 1; mFaza2 = 1; mFaza3 = 1; }

Макрос может также получать параметры:

#define mOn(mVar) { mFazaOff; mVar = 0; }

Если текст после директивы не умещается на одной строке, то в конце строки ставят знак \ (обратный слеш).

Вызов и выполнение операторов небольшого макроса кодируется компактнее и происходит *быстрее*, чем вызов процедуры с таким же набором операций. Но нужно помнить, что препроцессор проводит только подстановку, *не проверяя текста* на ошибки, которые затем будет очень трудно обнаружить.

*Хрестоматийный пример записи неправильного макроса [21] для вычисления квадрата **#define** $\text{sqr}(x) \ x*x$. При его использовании в случае $y = \text{sqr}(x + 1)$ препроцессор производит следующую подстановку:*

$y = x + 1 * x + 1 = 2*x + 1$, вместо нужной $(x + 1) * (x + 1)$.

*Поэтому этот макрос нужно определять следующим образом: **#define** $\text{sqr}(x) (x)*(x)$. Отсюда следует простой вывод – скобок не жалеть...*

§9.2. Ресурсы ОВМ х51 для языка Си

9.2.1. Общие сведения

Программы на Си для ОВМ х51 и «большого» процессора по большому счету почти не отличаются. Но особенности системы команд и скромные возможности ОВМ х51, ориентированной на работу с объектами в режиме реального времени, конечно же, учитываются компилятором. Но эти различия в большинстве случаев скрыты от программиста, который организует выполнения заданного алгоритма типовыми средствами языка высокого уровня. Однако, во избежание ошибок при написании программы, обязательно нужно учитывать имеющиеся в наличии ресурсы ОВМ.

В первую очередь это касается нестандартных типов данных (см. §9.5) и внутренней структуры ОВМ, которая представляется программисту в виде набора запоминающих ячеек, служащих для реализации заданного алгоритма.

*В компиляторе SDCC для семейства х51 определены три подобных набора (модели) **small**, **medium** и **large**. Чаще всего по умолчанию используется модель памяти **small**, в которой все переменные без явного указания класса памяти размещаются во внутренней памяти данных. В других моделях подобные переменные располагаются компилятором во внешнем ОЗУ.*

В базовом варианте ОВМ х51 минимальный набор (**small**) внутренних запоминающих ячеек (см. п. 7.5.1) состоит (см. рис. 9.1) из

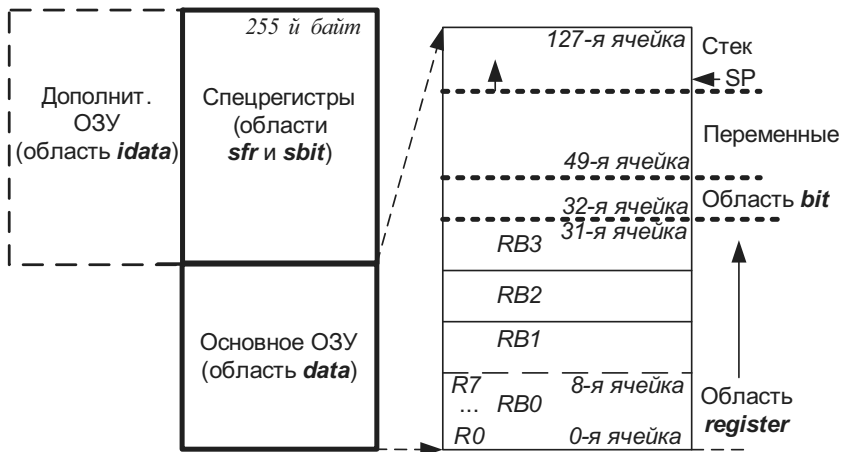


Рис. 9.1

основного ОЗУ (область **data**) и спецрегистров (область **sfr** – *Special Function Registers*). Сами коды команд (см. п. 7.3.2) размещаются в постоянной памяти команд (область **code**), в которой программист может разместить и константы (см. п. 9.3.4).

Спецрегистры, размещенные в области **sfr**, управляют работой ОВМ и ее аппаратуры. Каждый из спецрегистров имеет стандартное имя, определяемое во включаемом файле, и может использоваться в программах наравне с переменными пользователя (см. п. 9.2.4).

В *основном ОЗУ* (область **data**) размещаются оперативные регистры общего назначения (РОН), пользовательские переменные программы и стек, где обычно хранятся адреса возврата из процедур и данные компилятора.

Дополнительное ОЗУ в семействе ОВМ x51 размещено параллельно адресному пространству спецрегистров. Чтобы обеспечить разделение данных, в ассемблерных командах при работе со спецрегистрами используется прямая адресация, тогда как при обращении к дополнительному ОЗУ – косвенная. В программах на Си эта особенность дополнительного ОЗУ подчеркивается в его именовании **idata** первой буквой *i* (от *indirect* – косвенный).

9.2.2. Указание места размещения переменных

Учитывая изложенное, перед объявлением типа данных и имени переменной можно (если это нужно программисту) дополнительно

указать место размещения переменной (тип памяти), учитывая особенности структуры семейства **x51**:

register – в регистре общего назначения;

data – во внутреннем основном ОЗУ;

idata – во внутреннем дополнительном ОЗУ;

xdata – во внешнем ОЗУ;

code – в памяти команд (для констант);

extern – определяется в другом программном модуле.

Отметим, что есть и другие, не рассматриваемые здесь разновидности типов памяти, связанные с аппаратными особенностями ОВМ (страничная, энергонезависимая память и т. д.).

Примеры объявления переменных с различным типом памяти:

```
data unsigned char mCount10ms;
```

```
extern int mParol;
```

```
idata unsigned char mBufer[100];
```

9.2.3. Размещение локальных переменных

По умолчанию компиляторы Си используют так называемые динамические переменные, обозначаемые классом **auto**, которые не инициализируются и удаляются после завершения процедуры.

Эти переменные должны размещаться в стеке, но стек в ОВМ **x51** невелик, поэтому в большинстве случаев компиляторы размещают локальные переменные в РОН (**R0..R7**), выбирая при необходимости один из банков **RB0..RB3**.

Отметим также, что регистры и стек ОВМ в явном виде программистами используются сравнительно редко, тогда как компилятор интенсивно использует и регистры, и стек для своих целей. Однако, при поиске ошибок программисту иногда придется самому анализировать степень переполнения стека, которое не всегда обнаруживается компилятором особенно при рекурсивном вызове функций.

*Если необходимо увеличить размер стека, то его размещают во внешней памяти (когда она есть), пользуясь при запуске компилятора опцией **xstack**.*

9.2.4. Работа со спецрегистрами

Стандартные имена регистров аппаратуры, как отмечалось, объявляются в заголовочном файле применяемой разновидности ОВМ и могут использоваться в программе наравне с переменными пользователя. Например:


```

P1 = 0;
P2 = mAdrH;
mKodFromPC = SBuf;
TL0 = (char)(mKodInt);

```

Полезно также помнить, что при сбросе почти все спецрегистры обнуляются, за исключением регистров портов, которые устанавливаются в единичное состояние (ориентируются на ввод, см. п. 7.2.2).

*Нужно отметить, что в типовом случае в области **sfr** определены только битовые и байтовые стандартные переменные. Поэтому особого внимания требуют операции с разнородными данными, поскольку некоторые спецрегистры состоят из именованных битов, а некоторые имеют двухбайтовый формат. То есть важным моментом программ является преобразование типов данных (см. §9.6). Например, 16-разрядные регистры таймеров нуждаются в побайтовом обслуживании (см. п. 10.5.5). В некоторых компиляторах могут использоваться спецрегистры большей размерности (**sfr16** и **sfr32**). В связи с этим бывает полезно ознакомиться с включаемым файлом используемого процессора.*

9.2.5. Ассемблерные вставки

Для получения компактного бинарного кода или для быстрого выполнения некоторых действий вместо операторов языка Си можно использовать команды на языке ассемблера для используемого процессора. Для этого ассемблерные команды обрамляют ключевыми словами **_asm** и **_endasm**, например:

```

_asm
  mov A, mVar    ; В данном случае
  cpl a          ;      используются ассемблерные
  mov mVar, A    ;      правила записи.
_endasm;

```

Отметим, что ассемблерные команды допустимо писать одной строкой, что иногда удобно, например, при объявлении однострочного макроса.

Более подробно взаимосвязь ассемблера и операторов языка Си в данном пособии не рассматривается. При необходимости сведения об этом можно найти в описании компилятора [32].

Рекомендуется изучать результирующий ассемблерный листинг компилятора для разных программных вариантов и выбирать оптимальный, то есть задавать вопросы компилятору и правильно интерпретировать его ответы.

Но следует иметь в виду, что наличие ассемблерного текста ухудшает наглядность программы и усложняет ее переносимость на другие процессорные семейства. Поэтому подобные вставки следует использовать только в критических случаях (например, для повышения быстродействия) применяя язык Си для общего управления выполняемыми процессами.

§9.3. Объявления переменных и констант

9.3.1. Объявления переменных

Для 8-разрядного процессора x51 основным и естественным типом данных является **unsigned char**, используемый для хранения однобайтового, беззнакового и целого кода со значением от 0 до 255. Пример объявления:

```
unsigned char mCount10ms, mTemp;
```

*Отметим, что однотипные переменные объявляются через запятую. Точка с запятой в конце объявления обязательна. При необходимости можно определить знаковое однобайтовое число **signed char**, имеющее значения от -128 до +127.*

Для чисел с большим диапазоном значений можно объявлять двухбайтовые переменные **unsigned int** (значения от 0 до 65535) и **signed int** (от -32768 до 32767).

Объявление можно совместить с присваиванием начального значения – инициализацией переменной, например:

```
unsigned char mCount = 5, mTemp;
```

*Для своего удобства программист может переименовать двухсловный спецификатор **unsigned char** однословным (например, **mByte**) при помощи оператора создания синонима*

```
typedef unsigned char mByte;
```

И далее в тексте использовать при объявлении переменных свое укороченное имя типа данных:

```
mByte mCount10ms;
```

*Такое же переименование типа можно сделать при помощи директивы **define**. Некоторые компиляторы сами определяют подобные сокращения, как встроенные, например, имеют тип **byte** или **uchar**.*

В дальнейшем для компактности будем использовать синонимы **byte** вместо **unsigned char** и **word** вместо **unsigned int**.

Стандартные имена спецрегистров (**SBUF**, **TMOD** и др.) дополнительно объявлять не нужно, они задаются компилятору из включаемого файла используемого процессора.

Примеры использования переменных:

```
mByteR = SBUF;    // Чтение последовательного приемника
SBUF = mByteT;    //      и запись в передатчик (см. п. 10.8.1).
mCountByte++;    // Инкремент переменной.
P1 = ~mBuferIndik; // Вывод в порт P1 с инверсией (см. п. 9.6.1).
```

9.3.2. Объявление не удаляемых локальных переменных

Локальные переменные для экономии памяти обычно удаляются после завершения процедуры. Чтобы при повторном вызове процедура могла использовать прежнее значение локальной переменной, ей необходимо присвоить класс **static**, описывающий переменную, которая размещается в ОЗУ наравне с глобальными переменными, *инициализируется нулем* и не удаляется после завершения процедуры.

Переменная такого типа при новом вызове возвращает сохраненное значение. Пример такого объявления:

```
static char mVarStat;
```

9.3.3. «Изменчивые» переменные типа **volatile**

В программе могут работать несколько независимых и асинхронных процессов, зависящих от *внешних* событий и управляемых контроллером прерываний. Каждый из этих процессов может неожиданно для остальных изменять значения своих переменных. Чтобы заставить компилятор каждый раз читать значение такой переменной (особенно внешних сигналов), запретив сохранение ее во временном регистре (например, при оптимизации программного фрагмента), при описании подобных «деликатных» переменных нужно использовать модификатор **volatile**. Например:

```
volatile byte mCounterMoney;
volatile sbit mPinMISO;
```

Если программа временами работает не всегда корректно, то рекомендуется вспомнить про этот модификатор.

9.3.4. Объявления констант

Неизменные величины можно ввести в программу двумя способами, при помощи директивы **define** или ключевого слова **const**. Ниже дан пример, иллюстрирующий оба способа:

```
#define mTik 10000 // Десятичное число.  
const byte mMaska = 0x0a, // Шестнадцатеричное число  
mTikH = -mTik / 256, // Результат деления – в старший байт,  
mTikL = -mTik % 256; // остаток – в младший байт.
```

Отметим, что во втором случае константы записываются в кодовой памяти и для извлечения их компилятор несколько усложняет ассемблерную программу. Поэтому одиночные константы для компактности рекомендуется объявлять директивой **#define**, потому что в этом случае константа хранится в теле ассемблерной команды.

Упорядоченные константы обычно используют в виде массивов (см. ниже).

9.3.5. Объявления многобайтовых переменных и массивов

При необходимости в программе также можно использовать переменные большего размера, например, с типом **long** (4 байта) вместе с модификаторами **unsigned** и **signed**, явно указывающими беззнаковый или знаковый тип.

*Для проведения сложных расчетов над данными в принципе можно объявлять переменные с плавающей или фиксированной запятой типа **float**, подключив соответствующий библиотечный файл. Однако при этом нужно помнить об ограниченности вычислительных ресурсов ОВМ.*

Многобайтовые последовательности данных, образующиеся, например, при обработке сетевых пакетов (см. п. 8.10.5) или визуализации данных (см. п. 11.4.4), удобно объявлять в виде массивов переменных, элементы которых *нумеруются*, начиная с нуля:

```
byte mBuferr [8]; // 8 байтов для приема с номерами от 0 до 7.
```

Элементы массива при объявлении можно инициализировать, записывая значения в *фигурных* скобках (размер в этом случае можно не указывать):

```
byte mBuferrIndik [] = {2, 0, 1, 1};
```

Инициализация обязательно требуется при объявлении массивов констант в кодовой памяти:

```
code byte mATcmd [ ] = { 'ATE0', 13};
```

Примеры использования массивов:

```
mBuferR [mCountByte++] = SBUF;
```

```
SBUF = mATcmd [2];
```

```
P1 = mBuferIndik [mCountZnak];
```

При необходимости можно объявлять и многомерные массивы (с учетом имеющейся памяти).

Во избежание ошибок при обращении к массиву обязательно следует внимательно контролировать выход номера элемента массива за объявленные границы, поскольку компилятор этого не делает.

§9.4. Объявления функций

9.4.1. Общие сведения

Кроме директив, определений и объявлений переменных и констант, программа имеет функции – *главную* (всегда именуемую **main**), *стандартные* (подставляемые, если нужно, из включаемого или библиотечного файла) и *пользовательские*.

Главную функцию **main**, которой передается управление при сбросе процессора, должна иметь любая программа, поскольку именно функция **main** реализует требуемый алгоритм на основе набора операторов и других функций.

Стандартные (библиотечные) функции, описываемые во включаемых файлах, в данном пособии не рассматриваются. При необходимости их можно найти в описаниях компилятора [32]. Поэтому ниже дано только общее описание особенностей объявления функций пользователем.

В первую очередь отметим, что объявляемая функция должна иметь имя и тело функции, состоящее из набора операторов и обрешенное фигурными скобками.

Все функции, включая **main**, могут получать и возвращать параметры. Поэтому при объявлении, кроме имени и тела функции, можно указать список получаемых аргументов и тип возвращаемых данных.

*Однако, нужно помнить, что компилятор SDCC в большинстве случаев использует для получения и возврата данных 16-разрядный указатель данных **DPTR**, то есть желательно по возможности ограничиваться двухбайтовым форматом принимаемых и возвращаемых функцией данных.*

9.4.2. Функция, не получающая и не возвращающая данные

Если функция, производя какие-то действия, не получает параметры и не возвращает ничего, то ее помечают в двух местах неопределенным типом **void**:

```
void mProc (void) {  
    ...           // В фигурных скобках записывается тело функции.  
}
```

Допустима запись без второго ключевого слова **void**:

```
void mProc () {.....}
```

Отметим повторно, что после закрывающих фигурных скобок точка с запятой не ставится.

Вызов подобной функции в программе осуществляется по ее имени (идентификатору), например:

```
mProc();           // Или mProc;
```

9.4.3. Объявление функции, получающей параметры

Тип получаемых данных описывается после имени функции в круглых скобках в виде списка имен аргументов с указанием их типа:

```
void mProc (byte mVar1, byte mVar2) {  
    ...           // Здесь записывается тело функции, в котором  
    ...           // используются перечисленные переменные.  
}
```

При вызове подобной функции в поле аргументов записывают перечень используемых переменных:

```
mProc (mX, mY);
```

Чтобы избежать ошибок, связанных с неявным преобразованием типов данных (см. §9.6), желательно следить за тем, чтобы типы подставляемых аргументов соответствовали объявленным.

9.4.4. Объявление функции, возвращающей значение

Тип возвращаемых данных объявляется перед именем функции. Кроме того, чтобы функция могла вернуть значение, в теле функции должен быть оператор **return**, содержащий результат функции, например:

```

int mFunction (byte mVarIn) {
    int mVarOut = 1;
    ... // Здесь записывается тело функции, в котором
    ... // на основе получаемых аргументов
    ... // вычисляется значение выходной величины.
    return mVarOut;
}

```

Оператор **return** может применяться и без возвращаемой переменной, например, для принудительного завершения процедуры в одной из ветвей программы.

При вызове возвращающей значение функции следует обратить внимание на тип принимающей переменной:

```
mVarInt = mFunction (mX);
```

Тип должен соответствовать типу функции, в противном случае нужно помнить о правилах преобразования типов (см. §9.6).

9.4.5. Бесконечный цикл в главной функции

Обслуживание внешних объектов происходит в большинстве случаев непрерывно (безостановочно) и главная функция при этом работает в бесконечном цикле, реагируя на сигналы от датчиков или команды от центрального устройства. Подобный бесконечный цикл реализуют, например, при помощи оператора цикла **while**, задавая условие выполнения цикла в виде константы, отличной от нуля, например:

```

void main (void) {
    mProclnit();           // Начальная инициализация.
    while (1) {             // Условие всегда выполняется.
        ...                 // Рабочий бесконечный цикл.
    }                       // Скобка while.
}                           // Скобка main.

```

Бесконечный цикл можно оформить и при помощи оператора **do-while(1);**, а также оператора **for(;;);**, в котором пропущенное условие выполнения цикла считается всегда истинным.

9.4.6. Пример записи простейшей программы

Для вывода данных из ОВМ достаточно часто используется время-импульсная форма представления информации (см. §16.3) в виде импульсного сигнала с так называемой широтно-импульсной модуляцией (ШИМ). В простейшем случае генерацию ШИМ сигнала можно производить, например, при помощи функции программной задержки (см. §9.8):

```

/* Программа генерации ШИМ сигнала с периодически
   изменяющейся длительностью импульса.*/
#include <at89x51.h>           // Файл процессора.
typedef unsigned char mByte;   // Определен спецификатор byte.
                               // Объявление внешних сигналов и констант.
#define mPinPWM P1_1          // Назначение внешнего вывода.
#define mPeriod 255           // Константа для задания периода.
                               // Объявление переменных.
mByte    mKodTi,              // Код длительности импульса
          mCountT;             // Счетчик периодов
                               // Объявление функции задержки.
void mDelay(mByte mKodD) {
    mByte ml;                  // Локальная переменная.
    for (ml=0; ml<mKodD; ml++); // mKodD пустых циклов для задержки.
}

                               // Главная программа.
void main (void) {
    mKodTi = mCountT = 1;
    while (1) {                // Бесконечный цикл.
        mPinPWM = 1;           // Установка выходного сигнала
        mDelay(mKodTi);        // Отсчет длительности импульса.
        mPinPWM = 0;           // Сброс сигнала.
        mDelay(mPeriod - mKodTi); // Отсчет паузы.
        if (mCountT== 255) mKodTi++; // Изменить длительность?
        mCountT++;             // Считаем периоды.
    }
}

```

§9.5. Адресуемые биты ОВМ x51

9.5.1. Общие сведения

Главным отличием ОВМ x51 от других процессорных семейств является наличие двух подмножеств отдельно управляемых (побитно адресуемых) битов.

Первое подмножество битов (от 0 до 127) размещено в 16 смежных байтах ОЗУ после банков РОН (см. рис. 1.1) и служит для объявления компактных булевых переменных с типом **bit**.

Второе подмножество битов, пронумерованных от 128 до 255, расположено «вразброс» в области **sfr** в несмежных спецрегистрах, является их неотъемлемой частью и объявляется как пространство специальных битов **sbit**.

При работе с указанными подмножествами компилятор использует специальные битовые команды семейства **x51** (см. п. 7.10.6), что существенно упрощает бинарный машинный код.

Следует также отметить, что биты ОВМ x51 не являются аналогами битовых полей стандартного Си, которые могут быть размещены в произвольном месте памяти данных.

9.5.2. Объявления битовых переменных

Наличие в ОВМ **x51** специальных битовых команд и массива, нумерованных от 0 до 255 битов, позволяет и в Си объявлять в программе соответствующие битовые переменные, которые не являются стандартным типом данных.

Биты в ОЗУ, помеченные ключевым словом **bit**, обычно используются в качестве переменных типа **bool**. Чтобы отличать эти биты от битов спецрегистров, рекомендуется вводить в идентификатор слово **Flag**. Ниже дан пример объявления подобных битовых переменных (флажков), которые можно при необходимости инициализировать («0» или «1»):

```
bit mFlagReady, mFlagStop = 0;
```

Биты в области спецрегистров (типа **sbit**) для управления имеваемыми битами внутренней аппаратуры обычно определены во включаемом файле для данной ОВМ и совпадают в основном со стандартными именами, используемыми ассемблером. Например, имя **TR0** используется для бита запуска таймера 0, бит **EX0** управляет разрешением внешнего прерывания 0 и так далее.

*Однако некоторые имена спецбитов могут иметь отличия в обозначении. Например, некоторые компиляторы нумеруют биты портов не через точку, а через иной разделитель, например, знак подчеркивания – **P1_1**. Поэтому во избежание ошибок желательно ознакомиться с заголовочным файлом используемой ОВМ.*

При переименовании битов портов на свои, более наглядные имена рекомендуется использовать директиву подмены, а в имя бита порта вводить слово **Pin** (штырек), чтобы не путать их с флажковыми битами в ОЗУ:

```
#define mPinPWM P1_1  
#define mPinSvetodiод P3_5
```

Некоторое усложнение имен окупается обычно меньшим числом ошибок и возможностью быстро изменить программу в соответствии с разводкой печатной платы.

Особенности операций с побитно адресуемыми битовыми переменными даны ниже. Для манипуляций с битами в целочисленных переменных, не имеющих побитовой доступности, используются логические байтовые команды (см. §9.7).

9.5.3. Операции с битовыми переменными

Наравне с другими операндами эти биты могут принимать и передавать значения с учетом своего формата («0» или «1»). Наиболее часто используют установку, сброс и копирование бита:

```
TR0 = 1;           // Запустить таймер 0.  
mFlagStop = 0;     // Булева переменная равна значению false.  
mPinSvetodiod = RI;
```

Над битами можно также проводить операции логического сложения и умножения, например:

```
mPinSvetodiod = mFlagStop | mPinImp;  
mPinSvetodiod = mFlagReady & RB8;
```

Следует отметить, что две указанные логические операции реализуются при компиляции специальными битовыми командами семейства **x51** (см. п. 7.10.6). Но для выполнения более сложных операций, особенно связанных с преобразованием типов операндов, прямых аналогов битовых команд нет, и именованные биты требуют особой осторожности, поскольку результирующий код сильно зависит от используемого компилятора и его режима оптимизации. Именно при использовании именованных битов возможны ошибки, для обнаружения которых нужно изучать листинги выходного ассемблерного кода.

Поэтому рекомендуется сложные операции с битами (особенно при сокращенной форме записи) выделять отдельной строкой и проверять их результат при помощи ассемблерного листинга или отладчика.

Например, не всегда ожидаемо применительно к подобным битам (но в соответствии со стандартом?) реализуется операция побитового отрицания **~**. Поэтому для инвертирования значения бита (аналогично ассемблерной операции **Cpl**) следует использовать более «предсказуемую» операцию логического отрицания:

```
mPinOrFlag = !mPinOrFlag;
```

Другим вариантом инвертирования битовой переменной является использование операции «исключающего ИЛИ»:

```
mPinOrFlag = mPinOrFlag ^ 1;
```

Отметим, что это выражение допустимо записывать сокращенно:

```
mPinSvetodiod ^= 1;
```

Причем служебные знаки “^=” нужно писать без пробела.

Отметим, что при использовании именованных битов в условных операторах допустимо записывать имя бита без условия, так как бит уже является булевым результатом:

```
if (mFlagReady) mPinSvetodiod = 1;    // Тестирование бита.  
while (!TF0);                        // Ждать, пока TF0 = 0;
```

Особого внимания требуют операции с участием переменных разного формата, такие как занесение значения бита в целое число (см. п. 9.6.4) и обратное преобразование (см. п. 9.6.6).

§9.6. Типовые преобразования данных

Аппаратура ОВМ и внешние объекты управляются двоичными кодами разных форматов – битовыми, байтовыми и двухбайтовыми. Поэтому в программах часто приходится осуществлять взаимные преобразования переменных средствами языка Си. Часть из них осуществляется автоматически (неявно для программиста). Другая часть преобразований иницируется программными операторами. Ниже рассмотрены наиболее распространенные способы целочисленных преобразований разнородных двоичных данных.

9.6.1. Битовые операции для целочисленных операндов

В Си имеются шесть операторов (&, |, ^, ~, >>, <<), которые можно использовать для манипулирования битами в целых числах.

Оператор & (побитовое «И») выполняет операцию логического умножения отдельных одноименных битов операндов. То есть результирующий бит будет единичен только тогда, когда одноименные биты двух операндов тоже единичны. Такая операция часто используется для *обнуления* незначащих (не учитываемых в конкретном случае) разрядов числа (см. п. 9.7.4).

Оператор | (побитовое «ИЛИ») реализует операцию логического сложения отдельных одноименных битов операндов, то есть результирующий бит будет нулевым только тогда, когда одноименные

биты операндов тоже нулевые. Эту операцию применяют для *установки* выбранных битов (см. п. 9.7.3).

Операторы & и |, которые работают с операндами побитно, следует отличать от логических операторов && и ||, которые оценивают каждый операнд целиком, равен он нулю или нет.

Например, если mVar1 = 1, а mVar2 = 4, то результат побитовой операции (mVar1 & mVar2) будет равен нулю, а логической (mVar1 && mVar2) будет единичен.

Оператор ^ (побитовое «исключающее ИЛИ») реализует сложение по модулю 2 (см. п. 11.9.1) для одноименных разрядов чисел и часто используется для инвертирования выбранных разрядов (см. п. 9.7.5).

Оператор инверсии ~ (побитовое отрицание) инвертирует все биты числа.

9.6.2. Сдвиги переменных

позволяют перемещать в нужное место биты и группы битов целочисленных переменных. Необходимость в этом возникает при работе с отдельными битами, свойственной большинству программ, обслуживающих реальные объекты. Иногда операции сдвига используют для быстрого умножения (см. п. 10.2.3) или деления на степень двойки.

Операции сдвига перемещают разряды операнда *влево* (<<) или *вправо* (>>) на число позиций, указанных в следующем операнде. Оба операнда должны быть целыми, а второй операнд обязательно должен быть положительным.

Пусть для примера $mByteX = 128$ (двоичное значение 10000000 – единица в самом старшем бите байта). Применив операцию сдвига, эту единицу можно переместить в любое место байта, например в самый младший разряд:

$mByteY = (mByteX \gg 7);$

то есть после операции $mByteY = 1$ (поделили на $2^7=128$).

В том случае, когда результат операции снова заносится в прежний операнд, для упрощения записи иногда применяют сокращенную форму операции $mByteX \gg= 7$; которую из-за не наглядности в дальнейшем использовать не будем.

Тип полученного после сдвига результата – это тип первого операнда. Поэтому биты, выдвигающиеся за разрядную сетку сдвигаемой переменной (влево или вправо), безвозвратно теряются, так как переполнение при сдвигах не контролируется.

При сдвиге влево в освобождающиеся младшие биты записываются нули независимо от типа операнда.

При сдвиге вправо беззнаковых чисел в освобождающиеся старшие разряды заносятся нули. Для знаковых чисел в освобождающиеся старшие разряды копируется бит знака (ноль для положительных чисел и единица – для отрицательных).

Приведенный выше пример сдвига иллюстрирует перемещение только одной единицы. Естественно нужно помнить, что и другие разряды числа тоже перемещаются. Например, если $mByteX = 5$ (двоичное значение 00000101), то при сдвиге влево на 5 разрядов:

$mByteY = (mByteX \ll 5);$

будем иметь двоичное значение 10100000, то есть результат будет $mByteY = 160$ (умножили на $2^5=32$).

Более подробные сведения о применении сдвигов при обработке отдельных битов приведены в п.п. 9.7.6 – 9.7.8.

9.6.3. Преобразование «коротких» типов переменных в «длинные»

В операциях присваивания над переменными, имеющими разные форматы, результат автоматически преобразуется к типу переменной, получающей это значение.

Рассмотрим «удлинение» переменной на двух примерах записи явного (с указанием нового типа переменной) и неявного (автоматического) преобразования типа **char** в значение **int**:

$mVarInt = (int) mVarChar;$
 $mVarInt = mVarChar;$

Младший байт $mVarInt$ становится равным $mVarChar$, а старший заполняется нулями или битами знака. То есть разрядная сетка увеличивается за счет добавления *незначущей* старшей части и потери информации не происходит. Поэтому в таких случаях вполне можно не указывать новый тип присваивания.

9.6.4. Преобразование бита в целое число

Такой же принцип «удлинения» за счет наращивания разрядности старшей незначущей части применим для случая, когда битовая переменная преобразуется в целое число, например, в байт:

$mVarByte = mVarBit;$

Младший бит D_0 байтовой переменной *mVarByte* принимает значение бита *mVarBit*, а остальные биты *mVarByte* заполняются нулями.

Можно обойтись без потери старших битов принимающей переменной (с заранее обнуленным разрядом принимаемого бита), если использовать операцию «ИЛИ»:

$$mVarByte = mVarByte \mid mVarBit;$$

Для приема значения бита в произвольный *обнуленный* разряд (например, с номером *mNomer*) можно использовать условный оператор вместе с побитовой операцией «ИЛИ» с константной маской (см. п. 9.7.3):

$$\text{if}(mVarBit) \ mVarByte = mVarByte \mid (1 \ll mNomer);$$

Отметим также, что кроме логического сложения при преобразовании бита, можно использовать и арифметическое сложение (см. п. 13.4.5).

9.6.5. Преобразование «длинных» типов переменных в «короткие»

При таком преобразовании (например, *mVarChar* = *mVarInt*;) разрядная сетка укорачивается за счет отбрасывания старших разрядов, то есть происходит *потеря информации*, поскольку отбрасывается часть двоичного кода.

Чтобы не забывать об этих особенностях, желательно **явно указывать** преобразование типов – перед преобразуемой переменной писать в скобках новый тип, например,

$$mVarChar = (\text{char}) \ mVarInt;$$

или для большей наглядности программы, чтобы напоминать о том, что *отбрасывается старший байт*, обнулять его операцией логического умножения (см. п. 9.7.4):

$$mVarChar = mVarInt \ \& \ 0x00ff;$$

Но при этом бинарная программа после компилятора несколько усложняется.

В тех случаях, когда потеря информации нежелательна, можно предварительно сохранить старшую часть операнда в другой переменной меньшего формата. Например, для сохранения старшего байта целого числа нужно поместить его на место младшего при помощи операции сдвига на 8 разрядов (см. п. 9.6.2):

$$mVarByteH = (\text{Char}) \ (mVarInt \gg 8);$$

9.6.6. Преобразование целого числа в бит

Для операции приравнивания бита байту (например, *mVarBit = mVarChar;*) применяется *иное правило* вычисления значения бита на основе переменной – *бит равен нулю, если переменная равна нулю. Иначе, бит равен единице.*

Используя этот булевый (*логический*) принцип, легко пересылать значение *любого* бита, если выделить этот бит, сбросив в ноль все остальные (см. п. 9.7.4). Например, в выражении

$$mVarBit = mVarChar \& (1 \ll mNomer);$$

бит *mVarBit* примет значение бита с указанным номером в байтовой переменной. Подробности операций с отдельными битами целых чисел приведены в следующем параграфе.

§9.7. Работа с отдельными битами целых чисел

9.7.1. Указание места бита в целочисленной переменной

Многие биты расположены в переменных, не доступных для специальных битовых операций семейства *x51*. В подобных случаях для манипуляций с битами применяют логические побитовые операции (см. п. 9.6.1), которые осуществляются одновременно над всеми одноименными битами целого числа.

В качестве второго операнда при этом чаще всего используется константа того же типа данных, имеющая единицу на месте значимого бита. Остальные незначимые биты должны быть невидимыми для операций или, как говорят, «замаскированы» нулем. Поэтому указанную константу для операций с битами часто называют константной маской (см. п. 8.3.2).

Для образования *битовой константной маски*, которую назовем *mMaskaBit*, используются два способа (см. п. 8.1.1).

В одном случае биту маски присваивается константное значение, равное значимости (*весу*) выбранного разряда, то есть степени двойки, например, 1, 2, 4, 8, 16 и т. д. Тогда в качестве маски можно использовать имя бита:

$$mMaskaBit = mVesBit;$$

В другом случае биты в числе нумеруются по порядку, например, в байте от 0 до 7, и в программе определяется константа с *номером* бита, например:

```
#define mNomerBit 5
```

Тогда битовую маску можно получить при помощи операции сдвига:

```
mMaskaBit = (1 << mNomerBit);
```

то есть единица вдвигается влево на место нумерованного бита. Остальные биты будут обнулены (см. п. 9.6.2). В некоторых компиляторах для этой цели бывает определен специальный макрос с именем `_BV()`:

```
#define _BV(mNom) (1 << mNom)
```

который, получая номер бита, возвращает его вес (*Bit Value*).

Использовать имена битов, определенных в битовой зоне ОЗУ (bit) или стандартные имена побитно доступных спецрегистров (sbit) в этих выражениях нельзя!

9.7.2. Образование маски для нескольких значащих битов

осуществляется при помощи операции побитового «ИЛИ»:

```
mMaskaBits = (1 << mNomerBit1) | (1 << mNomerBit2);  
mMaskaBits = mVesBit1 | mVesBit2;  
mMaskaBits = mVesBit1 | (1 << mNomerBit2);
```

Чтобы не писать подобных громоздких выражений, в начале программы рекомендуется определить маску, как макрос, например:

```
#define mMaskaBits {(1 << mNomerBit1) | (1 << mNomerBit2);}
```

Отметим также, что использование числовых (неименованных) констант внутри программы затрудняет ее чтение и снижает в дальнейшем качество сопровождения проектов.

9.7.3. Установка отдельных битов целочисленного операнда

Для установки одного бита применяется операция побитового «ИЛИ» с использованием ранее рассмотренных констант `mNomerBit`, `mVesBit`, `mMaskaBits`, имеющих единицы на месте устанавливаемых битов и нули в других разрядах:


```

mVar = mVar | (1 << mNomerBit);
mVar = mVar | mVesBit;
mVar = mVar | mMaskaBits;

```

В программах иногда используется сокращенная форма записи операции, например

```
mVar |= mMaskaBits;
```

что удобно при оформлении однострочных макросов, но для большей наглядности желательно использовать полный формат.

Отметим также, что для разнородных операндов действуют ранее рассмотренные особенности преобразования типов данных (см. п. 9.6.3 и п. 9.6.5).

В некоторых случаях установку бита оформляют при помощи макроса

```
#define mSetBit (mReg, mNomerBit) {mReg |= (1 << mNomerBit);}
```

Подобный макрос может быть определен в библиотечном или включаемом файле.

9.7.4. Сброс незначащих битов

Для сброса части битов при сохранении (выделении) значащих битов переменных используют операцию побитового «И» с *инверсной* константной маской, имеющей ноль на месте сбрасываемого бита и единицы в остальных разрядах. Такая маска получается путем инверсии (служебный знак ~) ранее рассмотренных прямых масок:

```

mVar = mVar & ~(1 << mNomerBit);
mVar = mVar & ~mVesBit;
mVar = mVar & ~mMaskaBits;

```

Иногда подобная операция бывает определена во включаемом файле, как функция *ClrBit (mReg, mNomerOrVesBit)*.

9.7.5. Инвертирование отдельных битов

Для инвертирования выбранных битов в целочисленной переменной используется операция побитового исключающего «ИЛИ» (знак ^) с *прямой* константной маской:

```

mVar = mVar ^ (1 << mNomerBit);
mVar = mVar ^ mVesBit;
mVar = mVar ^ mMaskaBits;

```

Как уже отмечалось, эту операцию, как и другие подобные, оформляют в виде макроса *CplBit (mReg, mNomerOrVesBit)*.

9.7.6. Обмен частей переменных

Одной из типовых операций взаимного преобразования данных является обмен тетрад в байте, когда четыре младших бита помещаются на место старших, а четыре старших бита - на место младших. В ОВМ *x51* для обмена тетрад аккумулятора есть даже специальная ассемблерная команда **Swap A** (см. п. 7.8.2).

В Си можно обменивать тетрады *любой* байтовой переменной при помощи двух встречных операций сдвига:

$$mByteX = (mByteX \ll 4) | (mByteX \gg 4);$$

Заметим, что в первой операции сдвига биты младшей тетрады перемещаются в старшую, а младшая тетрада при сдвиге обнуляется. Во второй операции сдвига старшая тетрада перемещается на место младшей. А операция побитового «ИЛИ» объединяет байты со сдвинутыми тетрадами.

Аналогично можно осуществить обмен произвольных битов в беззнаковом целом (слове). Ниже приведен пример обмена младшего и старшего байтов:

$$mWordX = ((mWordX \ll 8) | (mWordX \gg 8));$$

9.7.7. Объединение (упаковка) битов разных переменных

В некоторых случаях, особенно при обслуживании нескольких внешних объектов через один порт, для *вывода* данных приходится объединять разные переменные, упаковывая группы битов на заданные места. Подобная задача возникает, например, в программах динамической индикации (см. п. 11.4.6), когда необходимо двоично-десятичную информацию установить в младшей тетраде, а номер знакоместа (декады) в следующих по старшинству битах. Или в простейших сетевых устройствах приходится одним байтом передавать адрес и код команды в разных битах.

Пусть для определенности необходимо объединить 5 битов одной переменной *mByteX* с 3 битами другой переменной *mByteY*, поместив последние в старшие разряды *mByteZ*. Для объединения групп битов используется побитовое «ИЛИ» со сдвигом вправо одного из операндов, причем незначащие биты первого байта должны быть предварительно обнулены (см. п. 9.7.4), а незначащие биты второго байта обнуляются автоматически при сдвиге (см. п. 9.6.2):

$$mByteZ = (mByteX \& mMaska3bits) | (mByteY \ll 5);$$

9.7.8. Разъединение (распаковка) переменной на биты

Необходимость в разделении переменной может возникнуть, например, в случае, если часть битов кодирует адрес абонента, а другая часть кодирует тип команды.

Пусть для определенности имеем обратный предыдущему пример и нужно разнести по разным байтам 5 младших и 3 старших бита. Проще всего получить младшие биты, обнулив (см. п. 9.7.3) значащие старшие биты:

```
mByte1 = mByteX & mMaska5bits;
```

Выделить старшие биты и поместить их на место младших можно при помощи обнуления младших и последующего сдвига результата. Но в этом случае можно обойтись без обнуления, ограничившись одним сдвигом, если вспомнить, что при сдвиге освобождающиеся биты заполняются нулями:

```
mByte2 = mByteX >> 5;
```

§9.8. Процедуры задержки

9.8.1. Общие сведения

При взаимодействии с внешними объектами процессор вынужден подстраиваться под них, отсчитывая, например, необходимые интервалы времени до или после каких-то событий. Подобный отсчет (формирование временного интервала) может производиться чисто программным путем или с использованием внутреннего таймера (см. §10.6).

Ниже для иллюстрации использования операторов цикла языка Си рассмотрены некоторые способы программной реализации процедур, которые выполняют задержку работы программы на некоторое время. Они традиционно имеют имя *Delay* и могут быть оформлены в виде библиотечных функций.

Программная задержка (пауза) может быть реализована любым из операторов цикла, поскольку от числа повторов этих операторов зависит время выполнения процедуры. Задавая требуемое число повторов в качестве параметра, можно управлять паузой в вычислениях.

Длительность одного повтора в цикле зависит в первую очередь от длительности машинного цикла процессора, которая определяется частотой кварцевого генератора и системой такти-

рования. Например, для стандартной ОВМ x51 длительность машинного цикла в 12 раз больше периода кварца и при частоте кварца 12 МГц равна 1 мкс.

Для оценки времени задержки можно воспользоваться ассемблерным кодом после компилятора, подсчитав время выполнения всего программного цикла и умножить на число повторов. Длительность одного повтора в операторах цикла зависит от количества ассемблерных команд, задействованных для выполнения цикла. Компилятор реализует операции **for** и **while** примерно схожими ассемблерными процедурами, не имеющими большого отличия между собой.

Существенным моментом является тип входного параметра. Чем короче формат, тем проще ассемблерная процедура и легче оценить расчетное время. Поэтому в дальнейшем в качестве типа входного параметра и локальных переменных будем использовать тип **unsigned char**, который является основным для 8-разрядной ОВМ x51.

Отметим также, что в этом случае оператор **do-while**, который проверяет условие выхода в конце цикла, реализуется более компактным кодом с применением ассемблерной команды **DJNZ** (см. п. 9.8.5).

Поэтому в дальнейшем при организации циклов будем по возможности использовать оператор **do-while**.

9.8.2. Реализация микросекундных задержек

Для очень малых (микросекундных) задержек на несколько машинных циклов использование операторов повтора может дать завышенные времена пауз. В таких случаях рекомендуется использовать ассемблерные вставки с пустыми командами, оформляемые, например, в виде макросов:

```
#define mDelay2mks {_asm nop nop _endasm;}
```

Для простоты будем считать в дальнейшем, что машинный цикл равен 1 мкс, что соответствует частоте кварца 12 МГц.

Число пустых ассемблерных команд выбирают, исходя из требуемого времени задержки, поэтому далее в примерах цифра в обозначениях микросекундных макросов будет равна количеству пустых команд в определении макроса.

Для небольших задержек можно использовать и операторы Си, не затрагивающие полезных переменных, например:

```
mTemp = mTemp;  
mTemp = 5+1;
```

Нужно только иметь в виду, что компилятор, оптимизируя выходной код, может исключить некоторые подобные операторы, особенно одинаковые и с последовательными операциями присваивания константы одной переменной. Поэтому в таких случаях рекомендуется проверять генерируемый компилятором ассемблерный код.

9.8.3. Реализация задержки при помощи оператора *for*

Данный оператор является наиболее наглядным и читаемым оператором цикла. Описанная ниже процедура задержки *mDelayFOR* имеет незаполненное тело оператора **for**, которое можно дополнить для увеличения времени задержки:

```
void mDelayFOR (byte mKodF) {
    byte ml;
    for (ml = 0; ml < mKodF; ml++) {
        ... // Здесь можно вставить дополнительные «пустые» команды.
    }
}
```

Для рассмотренной процедуры компилятор *SDCC* генерирует следующую последовательность ассемблерных команд:

<i>_mDelayFOR:</i>		
mov r2, dpl	; Значение <i>mKodF</i> (через DPL) в R2	(2 мкс)
mov r3, #0x00	; Переменная <i>ml</i> = 0 в R3	(2 мкс)
<i>_mD1:</i>	; Начало цикла for .	
clr c	; Сравнение	(1 мкс)
mov a, r3	; локальной переменной <i>ml</i>	(1 мкс)
subb a, r2	; и входного кода <i>mKodD</i> .	(1 мкс)
jc _mD2	; Проверка окончания	(2 мкс)
ret	; и выход из цикла.	(2 мкс)
<i>_mD2:</i>		
inc r3	; Инкремент локальной переменной (<i>ml++</i>)	(1 мкс)
jmp _mD1	;	(2 мкс)

9.8.4. Оценка времени задержки *mDelayFOR*

Нетрудно оценить, используя приведенные справа в скобках стандартные времена выполнения отдельных команд, общее время выполнения *основного* цикла $t_{\text{ц}}$ (от метки *_mD1* до последней команды), которое в данном случае равно 8 мкс.

Для получения более «круглой» (десятимикросекундной) цены одной единицы параметра *mKodF* внутрь цикла **for** можно вставить

пустые ассемблерные команды **nop**. Конкретный пример такой вставки рассмотрен в п. 9.8.5.

Для оценки общего времени задержки следует добавить время вспомогательных действий процедуры вне цикла t_b – для вызова процедуры с передачей параметра (≈ 4 мкс), инициализации локальных переменных (≈ 4 мкс) и возврата (≈ 2 мкс). Тогда время задержки (t_3) процедуры **mDelayFOR** будет равно в *микросекундах*:

$$t_3 = mKodF \cdot t_u + t_b. \quad (9.1)$$

Тогда, задаваясь требуемым временем задержки t_{TP} , можно получить формулу для приближенной оценки требуемого кода, как результата целочисленного деления:

$$mKodF = (t_{TP} - t_b) / t_u. \quad (9.2)$$

Формулу (9.2) можно реализовать и в программе, но в случае малых времен, чтобы не вносить дополнительной задержки, лучше вычислить код вне программы и использовать в виде константы.

Практически аналогичную по своим свойствам программируемую процедуру можно записать при помощи оператора **while**, но наиболее компактный ассемблерный код получается при использовании оператора **do-while**.

9.8.5. Реализация задержки при помощи оператора **do-while**

Этот оператор проверяет условие выхода в конце цикла, подобно составной ассемблерной команде **DJNZ** (см. п. 7.11.5), которая выполняет декремент и ветвление по ненулевому результату.

```
void mDelayDOWHILE (byte mKodW) {
    do {
        _asm
            nop                // Можно исключить
            nop                //      или добавить.
        _endasm;
    } while (--mKodW);
}
```

Компилятор **SDCC** для данного случая генерирует следующий ассемблерный код:

```
_mDelayDOWHILE:
    mov    r2, dpl           ; Входной код mKodD в R2      (2 мкс)
_mD:      ; Начало цикла do { }.
```

nop	;	(1 мкс)
nop	;	(1 мкс)
djnz r2, _mD	; Это while (--mKodD)	(2 мкс)
ret	;	(2 мкс)

Малое время основного цикла ($\approx 2\text{--}3$ мкс без пустых команд) позволяет рекомендовать указанную процедуру для генерации программируемых микросекундных пауз. Но для получения значений, кратных десяти, в тело цикла обычно добавляют пустые команды.

Подсчитав время цикла (используя, как и ранее, машинные длительности, помеченные в программе справа), можно получить формулу для оценки кода *mKodW*, подобную (9.2).

Для примера ниже приведены макросы для «круглых» пауз в 0,1 и 1 мс:

```
#define mDelay01ms mDelayDOWHILE (25)
#define mDelay1ms mDelayFOR (99)
```

Замечание о приближенности оценок времени задержки справедливо и в данном случае.

9.8.6. Эмуляция «паскалевской» процедуры *Delay*

Используя ранее определенный макрос *mDelay1ms*, можно эмулировать известную процедуру *Delay* языка Паскаль, которая выполняет задержку на заданное число миллисекунд:

```
void mDelayMS (word mKodMS) {
    do mDelay1ms;
    while (--mKodMS);
}
```

Параметр mKodMS задает число миллисекунд интервала ожидания. Данная процедура, как и предыдущие, является приближенной и зависящей от тактовой частоты процессора.

В тех случаях, когда нужны более короткие управляемые паузы, в рассмотренной процедуре вместо *mDelay1ms* можно использовать макрос *mDelay01ms*.

9.8.7. Использование и недостатки процедур задержки

Операции задержки нужны не сами по себе, а для синхронизации действий при обслуживании внешних объектов. Большая часть примеров по их использованию даны в §§10.1 и 10.2, где описывается ввод-вывод информации через порты ОВМ. Так, например, при ге-

нерации импульсов процедуры задержки определяют временные параметры импульсов (см. п. 10.1.1). При вводе информации от ключа с дребезгом (см. п. 10.1.4) пауза позволяет переждать переходный процесс. При обслуживании ЦАП и АЦП процессор может учитывать их инерционность, используя небольшие паузы.

При практическом использовании подобных процедур обязательно нужно помнить, что прерывания (см. §10.3), которые возникают в моменты работы программной задержки, неконтролируемо увеличивают время паузы.

Общим недостатком всех программных задержек является не столько их приблизительность (хотя в конкретном случае подстройкой нетрудно добиться и сравнительно точных отсчетов времени), а то, что процессор в них не совершает полезной работы.

Поэтому сравнительно длительные программные задержки следует применять лишь в простых устройствах, где процессор не загружен полностью. В более производительных системах следует использовать для отсчета времени внутренний таймер (см. §10.5).

§9.9. Примеры программ преобразования кодов

9.9.1. Вычисление контрольной суммы массива во внешнем ОЗУ

Одним из способов проверки правильности полученных кодов является метод контрольных сумм, когда массив данных снабжается дополнительным числом – контрольной суммой всех чисел определенного формата. При выходе суммы за рамки формата сложение чисел массива продолжается без учета переполнения. Например, контрольное число может вычисляться в формате байта без учета переполнения, когда суммирование осуществляется по модулю 256.

Для примера будем использовать 8-байтный массив во внешнем ОЗУ (область **xdata**) и по определенному адресу (**at 1000**):

```
xdata at 1000 byte mBufR [8];
```

Для индексации применим локальную байтовую переменную **mPtr**:

```
byte mProcCalcCS (void) {
```



```

    byte mPtr, mCS = 0;
    for (mPtr = 0; mPtr < 8; mPtr++) {
        mCS = (mCS + mBufR [mPtr])
    }
    return mCS;
}

```

Пример вызова этой функции:

```
mByteCS = mCalcCS();
```

9.9.2. Прием данных в буфер со сдвигом

В некоторых случаях необходимо последовательно заносить данные в буферный массив, например, сетевой пакет, данные от АЦП для вычисления скользящего среднего значения (см. далее), организация очереди для обработки данных и т. д.

В подобных случаях можно пользоваться счетчиком-указателем для локализации места в массиве. Но в ряде случаев удобнее пользоваться формой приема данных типа продвигающейся очереди, когда весь массив сдвигается и на освободившееся место записывается новая порция данных. Отметим, что в этом случае последний элемент массива данных считается уже обработанным и теряется.

Для примера рассмотрим прием кода со сдвигом от АЦП (см. п. 18.3.1) в массив из пяти чисел, объявленный в программе заранее:

```

word mProcGetADC () {
    ml = 4;
    do {
        mMass [ml] = mMass [ml - 1];           // Сдвиг.
    } while (--ml);
    mMass [0] = mGetADC;
}

```

9.9.3. Скользящее осреднение результатов оцифровки

Одним из простейших, но достаточно эффективных методов предварительной обработки числового ряда данных является скользящее осреднение (*moving average*) [26].

Предположим, что требуется вычислить скользящее среднее значение двухбайтовых данных после оцифровки для окна из 5 значений, используя данные предыдущего примера. Массив данных при

приеме сдвигается, поэтому программа упрощается и сводится к вычислению среднего значения элементов массива:

```
word mProcMovAv5 () {  
    word mMA5 = 0;  
    for (ml = 0; ml < 5; ml++) mMA5 = mMA5 + mMass[ml];  
    return mMA5 / 5;  
}
```

10 УПРАВЛЕНИЕ РЕСУРСАМИ x51 НА ЯЗЫКЕ Си

§10.1. Вывод-ввод информации битовыми командами

10.1.1. Генерация пачки импульсов

При управлении внешними объектами релейного типа одной из типовых задач является включение-выключение сигналов через определенные промежутки времени. В качестве примера рассмотрим генерацию симметричных прямоугольных импульсов с периодом *mKodT*, количество которых задано в переменной *mKvol*:

```
void mProclmpS (byte mKvol, mKodT) {  
    do {  
        mPin = 1;  
        mDelay (mKodT / 2);  
        mPin = 0;  
        mDelay (mKodT / 2);  
    } while (mKvol--)  
}
```

Данная процедура (которую можно использовать, например, для звуковой сигнализации), в цикле устанавливает и сбрасывает бит порта *mPin*, используя одну из ранее рассмотренных подпрограмм управляемой задержки (см. §9.8).

Эту процедуру можно также записать с использованием оператора **for**. Ниже это сделано на примере генерации несимметричных импульсов, для которых задана длительность импульса *mKodTi*, а длительность паузы вычисляется на основе периода *mKodT*, определенного в теле процедуры:

```
void mProclmpNS (byte mKvol, mKodTi) {
```

```

byte mKodT = 255, mI;
for (mI=0; mI < mKvol; mI++) {
    mPin = 1;
    mDelay (mKodTi);
    mPin = 0;
    mDelay (mKodT - mKodTi);
}
}

```

10.1.2. Генерация звука «бип-бип»

Продолжая пример со звуковой сигнализацией, можно реализовать прерывистый сигнал («бип-бип», почти как в первом советском спутнике), ограничившись количеством «бипов» *mKvoBip*, заданных в теле процедуры при инициализации:

```

void mBip (byte mKvol, mKodT) {
    byte mKvoBip=5;           // Количество «бипов».
    do {
        mProcImpS (mKvol, mKodT); // Это пачка импульсов – «бип».
        mPinSvetodiod ^= 1;       // Это световая сигнализация.
        mDelay (mKvol);           // Пауза между «бипами».
    } while (mKvoBip--)
}

```

Основой для этого примера являются ранее описанные процедуры генерации пачки импульсов и задержки (см. п. 10.1.1). В некоторых случаях звуковая сигнализация сопровождается световой. Для этого в программе используется оператор инвертирования бита световой сигнализации (см. п. 9.5.3).

10.1.3. Ввод сигнала от контактного датчика

Пусть необходимо в зависимости от состояния внешнего ключа организовать движение «бегущего огонька» влево или вправо (рис. 10.1). Чтобы состояние ключа K_K было правильно определено программой нужно, чтобы внутренний ключ бита порта K_P находился в непроводящем состоянии, то есть в бит порта была предварительно записана единица (см. п. 7.2.2).

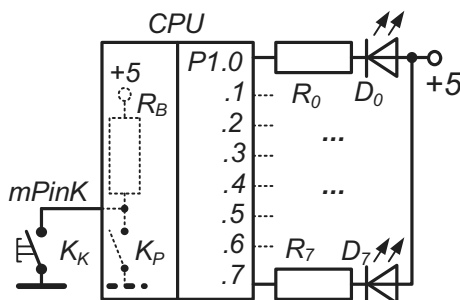


Рис. 10.1

Особенностью используемой схемы является также инверсный уровень активного сигнала. Например, если кнопка не нажата (ключ не проводит), то на выводе *mPinK* будет читаться единичный сигнал, обусловленный наличием внутреннего сопротивления R_B , подключенного к источнику питания (см. п. 7.2.2). Когда кнопка нажата, контактный ключ K_K проводит, и с вывода *mPinK* будет читаться нулевой (активный) сигнал.

В дальнейшем при чтении подобных сигналов будем считать активным именно нулевой уровень и при необходимости инвертировать его.

Учитывая, что «бегущий огонек» реализуется операцией сдвига, сделаем ее условной и зависящей от значения бита *mPinK*. Для простоты не будем учитывать «дребезг» контакта (см. далее). Кроме того, отметим, что светодиоды индикатора, подключенные к положительному полюсу источника питания, будут светиться при нулевом выходном сигнале, поэтому в программе используется побитовая операция инвертирования:

```
void mProcFire (void) {
    P1 = ~1;           // Зажигаем нулем один светодиод.
    while (1) {        // Бесконечный цикл.
        mDelayMS(1000); // 1-секундная пауза.
        if (mPinK) P1 = P1 >> 1; // Сдвиг
        else P1 = P1 << 1;       // «Огонька».
        if (!(~P1)) P1 = ~1;     // «Огонек» пропал? Зажигаем снова.
    }
}
```

10.1.4. Счет числа нажатий кнопки с «дребезгом»

При нажатии и отпускании кнопки (контактного ключа) возникает переходный процесс, называемый «дребезгом» (рис. 10.2), который длится 5–10 мс, и может привести к неправильному считыванию информации. Простейшим способом игнорирования «дребезга» контактов является реагирование программы на первый перепад сигнала и последующая пауза, бульшая ожидаемого времени «дребезга».

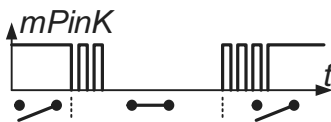


Рис. 10.2

Пусть для примера необходимо реализовать программный счетчик числа нажатий кнопки с выводом числа импульсов (в инверсном

коде) через двоичный светодиодный индикатор, подключенный к порту **P1** (используем схему рис. 10.1):

```
void mProcCount (void) {  
    P1 = 0; // Обнуляем счетчик-индикатор.  
    while (1) { // Бесконечный цикл.  
        while (mPink); // Ждем нажатия.  
        mDelayMS(10); // Пережидаем «дребезг» контакта.  
        while (!mPink); // Ждем отпущения  
        mDelayMS(10); // Пережидаем «дребезг» контакта.  
        P1++; // Счет числа срабатываний  
    }  
}
```

Отметим также, что счет числа импульсов (без «дребезга») можно организовать с использованием внешнего прерывания или таймера (см. п.п. 10.4.4 и 10.7.5).

§10.2. Вывод-ввод байтовыми командами

Для иллюстрации использования байтовых команд при вводе-выводе отдельных битов в данном параграфе не будем пользоваться битовыми командами x51.

10.2.1. Управление простейшим светофором

Пусть необходимо разработать устройство для управления лампами простейшего светофора на пешеходном переходе в трех режимах работы, автоматическом, ручном (по запросу пешехода) и необслуживаемом (мигающий желтый).

Функциональная схема приведена на рис. 10.3. Лампы светофора и пешеходов подключаются через ключевые усилители мощности, которые на схеме не показаны. Для задания режима используется переключатель *Sw1* (на три положения), а для запросов пешехода – кнопки *Sw2* и *Sw3*, соединенные по схеме «монтажного ИЛИ».

*Особенностью схемы является то, что биты для вывода упакованы вместе с битами для ввода в байтовый формат порта **P1**, что нужно учитывать при программировании.*

На рис. 10.4 приведены временные диаграммы работы в ручном режиме. На основе этих диаграмм с учетом значимости (веса) используемых битов можно определить байтовые десятичные константы для различных состояний светофора:

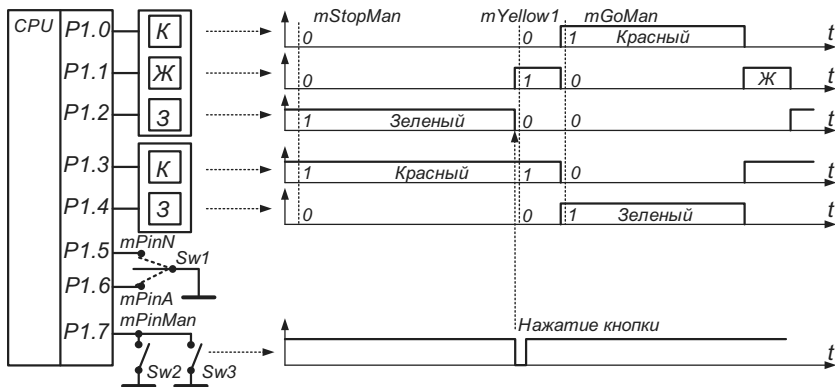


Рис. 10.3

Рис. 10.4

```

#define mStopMan    12    // Красный для пешеходов,
                          //      зеленый для автомобилей.
#define mGoMan      11    // Зеленый для пешеходов,
                          //      красный для автомобилей.
#define mYellow1    10    // Красный для пешеходов,
                          //      желтый для автомобилей.
#define mYellow2     2    // Желтый для автомобилей.
#define mPinN       32    // Биты необслуживаемого
#define mPinA       64    //      и автоматического режимов.
#define mPinMan     128    // Кнопки пешеходов.
#define mPins {mPinMan + mPinA + mPinN} // Биты для ввода.

```

10.2.2. Программа для простейшего светофора

Работа светофора, как видно из временных диаграмм на рис. 10.4, заключается в формировании временных интервалов и включении соответствующих сигналов в зависимости от режима работы. Причем автоматический и ручной режимы отличаются только наличием ожидания запроса пешеходов при движении автомобилей, поэтому эти два режима будем реализовывать единым фрагментом *mProcAvt*, описанным далее.

Главная программа должна начинаться с анализа битов режима, при котором нужно учитывать, как и прежде, что активным уровнем при выборе режима или нажатии кнопки является нулевой:

```

void main (void) {
    P1 = mPins;           // Гасим все лампочки, сохраняя единицы на кнопках.
    while (1) {           // Бесконечный главный цикл.
        if (!(P1 & mPinN)) mProcMig(); else mProcAvt();
    }
}

```

```

    // Если не «мигающий» режим, то автоматический или ручной.
  }
}

```

Процедура для необслуживаемого режима является самой простой, она должна включать и выключать желтый сигнал светофора:

```

void mProcMig (void) {
    P1 = mPins | mYellow2;           // Включаем желтый,
    mDelayMS(1000);                  // пауза 1 с,
    P1 = mPins;                      // выключаем все,
    mDelayMS(1000);                  // пауза 1 с.
}

```

Автоматический и ручной режимы в процедуре *mProcAvt* имеют большее число состояний, причем в ручном режиме дополнительно должно осуществляться ожидание нажатия кнопок пешеходами. В связи с этим отметим, что для правильного ввода биты, помеченные словом *Pin*, должны находиться в единичном состоянии (см. п. 7.2.2). Поэтому вывод байта в порт *P1* осуществляется при помощи логической операции «ИЛИ» и участия константной маски *mPins* с единицами на месте кнопок:

```

void mProcAvt (void) {
    P1 = mPins | mStopMan;           // Зеленый для автомобилей,
    mDelayMS(30000);                 // пауза 30 с.
    if (P1 & mPinA) while (!(P1 & mPinMan)); // Если режим ручной,
                                           // то ждем пешехода.

    P1 = mPins | mYellow1;           // Включаем желтый,
    mDelayMS(2000);                  // пауза 2 с.
    P1 = mPins | mGoMan;             // Зеленый для пешехода,
    mDelayMS(15000);                 // пауза 15 с.
}

```

Рассмотренные схема и программа являются иллюстративными. Их можно усложнить, например, ввести мигание зеленого сигнала перед включением красного, задействовать таймер для отсчета паузы, добавить связь с удаленным компьютером и т. д.

10.2.3. Вычисление скан-кода матрицы ключей

При обслуживании клавиатуры для уменьшения числа используемых внешних выводов контактные ключи часто объединяются в матрицу (рис. 10.5). Причем для вычисления порядкового номера контакта (скан-кода) необходимо определить две координаты нажатой клавиши – номер строки, который обозначим переменной *mStroka*, и номер столбца (*mStolb*).

Тогда, учитывая, что счет порядкового номера клавиши происходит в сторону увеличения номера столбца, вычисление скан-кода нажатой клавиши можно осуществлять по формуле:

$$mScanKod = mStroka + mStolb * mKvoStrok; \quad (10.1)$$

где $mKvoStrok$ – количество строк в матрице.

Чтобы упростить вычисления и заменить умножение операций сдвига влево, обычно выбирают величину $mKvoStrok$ равной степени двойки (2, 4, 8 и т. д.).

Для нашего случая $mStolb = 0..3$, $mStroka = 0..3$, то есть $mKvoStrok = 4$, и умножение можно заменить двойным сдвигом.

$$mScanKod = mStroka + (mStolb \ll 2);$$

При определении координаты нажатой клавиши используют в основном два метода: поочередный опрос (сканирование) столбцов и более простой метод – двунаправленный опрос матрицы, который рассматривается далее.

10.2.4. Двунаправленный опрос матрицы ключей

основан на принципе переключения направления опроса матрицы клавиатуры, что легко делается при использовании ОВМ.

Строки и столбцы подключаются к двум различным тетрадам, одна из которых первоначально запрограммирована на вывод, а другая – на ввод.

Вначале включаются все столбцы (рис. 10.6) – обнулена старшая тетрада, а в младшей – все единицы, чтобы обнаружить нулевой сигнал нажатия (см. п. 10.1.3). После этого в цикле читается состояние строк. При наличии нажатия определяется номер строки, где находится нажатая клавиша.

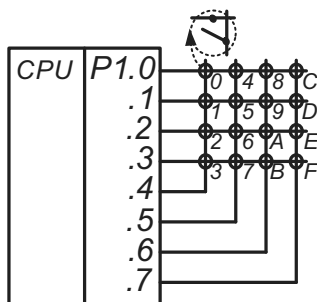


Рис. 10.5

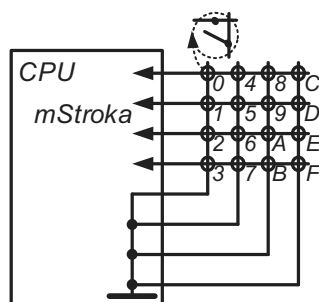


Рис. 10.6

После этого происходит перепрограммирование направления ввода-вывода (старшая тетрада – на ввод, а младшая – на вывод), то есть столбцы и строки как бы меняются местами (рис. 10.7). И определяется номер столбца для нажатой клавиши. Зная номер строки и столбца, можно определить (закодировать) порядковые номера активных строки и столбца (см. п. 11.7.5), а затем вычислить скан-код клавиши (см. п. 10.2.4).

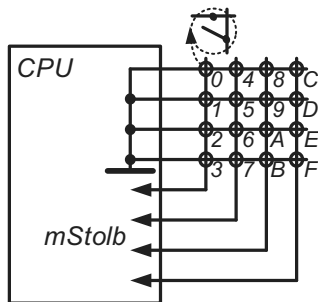


Рис. 10.7

10.2.5. Текст программы для двунаправленного опроса

имеет практически линейную структуру:

```
byte mProcGetChar (void) {
    P1 = 0x0f;           // Включаем все столбцы.
    While (P1!=0x0f);    // Ждем нажатия клавиши.
    mDelayMS (10);       // Пережидаем дребезг.
    mStrok = mCoder(~P1 & 0x0f); // Читаем-кодируем строку (см. п. 11.7.5).
    P1 = 0xf0;           // Включаем все строки.
    mStolb = mCoder(~P1 >> 4); // Читаем-кодируем столбец (см. п. 11.7.5).
    return (mStrok + (mStolb << 2)); // Возвращаем скан-код (см. п. 10.2.3).
}
```

Отметим, что при опросе состояния тетрады полученный байт порта *инвертируется*, а неинформационная тетрада гасится операцией «И» (см. п. 9.7.4) или заполняется нулями при сдвиге (см. п. 9.6.2).

Реальные программы, конечно, более сложны, они могут обрабатывать одновременно нажатые клавиши, отслеживать как нажатие, так и отпускание клавиши и производить другие необходимые действия.

§10.3. Регистры и биты системы прерывания

10.3.1 Управление разрешениями и приоритетами

В стандартной модели x51 прерывания обслуживают два *побитно доступных* регистра **IE (Interrupt Enable)** – для управления разреше-

ниями прерываний и **IP (Interrupt Priority)** – для управления приоритетами. Как правило, биты этих регистров используются без упоминания регистров.

*Например, разрешение отдельных источников прерывания осуществляется в регистре **IE** при помощи именованных битов, приведенных ниже.*

*Два бита **EX0, EX1 (Enable eXternal)** – разрешают прерывания от внешних сигналов на выводах **Int0 (P3_2)** и **Int1 (P3_3)**.*

*Два бита **ET0, ET1 (Enable Timer)** – разрешают прерывания по переполнению таймеров **T0** и **T1**.*

*Бит **ES (Enable Serial)** – разрешает прерывание от приемника и передатчика последовательного порта.*

Эти биты при инициализации или по ходу программы можно устанавливать или сбрасывать:

```
EX0 = 0;      // Запретить внешнее прерывание Int0.  
ET0 = 1;      // Разрешить прерывание по таймеру T0.  
EA = 1;      // Разрешить все разрешенные прерывания.
```

Управление двумя уровнями приоритетов («1» – высокий, «0» – низкий) осуществляется при помощи аналогичных битов **PS, PX0, PX1, PT0, PT1** в регистре **IP**. Напомним, что источник прерывания с высоким приоритетом может прервать работу низкоприоритетной процедуры.

Часть битов, управляющих работой системы прерываний и описанных ниже, были размещены разработчиками ОВМ в спецрегистрах, не имеющих отношения к прерываниям. Особых неудобств это не вызывает, так как подобные регистры доступны побитно, а программист обращается к битам без указания места их размещения.

10.3.2. Биты типа внешних прерываний

Биты управления типом внешних прерываний **IT0, IT1 (Interrupt Type)** упакованы вместе с таймерными битами в регистре **TCON**. И тип внешнего прерывания программистом задается по простому правилу: если прерывание вызывается нулем, то бит типа должен быть равен нулю. Если перепадом, то – единице:

```
IT0 = 1;      // Отрицательный перепад запоминается (см. далее).  
IT0 = 0;      // При нулевом уровне (не фиксируется внутри).
```

10.3.3. Биты запросов прерываний

также упакованы вне спецрегистров прерывания и перечислены ниже.

IE1, IE0 (Interrupt Edge) – биты запросов от перепадов сигналов внешних прерываний, которые при **IT** = 1 фиксируются при помощи внутренних триггеров.

TF0, TF1 (Timer overflow Flag) – флажки переполнения таймеров (17-е разряды таймеров).

TI (Transmitter Interrupt), RI (Receiver Interrupt) – запросы от передатчика и приемника последовательного порта, которые используют общую особую ячейку (вектор прерывания) в памяти команд.

Четыре первых бита размещены в **TCON**, а два последних – в **SCON**. Отметим также, что при сбросе все указанные биты обнуляются, и нет большой нужды сбрасывать их при инициализации.

Общей особенностью запросов прерываний является то, что эти биты *устанавливаются внутренней аппаратурой*, а сбрасываются или системой прерываний после перехода на прерывающую процедуру или программистом, если механизм прерываний не задействован.

Исключением являются биты **TI** и **RI** запросов от последовательного порта, вызывающие общую прерывающую процедуру. Они устанавливаются автоматически при готовности передатчика или приемника, а **сбрасываться должны программой**, например, в процедуре прерывания (см. параграфы 10.10 и 10.11).

Примеры сброса битов запросов:

TF0 = 0;

TI = 0;

Все рассмотренные флажки запросов можно читать и проверять программно, не привлекая механизм прерываний, используя, например, условные операторы **if** или **while**:

while (IRI); // Ждать, пока приемник не получит посылку.

Но чаще всего перечисленные биты запросов устанавливаются, анализируются и сбрасываются (кроме **TI** и **RI**) механизмом прерываний, неявно для программиста.

§10.4. Оформление прерывающих процедур

10.4.1. Объявление прерывающей процедуры

Прерывающая программа на ассемблере должна явно содержать ряд стандартных фрагментов – заполнение особой ячейки в ПЗУ (вектора прерываний), сохранение и восстановление модифицируемых программой регистров общего назначения (см. п. 8.5.2). Эти фрагменты оформляются в выходном коде компилятора автоматически при объявлении типа процедуры с использованием ключевого слова ***interrupt*** (или ***signal*** в некоторых компиляторах):

```
void mIntProc () interrupt mNomerInt using mBank {
    ...           // Здесь размещается текст прерывающей процедуры.
}
```

В качестве номера прерывания *mNomerInt* можно подставлять константы от 0 (внешнее прерывание 0) до 4 (прерывание от последовательного порта). Но для лучшей читаемости программы лучше использовать символические имена из заголовочного файла OBM (в нашем случае *at89x51.h*):

```
IE0_VECTOR      // Адрес 0x03 для внешнего прерывания 0.
TF0_VECTOR      // Адрес 0x0b для переполнения таймера 0.
IE1_VECTOR      // Адрес 0x13 для внешнего прерывания 1.
TF1_VECTOR      // Адрес 0x1b для переполнения таймера 1.
SI0_VECTOR      // Адрес 0x23 для последовательного порта.
```

Ключевое слово ***using***, указывающее номер используемого регистра банка памяти *mBank* (от 0 до 3), чаще всего опускают, позволяя компилятору самому определить его.

*Обычная процедура прерывания на Си сохраняет в стеке много лишнего. Для минимизации можно использовать ключевое слово ***_naked***, а процедуру прерывания написать на ассемблере (см. п. 9.2.5). Тогда генерируемый компилятором Си код будет более компактен. Подобный прием можно применять и для обычных процедур.*

В критических случаях, когда прерывание прерывающей процедуры нежелательно, используют ключевое слово ***critical***, тогда все прерывания до завершения процедуры будут запрещены. В принципе это можно сделать и самим, сбросив бит ***EA*** в начале процедуры и установив его в конце.

И вообще следует по возможности упрощать прерывающую программу, чтобы ускорить ее выполнение и не монополизировать доступ к процессору.

10.4.2. Результат работы прерывающих процедур

В отличие от остальных процедур функции-прерывания не должны иметь аргументов и возвращать значения. Они могут только изменять ранее определенные переменные, но делают это неожиданно (асинхронно) для основной программы и остальных процедур. Вследствие этого изменения общих переменных или портов ввода/вывода, производимые в ходе прерывания, могут оказаться какое-то время незамеченными.

Поэтому, чтобы заставить компилятор каждый раз считывать значение такой переменной (то есть запретить сохранение ее во временном регистре), обязательно нужно использовать при описании подобных переменных модификатор **volatile** (см. п. 9.3.3).

10.4.3. Инициализация прерываний

осуществляется в начале или по ходу программы и заключается в установке разрешений используемых источников прерываний и, если нужно, их приоритетов (см. п. 10.3.1). Для внешних прерываний обязательно нужно установить тип, по уровню или перепаду (см. п. 10.3.2). Ниже приведены несколько операторов для подобной инициализации:

IT0 = 1;	// Тип внешнего прерывания 0 - по перепаду.
EX0 = 1;	// Разрешить внешнее прерывание 0
ET0 = 1;	// и по переполнению таймера T0.
EA = 1;	// Разрешить работу системы прерываний.

Вместе с тем, следует отметить, что достаточно часто разрешения прерываний в программе явно не сгруппированы, поскольку происходят в рамках задания режимов работы отдельных устройств, например, таймера или последовательного порта (см. п.п. 10.6.4 и 10.9.6).

10.4.4. Пример процедуры для счета внешних импульсов

Пусть необходимо считать количество импульсов на входе **Int0** (внешний вывод **P3_2**). Поскольку процедура прерывания не должна возвращать значение и иметь параметров (см. п. 10.4.2), будем полагать, что ранее для счета была объявлена переменная **mCountImp**, (с моди-

фикатором **volatile**), размерность которой определена в соответствии с техническим заданием. Подставляя в ранее рассмотренную форму процедуры (см. п. 10.4.1) стандартное имя вектора нулевого внешнего прерывания (**IE0_VECTOR**), получим следующий текст:

```
void mProclnt0 (void) interrupt IE0_VECTOR {
    mCountImp++;           // Инкремент счетчика импульсов.
}
```

10.4.5. Измерение частоты внешних импульсов

Предположим, что счет внешних импульсов нам нужен для измерения частоты сигнала, которая численно равна количеству импульсов на интервале в одну секунду (см. п. 13.4.1). Пусть односекундный интервал отмеряет таймер, работающий в режиме прерывания по переполнению, и его прерывающая программа устанавливает в начале секундного интервала флаг *myFlag1s* (см. п. 10.6.6).

Таким образом, будут независимо работать два процесса: счет импульсов и счет времени. И можно, например, в главном цикле программы, организовать их взаимодействие, разрешая и запрещая счет внешних импульсов по прерыванию:

```
    while (!mFlag1s);           // Ждем начало измерения.
    EX0 = 1;                    // Разрешаем счет импульсов.
    mFlag1s = 0;                // Сбрасываем флажок.
    while (!mFlag1s);           // Ждем окончания измерения.
    EX0 = 0;                    // Запрещаем счет импульсов.
    mFlag1s = 0;                // Сбрасываем флажок.
    mToBufIndik(mCountImp);     // Результат – на индикацию (см. §11.4).
```

Примеры прерывающих программ для таймеров и последовательного порта даны в соответствующих §§ 10.6, 10.10 и 10.11.

§10.5. Регистры, биты и режимы таймеров

В стандартном варианте ОВМ x51 имеет два 16-разрядных таймера-счетчика. Таймер 0 представляется программисту в виде байтовых спецрегистров данных, старшего **TH0** и младшего **TL0**. Таймер 1 – соответственно **TH1** и **TL1**. Кроме того, таймеры имеют и два общих 8-разрядных регистра, режима **TMOD** и управления **TCON**.

10.5.1. Регистр *TCON* и пуск-останов счета

Регистр *TCON* является побитно доступным, поэтому его биты чаще всего используются без упоминания самого регистра, тем более, что в нем, кроме таймерных битов, размещены 4 бита системы прерываний (*IT0*, *IT1*, *IE0* и *IE1* – см. п. 10.3.2).

Таймерных битов в регистре *TCON* четыре. Два бита для пуска – *TR0*, *TR1* (от *Timer Run*). Они устанавливаются и сбрасываются программно, например:

```
TR0 = 1;           // Разрешить счет таймера 0.
TR1 = 0;           // Остановить счет таймера 1.
```

Еще два бита служат для индикации переполнения разрядной сетки таймеров – *TF1*, *TF0* (от *Timer overflow Flag*). Устанавливаются *автоматически* при переполнении 8- или 16-разрядной сетки и служат в качестве запроса прерывания по переполнению (см. п. 10.3.3), то есть для аппаратного вызова подпрограмм обслуживания внешних устройств в моменты, определяемые по таймеру.

Сбрасываются биты переполнения неявно для программиста (аппаратно) при передаче управления системе прерываний.

Иногда в программе эти флажки переполнения могут анализироваться и сбрасываться без привлечения механизма прерывания (см. п. 10.6.3).

10.5.2. Формат регистра *TMOD* и задание режима

В отличие от многих спецрегистров *x51* регистр *TMOD* не доступен побитно, поэтому при задании кода режима необходимо использовать *байтовые операции*.

Для этого необходимо учитывать местоположения битов в регистре (рис. 10.8). Биты для таймера 0 упакованы в младшей тетраде, таймера 1 – в старшей. Для облегчения записи и чтения текста программ следует использовать символические имена битов режима определенные компилятором в заголовочном файле для конкретной ОВМ.

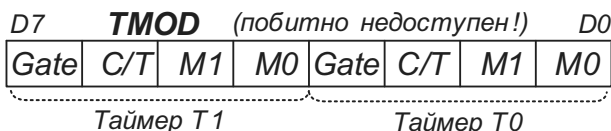


Рис. 10.8

Обычно эти имена представляют комбинацию имени бита и номера таймера.

Например, в компиляторе **SDCC** биты режима для таймера 0 обозначаются как **M0_0**, **M1_0**, для таймера 1 – **M0_1**, **M1_1**. Эти биты задают в двоичном коде один из 4 возможных режимов работы для каждого из таймеров:

- *режим 0* – 13-разрядный режим, имитирующий работу таймера в устаревшем семействе ОВМ **x48**,
- *режим 1* – 16-разрядный счет,
- *режим 2* – 8-разрядный счет с перезагрузкой,
- *режим 3* – 8-разрядный счет в обоих байтах таймера 0, таймер 1 остановлен.

Не останавливаясь на подробном описании режимов работы таймеров, которые можно найти в [17], отметим, что чаще всего используются режимы 1 и 2, особенности которых рассмотрены в п.п. 8.7.2 и 8.7.3.

Вторая пара битов (**GATE** и **C/T**) каждой тетрады регистра **TMOD** позволяет организовать внешнее управление таймерами. Например, биты **C/T** (для таймера 0 используется имя **C_T0**, для таймера 1 – **C_T1**) определяют тип входных сигналов. При нулевом их значении на вход таймера подаются внутренние тактовые импульсы (с частотой в 12 раз меньше частоты кварца), а при единичном значении – внешние импульсы от выводов **P3_4** или **P3_5** (см. п. 8.6.2).

Биты внешнего разрешения работы таймеров (**GATE0** и **GATE1**) позволяют внешним сигналам действовать подобно битам запуска **TR0**, **TR1**, разрешая или запрещая счет. Более подробно их действие будет рассмотрено в §10.7.

10.5.3. Задание режима счета

обычно происходит в начале программы на этапе инициализации путем засылки символической байтовой константы в регистр **TMOD**.

*Нужно обязательно обратить внимание на то, как кодируются константы, присвоенные битам **TMOD** в заголовочном файле для ОВМ, весами разрядов (1, 2, 4, .. 128) или порядковым номером бита (от 0 до 7).*

В первом случае для установки нескольких битов (например, для задания режима 1 в таймере 0, и режима 2 в таймере 1) используют

операцию «ИЛИ» с операндом в виде набора *устанавливаемых в единицу имен битов*:

```
TMOD = M1_1 | M0_0;           // Таймер 0 – M1 = 0, M0 = 1.  
                                // Таймер 1 – M1 = 1, M0 = 0.
```

Во втором случае операндами должны быть единицы, сдвинутые влево на место именованных битов:

```
TMOD = (1 << M1_1) | (1 << M1_0);
```

Чтобы избежать ошибок, связанных с кодировкой отдельных битов режима и облегчить дальнейшее сопровождение программ, можно использовать *свои* символические константы, например, для режимов 1 и 2:

```
#define mVesMode16bitForT0           1  
#define mVesMode8bitReloadForT0      2  
#define mVesMode16bitForT1          16  
#define mVesMode8bitReloadForT1      32
```

Тогда оператор загрузки регистра **TMOD** станет более наглядным:

```
TMOD = mVesMode8bitReloadForT1 | mVesMode16bitForT0;
```

10.5.4. Изменение режима только одного из таймеров

В некоторых случаях изменение режима одного из таймеров должно происходить по ходу программы. В этом случае биты другого таймера затрагивать нежелательно, поскольку они могут устанавливаться в независимом от разработчика программном модуле (или библиотечной функции), поэтому эти биты нужно сохранить. Для этого необходимо сначала прочитать состояние регистра **TMOD**, обнулить заменяемую тетраду режима (используя логическую операцию «И») и установить новые биты (при помощи операции «ИЛИ»). Ниже дан пример последовательности действий при задании нового режима только таймера 0:

```
TMOD = TMOD & 0xf0;           // Обнуляем младшую тетраду  
TMOD = TMOD | (mMode16bitForT0); // Заносим режим T0
```

Эти две операции для компактности можно записать слитно:

```
TMOD = (TMOD & 0xf0) | (mMode16bitForT0);
```

Пример инициализации таймера приведен в п. 10.6.4.

10.5.5. Загрузка 16-разрядных регистров данных таймеров

При загрузке таймеров нужно иметь в виду две особенности.

Первая заключается в том, что коды в 16-битные таймеры нужно загружать при остановленном счете (бит **TR** = 0) и *побайтно*, поскольку компилятор в области спецрегистров **sfr** определяет чаще всего только байтовые переменные. Если загружаемый код уже подготовлен в целочисленной переменной или константе, то осуществляется поочередное копирование старшего и младшего байтов целого числа с *преобразованием типа* (см. п. 9.6.5), например:

```
TH0 = mIntVarOrConst >> 8;           // Загрузка старшего  
TLO = mIntVarOrConst;                 // и младшего байта
```

*Отметим, что константу рекомендуется объявлять директивой **define**, при этом упрощается код программного фрагмента (см. п. 9.3.4).*

Вторая особенность заключается в том, что таймер инкрементирует свое содержимое и код загрузки должен отсчитываться по числовой оси в большинстве случаев в обратную сторону от момента переполнения, то есть со знаком минус (см. п. 8.7.2).

10.5.6. Чтение регистров данных таймеров

Чтение, как и загрузку, чаще всего производят при *остановленном* счете, поскольку при чтении «на лету» возможна ошибка в старшем байте в момент переполнения младшего (см. п. 8.6.5).

Главной особенностью чтения является *преобразование отдельных байтов* таймера в целое беззнаковое число, которое происходит в два этапа. На первом этапе читается младший байт таймера, а старший при этом заполняется нулями. На втором – старший байт таймера со сдвигом объединяется по «ИЛИ» с ранее обнуленным байтом переменной.

Пример, для таймера 0:

```
mIntVar = TLO;  
mIntVar = mIntVar | (TH0 >> 8);
```

или в слитной записи:

```
mIntVar = TLO | (TH0 >> 8);
```

§10.6. Формирование интервалов времени при помощи таймеров

10.6.1. Общие сведения

При счете времени на вход таймера подаются внутренние тактовые импульсы (то есть бит $C/T = 0$), период которых обычно совпадает с длительностью машинного цикла. При этом стараются использовать внешний кварц с частотой $F_{кв} = 12 \text{ МГц}$, тогда частота обновления таймера будет равна 1 МГц , то есть таймер будет считать *микросекунды* (за исключением редко используемого режима 0). Такая круглая и точная величина дискреты упрощает организацию временных пауз, задержек и системных цифровых часов.

Содержимое таймера для счета времени можно читать с остановом счета или на лету (не рекомендуется). Но программа не будет успевать правильно читать и использовать полученный код, потому что время выполнения команд чтения и обработки двухбайтовых данных будет наверняка больше времени обновления таймера.

Поэтому для таймера своеобразным «моментом истины», который легко обнаруживается аппаратурой системы прерываний и программой, является момент переполнения таймера, фиксируемый по установке флажка **TF0** (или **TF1**) в единицу. Поэтому при использовании таймеров флажки переполнения играют определяющую роль. В дальнейшем для определенности будем использовать таймер 0.

10.6.2. Определение кода загрузки таймера

Ранее (см. п. 8.7.2) было показано, что для формирования заданного интервала времени код времени *mKodTime* должен откладываться по числовой оси в обратную сторону от момента переполнения (см. рис. 8.5), то есть со знаком минус. Для *знаковых* целочисленных величин это осуществляется при использовании оператора смены знака.

Однако знаковые целые числа имеют вдвое меньшее абсолютное значение (диапазон от -32768 до +32767), чем беззнаковые (от 0 до 65535). И, чтобы использовать все 16 битов регистров данных, вычисляют *обратный код* отрицательного числа, *инвертируя* все биты переменной и *добавляя* к ним *единицу* [24]. Кстати, этой единицей чаще всего пренебрегают, поскольку ее влияние на общий результат оказывается меньшим, чем время реакции системы прерывания

и прерывающей процедуры. Ниже приведен пример загрузки такого инверсного кода:

```
TH0 = ~(mKodTime >> 8);
TL0 = ~mKodTime;
```

10.6.3. Отсчет заданного времени без использования механизма прерываний

Пусть таймер 0 работает в режиме 16-битного счета, а код времени (в *мкс*) был предварительно задан в целочисленной переменной или константе *mKodTime*. Нижеприведенный фрагмент программы осуществляет загрузку таймера, его пуск и ожидание окончания заданного временного интервала, то есть установки флага переполнения:

```
void mDelayTimer0 (word mKodTime) {
    TR0 = 0;           // Останов таймера и сброс
    TF0 = 0;           // флага на всякий случай.
    TH0 = ~mKodTime >> 8; // Загрузка на отсчет
    TL0 = ~mKodTime;    // заданного времени
    TR0 = 1;           // Пуск таймера.
    While (!TF0);      // Ожидание окончания интервала времени.
    TR0 = 0;           // Останов таймера.
    TF0 = 0;           // Сброс бита переполнения
}
```

Отметим, что в подобных процедурах, как и ранее в процедурах задержки (см. §9.8), процессор не делает полезной работы, ожидая переполнения таймера. Поэтому при отсчете больших интервалов времени и организации, например, цифровых часов желательно использовать механизм прерываний (см. п. 10.6.5).

10.6.4. Пример инициализации таймера при работе в режиме прерывания

Прежде чем описывать процедуру прерывания по переполнению таймера, рассмотрим пример инициализации таймера 0, оформленный в виде процедуры, которая получает код требуемого времени, задает режим работы, побайтно загружает регистр данных, разрешает прерывания и счет таймера:

```
void mInitTimer0 (word mKodTime) {
    TR0 = 0;           // Останов таймера.
    TMOD = (TMOD & 0xf0) | (mMode16bitForT0);
```

```
TH0 = ~mKodTime >> 8;      // Загрузка старшего и
TL0 = ~mKodTime;           // младшего байтов.
ET0 = 1;                   // Разрешаем
EA = 1;                     // прерывания.
TR0 = 1;                   // и счет.
}
```

Нужно отметить, что режим работы таймера 1 чаще всего задается в рамках инициализации последовательного порта (см. п. 10.9.6) или в других процедурах см. п. 10.5.4).

10.6.5. Отсчет времени с использования механизма прерываний

В предыдущей процедуре *mDelayTimer0* (см. п. 10.6.3) основное время программы тратилось на ожидание момента установки флажка переполнения. Между тем в это время процессор мог бы выполнять другую работу по обслуживанию внешних объектов или обработке данных. Для этого ожидание переполнения таймера следует возложить на систему прерываний, которая при установке флажка **TF0** вызывает соответствующую процедуру прерывания. Отметим, что система прерываний и таймер должны быть предварительно инициализированы (см. п.п. 10.4.3 и 10.6.4).

Чтобы периодически отсчитывать заданный интервал времени, пропорциональный коду *mKodTime*, прерывающая программа должна обязательно перезагружать таймер на новый отсчет. Кроме того, в заданный момент времени она может выполнять и какие-то необходимые действия – устанавливать-сбрасывать биты портов, обслуживать цифровой индикатор или вести счет системных «тиков», секунд, минут и часов.

Но перегружать работой процедуру прерывания обычно не рекомендуется, так как в большой системе, обслуживающей много внешних процессов, можно задержаться с реакцией на событие, требующее немедленного реагирования. Поэтому в системах реального времени, где время реагирования критично, процедуры прерываний производят минимум необходимых действий и устанавливают флажки (семафоры) для других неспешных операций, которые выполняются под управлением диспетчера в главной процедуре. Пример использования таких флажков приведен в п. 10.4.7.

Пусть в нашем случае процедура прерывания будет производить минимум действий – устанавливать флажок системного «тика»

mFlagTik, который нужно определить заранее, и считать эти «тики» в переменной *mCountTik*.

Объявление прерывающей процедуры описано ранее (см. п. 10.4.1), в качестве номера прерывания следует использовать стандартное символическое имя **TF0_VECTOR**. Напомним также (см. п. 10.4.2), что прерывающие процедуры не получают аргументов и не возвращают результаты.

```
void mProclntT0 () interrupt TF0_VECTOR {
    TR0 = 0;                // Останов таймера.
    TH0 = ~mKodTime >> 8;  // Загрузка на отсчет
    TL0 = ~mKodTime;        // заданного времени.
    TR0 = 1;                // Пуск таймера.
    mFlagTik = 1;           // Установка флажка тиков.
    mCountTik ++;           // Счет системных "тиков".
    // Здесь при необходимости можно выполнять и другие действия.
}
```

Таймер для счета времени в системе обычно бывает один, а процессов, которым нужна привязка ко времени, – несколько. Поэтому, устанавливая флажки на соответствующих этапах счета, можно организовать временные отметки разного масштаба, которыми могут пользоваться программные процедуры.

*Следует отметить, что сброс флажков должна производить процедура, которая выполняется **последней**. Чтобы избежать ошибок, каждому фрагменту (процессу) можно назначить свой флажок.*

10.6.6. Использование таймера для отсчета одной секунды

При измерении частоты или счете времени иногда требуется привязка к односекундному интервалу времени (см. п. 10.4.5). Для этой цели можно использовать ранее рассмотренную процедуру прерывания (см. п. 10.6.5), включив туда установку флажка *mFlag1s*, считая, что количество «тиков» в одной секунде определяется константой *mTikPer1s*:

```
void mProc1s () interrupt TF0_VECTOR {
    TR0 = 0;                // Останов таймера.
    TH0 = ~mKodTime >> 8;  // Загрузка на отсчет
    TL0 = ~mKodTime;        // заданного времени.
    TR0 = 1;                // Пуск таймера.
    mCountTik ++;           // Счет системных "тиков".
    mFlagTik = 1;           // Установка флажка "тика".
}
```

```

    if (mCountTik == mTikPer1s) {
        mCountTik = 0;
        mFlag1s = 1;    // Установка флажка 1 с.
    }
}

```

Отметим повторно, что сброс бита *mFlag1s* будет осуществлять процедура, которая последней использует этот флажок (см. п.п. 10.4.5 и 10.4.7).

10.4.7. Использование флажков-семафоров

Пусть в программе с использованием флажка *mFlagTik* необходимо генерировать на внешнем выводе *mPin* симметричные импульсы с полупериодом, равным «тику», а также считать секунды, минуты и часы в переменных *mCountS*, *mCountM*, *mCountH* для организации цифровых часов:

```

if (mFlagTik) {
    mFlagTik = 0;                // Сброс обязательен.
    mPin = !mPin;                // Инвертирование импульса.
    if (mFlag1s) {
        mFlag1s = 0;            // Сброс обязательен.
        mCountS ++;             // Счет секунд.
        if (mCountS == 60) {
            mCountS = 0;
            mCountM ++;         // Счет минут.
            if (mCountM == 60) {
                mCountM = 0;
                mCountH ++;     // Счет часов.
            }
            // Аналогично можно осуществить
            //      счет дней и месяцев.
        }
    }
}
}

```

§10.7. Измерение длительности и частоты импульсов

10.7.1. Внешнее управление счетом таймера

Чтобы перевести один из таймеров в режим внешнего управления следует установить соответствующий бит **GATE** в регистре **TMOD** (см. п. 10.5.2). В этом случае единичное значение внешнего сигнала

на ножке **P3_2** (альтернативное имя **Int0**) для таймера 0 или **P3.3** (**Int1**) для таймера 1 будет разрешать счет внутренних импульсов, а нулевое – запрещать (рис. 10.9). Для счёта используются внутренние импульсы (то есть бит **C/T** = 0), период которых будем считать равным 1 мкс.

*Очевидно, что в этом случае прерывания по используемому входу **Int0** или **Int1** должны быть запрещены.*

Отметим также, что общий пуск-останов таймеров по-прежнему осуществляется битами **TR0** или **TR1**, но таймер считать не будет, пока не придет внешнее разрешение.

Например, для измерения длительности импульса (см. п. 13.4.1) при помощи таймера 1 (рис. 10.9), нужна следующая последовательность действий:

- обнулить регистры данных таймера;
- установить бит **GATE1** и режим 16-разрядного счета);
- дождаться окончания импульса, анализируя бит **Int1**;
- запустить таймер на счет (но считать таймер будет только при единичном внешнем сигнале);
- дождаться момента завершения измерения, отследив программно начало и затем окончание импульса;
- сохранить оба байта таймера в целочисленной переменной.

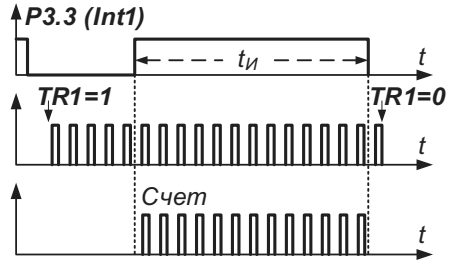


Рис. 10.9

10.7.2. Программа для измерения длительности импульса соответствует вышеописанной последовательности действий:

```
word mProclzmDlitImp () {
    TH1 = 0;                // Обнуление
    TL1 = 0;                // счетчика.
    TMOD = (TMOD & 0x0f) | (Gate1 | mMode16bitForT1);
    // Внешний запуск, 16-разрядный счёт.
    while (Int1);           // Пережидаем случайное попадание на импульс.
    R1 = 1;                 // Разрешаем работу таймера,
                           // но считать он будет при единичном Int0.
    while (!Int1);          // Ждем начало импульса.
```

```
while (Int1);           // Ждем окончание импульса.  
TR1 = 0;              // Останавливаем таймер.  
return (TL1 | (TH1 << 8)); // Возвращаем два байта.  
}
```

Таким образом, можно измерить *внутренней аппаратурой* ОВМ длительность импульса до 65535 мкс. Программа только отслеживает моменты, когда разрешить счет таймера (до начала импульса, заранее) и когда запретить счет (после окончания импульса), т. е. запаздывания в программе на точность измерений не влияют.

В более современных ОВМ [31] для измерения длительности импульсов используется так называемый режим захвата (capture), в котором внешний сигнал управляет не только разрешением счета, но и позволяет в момент окончания импульса копировать содержимое таймера во вспомогательный регистр для последующей обработки.

Приведенная программа является иллюстративной и будет работать неправильно, если длительность импульса выйдет за рамки емкости счета таймера, и таймер переполнится, начиная отсчет с нуля. Чтобы получить правильный результат при измерении длинного импульса, нужно организовать счет количества переполнений в отдельной процедуре прерывания по таймеру.

10.7.3. Счет переполнений таймера при измерении длинных импульсов

Процедура прерывания по переполнению таймера объявляется, как и прежде (см. п. 10.6.4), но при инициализации и в теле процедуры не нужна загрузка, поскольку счет таймера после переполнения продолжается от нулевого значения. Поэтому, кроме сброса бита переполнения, процедура должна инкрементировать значение переменной выбранного типа. Пусть для определенности счет переполнений осуществляется в целочисленной беззнаковой переменной *mIntCount*, ранее инициализированной нулем.

```
void mIntT1 () interrupt TF1_VECTOR {  
    mIntCount ++;  
}
```

Общий вид процедуры измерения остается таким же, как в п. 10.7.2, то есть выполняются инициализация и отслеживание уровней входного сигнала. Отличие заключается в образовании результата, который в данном случае становится 4-байтовым (типа **long**). Поэтому

двухбайтовый результат работы процедуры `mProclzmDlitImp` нужно объединить с двумя байтами счетчика переполнений таймера с учетом их значимости:

$$mLongDlitImp = (long)mProclzmDlitImp | (long)(mIntCount << 16);$$

10.7.4. Использование механизма прерываний для определения начала и окончания импульса

Работа таймера под управлением внешнего сигнала требует, тем не менее, определенных действий и от программы, которая должна отследить начало и окончание измеряемого импульса. То есть процессор во время ожидания не может производить другой полезной работы. Избежать подобного простоя можно было бы при помощи механизма внешнего прерывания, например, при появлении импульса запускается процедура прерывания и разрешается счет таймера, а при окончании сохраняется результат таймера. Однако подобная возможность имеется только в некоторых современных ОВМ (например, в семействе AVR), внешнее прерывание в которых может реагировать не только на отрицательный фронт сигнала, как в семействе x51, но и на положительный.

Вместе с тем реагирование системы прерываний только на один перепад может оказаться полезным при измерении периода импульсов. В этом случае при первом срабатывании процедура прерывания разрешает счет таймера, а при втором счет останавливается и читается результат.

10.7.5. Измерение частоты импульсов

осуществляется двумя основными способами, *прямым* подсчетом импульсов за 1 секунду (см. п. 13.4.1) и *косвенным*, измерением периода импульсов и вычислением обратной величины ($F = 1/T$).

Второй способ отличается высоким быстродействием, но требует соответствующих вычислительных ресурсов от ОВМ (команд или процедур многобайтового деления), рассмотрение которых выходит за рамки настоящего пособия.

Метод прямого счета импульсов был использован ранее в примерах процедуры внешнего прерывания (см. п. 8.5.3 и 10.4.5). Для этой цели можно использовать не программный, а аппаратный счет при помощи внутреннего таймера, если установить бит **С/Т** в «1». Внешние сигналы для счёта должны подаваться на один из выводов **P3_4** (альтернативное имя **T0**) или **P3_5 (T1)**.

Но следует отметить, что, как и ранее, один из таймеров должен отсчитывать калиброванный односекундный интервал, то есть будут задействованы оба таймера ОВМ. И поэтому в подобных случаях желательно использовать ОВМ с тремя таймерами, чтобы иметь возможность, например, управлять частотой последовательного порта.

Пусть для определенности будем использовать таймер 1 для счета внешних импульсов, а таймер 0, как и ранее, для задания односекундного интервала (см. п. 10.6.6). При инициализации следует учесть, что таймер 1 должен считать внешние импульсы. Будем полагать также, что таймер 0, считающий внутренние импульсы был инициализирован ранее (см. п. 10.6.4).

Алгоритм измерения частоты характеризуются последовательными действиями, которые нетрудно проследить по тексту программы:

```
word mProclzmF () {
    TH1 = 0;                // Обнуление
    TL1 = 0;                // таймера.
    TMOD = (TMOD & 0x0f) | mMode16bitForT1 | C_T1;
    // 16-разрядный счёт внешних импульсов в таймере 1.
    while (!mFlag1s);       // Ждем начало 1 с.
    TR1 = 1;                // Разрешаем работу счетчика внешних импульсов.
    mFlag1s = 0;            // Сбрасываем флажок.
    while (!mFlag1s);       // Ждем окончания 1 с.
    TR1 = 0;                // Останавливаем счет внешних импульсов.
    mFlag1s = 0;            // Сбрасываем флажок.
    return (TL1 | (TH1 << 8)); // Возвращаем два байта.
}
```

Программа иллюстративная, в ней, например, отсутствует контроль переполнения таймера 1.

§10.8. Ресурсы последовательного порта

Последовательный порт является, пожалуй, самым важным периферийным внутренним узлом, обеспечивающим удаленную связь с другими микропроцессорными системами. Недаром именно последовательный порт служит стандартным каналом библиотечного ввода-вывода. Но, учитывая ограниченные ресурсы ОВМ, полноформатные функции языка Си обычно в простейших системах не используют, ограничиваясь собственными упрощенными процедурами приема-передачи.

10.8.1. Регистры данных приемника и передатчика

Последовательный порт представляет собой два отдельных устройства – приемник (*Receiver*) и передатчик (*Transmitter*), работающих с асинхронным последовательным форматом данных (см. рис. 8.1) и выполненных на основе байтовых *регистров сдвига*, (см. п.п. 12.6.1 и 12.6.3) что позволяет аппаратно (независимо от программы) выполнять преобразования форматов данных.

Приемник последовательные во времени биты со входа **RxD** (вывод **P3_0**) превращает в параллельный формат. Передатчик выполняет обратное действие – данные параллельной формы передает в последовательном виде на выход **TxD** (вывод **P3_1**).

Регистры *параллельных* данных приемника и передатчика имеют байтовый тип и общее имя **SBUF (Serial Buffer)**, но из регистра приемника можно только *читать* данные, например:

```
myBufR [mPtrR] = SBUF; // Копирование кода в буфер приема.
```

а в регистр передатчика только *записывать* данные:

```
SBUF = myBufT [mPtrT]; // Передача кода из буфера передачи.
```

Однако, при работе с последовательным портом, нужно иметь в виду, что процесс приема-передачи последовательных битов занимает определенное время, зависящее от скорости передачи. Поэтому перед обслуживанием приемника или передатчика следует убедиться в готовности устройства (см. §§10.9 и 10.10).

10.8.2. Регистр управления последовательным портом

Кроме регистров данных последовательный порт имеет *побитно доступный* регистр управления **SCON (Serial Control)**, восемь битов которого условно делятся на три группы.

Первая группа битов задает режимы работы последовательного порта и объединяет четыре бита, **SM0, SM1, SM2** (от *Serial Mode*) и **REN (Receive Enable)**. Они описаны в п. 10.8.3, а работа с ними в п. 10.9.6.

Еще один бит режима (**SMOD** – бит удвоения скорости приема-передачи) расположен вне регистра управления **SCON**, в побитно недоступном регистре управления мощностью **PCON** (см. п. 10.9.5)

Вторая группа битов состоит из двух битов, **TB8 (Transmit Bit 8)** и **RB8 (Receive Bit 8)**, которые являются необязательными 9-ми разря-

дами (**D8**) передатчика и приемника. Они позволяют при необходимости увеличить количество информационных битов в последовательной посылке до 9 (см. далее).

Третья группа также имеет два бита, **RI (Receive Interrupt)** и **TI (Transmit Interrupt)**, которые являются индикаторами готовности приемника и передатчика и используются системой прерываний в качестве запросов на обслуживание. Эти биты устанавливаются автоматически аппаратурой приемника или передатчика. Сбрасываться эти биты должны программно в процессе обработки, как описано в §§10.10 и 10.11.

10.8.3. Биты задания режима работы последовательного порта (**SM0, SM1, SM2, REN**)

Бит разрешения приема **REN** обычно устанавливается программой при инициализации:

REN = 1;

Но иногда, например, чтобы не слушать свою передачу в общем канале связи, приемник отключают, программно сбрасывая этот бит.

Бит **SM2** управляет особым режимом работы приемника – режимом для аппаратного различения в последовательном потоке кодов команд главной удаленной станции от кодов чужих данных. Особенности этого, редко используемого режима рассмотрены ранее (см. п. 8.8.4).

С помощью двух битов **SM0** и **SM1** можно закодировать четыре режима работы последовательного порта.

В режиме 0 (**SM0=0, SM1=0**) реализуется синхронный режим приема-передачи байтов. Передатчик работает в этом режиме, как генератор тактовых импульсов (с выходом **TxD**), а приемник работает (по входу **RxD**) подобно тактируемому регистру сдвига без стартового и стоповых битов, и может передавать или принимать синхронные данные (см. п. 8.8.3).

Режим 1 (**SM0=0, SM1=1**) задает 8-битный асинхронный режим с частотой, определяемой таймером 1.

Режим 2 (**SM0=1, SM1=0**) использует 9-битный асинхронный формат с фиксированной частотой $F_{KB}/64$ или $F_{KB}/32$ в зависимости от бита **SMOD** в регистре **PCON**.

Режим 3 (**SM0=1, SM1=1**) реализует 9-битную асинхронную посылку с частотой, задаваемой таймером 1.

Чтобы не запоминать указанные комбинации битов, в программе можно использовать один из макросов для включения выбранного режима:

```
#define mSetSynModePS          SM0 = 0; SM1 = 0;
#define mSetModePS8bitT1      SM0 = 0; SM1 = 1;
#define mSetModePS9bitFix     SM0 = 1; SM1 = 0;
#define mSetModePS9bitT1      SM0 = 1; SM1 = 1;
```

Но в некоторых случаях удобнее пользоваться байтовыми командами с весами установленных битов режима с учетом месторасположения битов, например:

```
#define mVesSynModePS          0
#define mVesModePS8bitT1      64
#define mVesModePS9bitFix     128
#define mVesModePS9bitT1      128+64
```

Пример инициализации последовательного порта приведен в п. 10.9.6.

10.8.4. Особенности работы с девятым битом

При необходимости он может включаться в асинхронную последовательную посылку в качестве старшего бита данных, бита четности или бита указателя типа посылки (команда/данные).

Передаваемый бит **TB8** можно в программе установить, сбросить или инвертировать:

```
TB8 = 1;
TB8 = 0;
TB8 = ! TB8;
```

Он может также принять вычисленное значение, например, из младшего бита целочисленной переменной (с учетом правил преобразования формата п. 9.6.6):

```
TB8 = mVar & 1;
```

Принимаемый бит **RB8** образуется аппаратно при приеме 9-битного последовательного кода и может копироваться, например, в младший бит целочисленной переменной:

```
mVar = RB8;
```

или анализироваться программой:

```
if (RB8) mProcCmd; else mProcData;
```

§10.9. Инициализация последовательного порта

10.9.1. Общие сведения

При инициализации последовательного порта нужно выполнить ряд действий, затрагивающих сразу три внутренних устройства ОВМ – собственно последовательный порт, таймер 1 для тактирования и систему прерываний. Поэтому указанные действия упрощенно сводятся к инициализации нескольких спецрегистров:

- регистра управления последовательным портом **SCON**;
- регистра режима таймеров **TMOD** и регистра данных **TH1**, если для тактирования порта используется таймер 1, определяющий скорость приема-передачи;
- регистров системы прерываний (разрешения **IE** и приоритета **IP**), если в приемнике или передатчике используется механизм прерываний;
- регистра управления мощностью **PCON**, в котором упакован бит **SMOD** удвоения скорости приема-передачи, если необходимо удвоение скорости.

10.9.2. Задание режима работы последовательного порта битовыми командами

Как отмечалось ранее (см. п. 10.8.2), управление режимами последовательного порта осуществляется 4 битами регистра **SCON** (**SM0**, **SM1**, **SM2**, **REN**).

Поэтому при инициализации необходимо задать (при помощи битов **SM0** и **SM1**) один из четырех возможных режимов работы, например, ранее описанным (см. п. 10.8.3) макросом:

```
mSetModePS8bitT1;
```

Включить приемник:

```
REN = 1;
```

Задать, если нужно, режим аппаратного различения команды в потоке сетевых данных:

```
SM2 = 1;
```

Но при инициализации регистра **SCON** есть один момент, связанный с особенностью архитектуры семейства **x51**. Дело в том, что

при сбросе все биты регистра обнуляются, включая бит готовности передатчика **TI**. Поэтому некоторые процедуры передачи, предварительно проверяющие готовность передатчика (см. п. 10.11.1), могут зависнуть и нарушить тем самым работу всей системы. Чтобы этого не произошло, кроме ранее указанных и очевидных действий с содержимым регистра **SCON** обязательно нужно установить бит готовности передатчика:

TI = 1;

У более современных ОВМ [31] аналогичный бит готовности передатчика устанавливается при сбросе, как и полагается, в единицу.

10.9.3. Задание режима работы последовательного порта байтовой командой

Для минимизации генерируемого кода при инициализации регистра **SCON** можно пользоваться загрузкой байтовой константы с необходимыми установленными битами, например:

SCON = 0x52;

// SM1 = 1; REN = 1; TI = 1;

Однако использование числовой константы в тексте затрудняет в дальнейшем сопровождение программ. Более наглядным приемом является использование именованных констант с весами устанавливаемых битов (см. п. 10.8.3), например:

SCON = mVesModePS8bitT1 | mVesBitREN | mVesBitTI;

10.9.4. Задание скорости и инициализация таймера 1

Скорость приема-передачи последовательного порта может быть фиксированной или регулируемой при помощи таймера 1.

Фиксированная скорость применяется достаточно редко, например, когда нужно освободить таймер 1 для других системных целей, хотя в этом случае целесообразнее перейти на другую модификацию ОВМ с большим числом таймеров.

Если в регистре **SCON** задан режим тактирования от таймера 1, то последний автоматически подключается к последовательному порту, определяя период приема-передачи битов. Однако при инициализации обязательно нужно задать режим работы таймера 1 с перезагрузкой и занести байтовую константу для выбранной скорости в его регистры данных **TH1** и **TL1**.

При задании режима работы таймера 1 следует иметь в виду, что в тетрадах регистра **TMOD** упакованы биты двух таймеров (см. п. 10.5.2). Поэтому инициализацию таймера 1 нужно производить, не затрагивая младшей тетрады, поскольку другой модуль программы мог уже инициализировать биты таймера 0. Поэтому будет более корректно (см. п. 10.5.4), прочитав текущее значение регистра **TMOD**, сохранить биты таймера 0, далее обнулить старшую тетраду с битами таймера 1 и установить новые биты таймера 1, например:

TMOD = (TMOD & 0x0f) | mMode8bitReloadForT1;

Код загрузки регистра **TH1**, который должен обеспечить стандартное значение частоты приема-передачи (например, 1200 Гц, 2400 Гц и т. д.), обычно выбирается из формул и таблиц, приводимых в фирменной документации и литературе. Нужно только помнить, что код зависит не только от требуемой стандартной скорости, но и от частоты внешнего кварца (см. п. 8.9.2). Если нужного кварца нет, то возможны некоторые вариации кода в пределах допустимой погрешности частоты (см. п. 8.9.3).

Для упрощения использования в программах эти начальные коды загрузки можно записать в виде именованных констант для разных скоростей и частот кварца и использовать по мере надобности. Например, ниже приведены константы загрузки для кварца с частотой 11059 кГц и стандартного ряда частот от 1200 Гц до 9600 Гц:

```
#define mTH1for1200kv11059      0xE8
#define mTH1for2400kv11059      0xF4
#define mTH1for4800kv11059      0xFA
#define mTH1for9600kv11059      0xFD
```

Тогда оператор, определяющий скорость, будет более наглядным, например:

TH1 = TL1 = mTH1for9600kv11059;

И в заключение нужно не забыть запустить таймер:

TR1 = 1;

Пример инициализации приведен в п. 10.9.6.

10.9.5. Удвоение скорости приема-передачи

В некоторых ситуациях полезной особенностью последовательного порта является возможность удвоения выбранной скорости работы при помощи бита **SMOD** в регистре **PCON** (рис. 10.10).

D7	PCON (побитно недоступен!)					D0	
SMOD	-	-	-	GF1	GF0	PD	IDL

Рис. 10.10

Этот регистр *недоступен для битовых команд*, поэтому манипуляции указанным битом нужно проводить с использованием логических операций и байтовых констант. Например, установка бита для удвоения скорости осуществляется при помощи операции «ИЛИ» и именованной константы, учитывающей положение или вес бита в регистре:

PCON = PCON | mVesBitSMOD;

В некоторых случаях используют и очистку указанного бита с помощью операции «И»:

PCON = PCON & mVesBitSMOD;

Примечание: В синхронном режиме (под номером 0) бит удвоения **SMOD** не работает.

10.9.6. Пример инициализации последовательного порта

С учетом изложенных пояснений можно объединить выше рассмотренные операторы инициализации в единую процедуру:

```
void mInitPS (void) {
    SCON = mVesModePS8bitT1 | mVesBitREN | mVesBitTI;
    TR1 = 0; // Останов таймера T1.
    TMOD = (TMOD & 0x0f) | mMode8bitReloadForT1;
    TH1 = TL1 = mTH1for9600kv11059; // Задана частота обмена
    TR1 = 1; // Запуск таймера T1.
    PCON = PCON | mVesBitSMOD; // Удвоение скорости, если нужно.
    ES = 1; // Разрешить прерывание
    EA = 1; // при необходимости).
}
```

§10.10. Процедуры приема для последовательного порта

10.10.1. Программное ожидание посылки

Если прерывание от последовательного порта не используется, то перед чтением регистра приемника нужно в цикле ждать принятия новой

порции данных, анализируя флажок **RI**. После этого прочитать данные и сбросить флажок во избежание повторного чтения. Ниже приведена простейшая процедура для подобного программного приема:

```
byte mGetChar () {
    while (!RI);           // Ожидать приема байта.
    RI = 0;                 // Флаг приема сбросить.
    return SBUF;           // Полученный байт вернуть.
}
```

Данная процедура может зависнуть, не дождавшись посылки от удаленного абонента. Для исключения этой ситуации можно организовать аварийный выход из процедуры по так называемому «тайм-ауту». Для этого в операторе **while** обычно производят счет циклов опроса признака **RI** и сравнение с предельным значением. При успешном выходе устанавливается флажок – «байт принят». При выходе из цикла по «тайм-ауту» флажок сбрасывается. Но более удобно и логично обслуживать приемник с привлечением механизма прерываний, потому что чужая посылка обычно является неожиданной (асинхронной) для программы, которая на ожидание и повторные опросы может потратить много времени.

10.10.2. Прием по прерыванию одного байта

Особенностью системы прерывания ОВМ x51 для последовательного порта является то, что один вектор прерываний используется для двух устройств – приемника и передатчика. Поэтому процедура прерывания для них должна быть единой и различать, какое из устройств требует обслуживания.

Для этого в самом начале процедуры прерывания, обязательно должен быть условный оператор, обеспечивающий разделение фрагментов программы, отвечающих за прием и передачу. Например, в нижеприведенной программе осуществляется только прием, и игнорируются запросы передатчика.

Кроме того, поскольку прерывающая процедура не может возвращать значение, поэтому прием должен происходить в заранее определенную буферную переменную или массив.

```
void mProcIntPS1() interrupt SI0_VECTOR {
    if ( RI ) {             // Приемник запросил обслуживание?
        RI = 0;               // Да, сброс запроса.
        mBufR = SBUF;        // Читаем принятый байт.
        mFlagReadyR = 1;     // Флажок "байт принят".
    } else {                 // Передатчик не обслуживаем,
        TI = 0;              // но запрос передатчика сбрасываем,
```

```

    mFlagReadyT = 1; // а свой флажок устанавливаем.
}
}

```

*Заметим, что при приеме байта одновременно устанавливается пользовательский флажок **mFlagReadyR**, который сообщает главной программе или ее процедурам о том, что имеется новая порция информации, требующая обработки.*

Кроме того нужно обратить внимание, что передатчик может не обслуживаться процедурой прерывания, но запрос передатчика должен быть сброшен, чтобы не вызвать повторного ложного вызова прерывания. В этом случае возникает двусмысленная ситуация, когда передатчик после окончания посылки был готов и установил **TI**, но программа прерывания этот признак готовности сбросила, а другие процедуры могут напрасно ждать готовности передатчика. Чтобы исключить это противоречие, можно воспользоваться своим флажком готовности передатчика **mFlagReadyT**, который дублирует признак **TI**, но не будет вызывать прерывания, и который можно использовать в программах для определения готовности передатчика. Можно также дожидаться окончания посылки и сбрасывать бит **TI** (см. п. 10.11.2), но при этом процессор не будет производить полезной работы.

Кстати, в большинстве современных процессоров приемник и передатчик имеют отдельные векторы прерывания (и процедур), и подобная проблема не возникает [31].

10.10.3. Прием по прерыванию заданного количества байтов

Размер пакета с данными может быть заранее определен в константе **mBufSize** или вычислен по ходу работы в соответствующей переменной.

В этом случае пользуются счетчиком числа принятых байтов – своеобразным указателем на очередное место в буфере **mBufR [mBufSize]**, выделенным заранее. Когда буфер заполнен, это означает, что весь пакет принят и его можно обрабатывать:

```

void mProclntPS2 () interrupt SI0_VECTOR {
    if( RI ) { // Приемник запросил обслуживание?
        RI = 0; // Очистка запроса
        mBufR [mPtrR++] = SBUF; // Прием в буфер по указателю.
        if( mPtrR == mBufSize ) { // Весь пакет принят?
            mFlagReadyPaket = 1; // Флажок "пакет принят".
        }
    }
}

```

```
    } else { TI = 0; mFlagReadyT = 1; } // Передатчик не обслуживаем
  }
```

Заметим, что счетчик *mPtrR* перед началом приема пакета должен быть обнулен.

При необходимости можно организовать безадресную форму приема в буфер со сдвигом (см. п. 9.9.2).

10.10.4. Прием строки символов по прерыванию

Некоторые пакеты не имеют определенного размера и передаются с конечным ограничителем, например, модем может отвечать на команды числовой строкой, оканчивающейся кодом «13» – «возврат каретки», по которому можно определить завершение порции данных. В этом случае принятый байт нужно наряду с записью в буфер проанализировать на равенство завершающему коду *mKodEnd*:

```
void mProcIntPS3 () interrupt SIO_VECTOR {
  if ( RI ) {                                     // Приемник запросил обслуживание?
    RI = 0;                                       // Очистка запроса.
    mBufR [mPtrR++] = SBUF;                       // Прием.
    if (SBUF == mKodEnd) mFlagReadyPaket = 1; // Пакет принят?
  } else { TI = 0; mFlagReadyT = 1; } // Передатчик не обслуживаем.
}
```

Реальные программы наряду с проверкой завершающего кода одновременно осуществляют контроль выхода за границы выделенного буфера.

10.10.5. Примеры простейшей обработки принятой информации

Одиночный байт можно выводить для контроля на индикацию, например, при тестировании последовательного канала:

```
if (mFlagReadyR) mByteToBCDToBufInd (mBufR);
```

использовать в качестве системного параметра:

```
if (mFlagReadyR) mSysVar = mBufR;
```

или анализировать:

```
if (mFlagReadyR) {if (mBufR==mKodCmd) mProcCmd;}
```

Принятый пакет обычно имеет оговоренный формат, который используется при распаковке на отдельные составляющие, например, адрес, команду и данные:

```

if (mFlagPaketReady) {
    mAdr = mBufR [0];           // Код адреса устройства.
    mCmd = mBufR [1];           // Код команды.
    mData = mBufR [2];          // Код данных.
}

```

§10.11. Процедуры передачи для последовательного порта

10.11.1. Передача одного байта без прерывания

Информация для передачи вычисляется по ходу программы или является ответом на ранее принятую, то есть является ожидаемой для программы. Поэтому использовать прерывания для передачи в большинстве случаев необязательно.

Программа передачи байта должна дожидаться готовности передатчика, отслеживая бит **TI**, а затем обнулить используемый флажок и записать байт в регистр данных:

```

void mSendByte (byte mDat) {
    while (!TI);           // Ждем завершения предыдущей
    TI = 0;                 // передачи и сбрасываем флажок.
    SBUF = mDat;            // Передача началась.
}

```

Если используется прием по прерыванию, то вместо бита **TI** можно проверять и сбрасывать свой флажок **mFlagReadyT** (см. п. 10.10.2).

10.11.2. Передача байта с ожиданием окончания посылки

При работе на общий канал связи после передачи своего байта нужно освободить канал. Для этого нужно дождаться окончания последовательной посылки и отключиться от линии связи:

```

void mSendByteW (byte mDat) {
    mPinTxEn = 1;           // Подключаемся к каналу связи.
    SBUF = mDat;            // Передача началась.
    while (!TI);           // Ждем окончания посылки.
    TI = 0;                 // Сбрасываем флаг готовности
    mPinTxEn = 0;          // Отключаемся от канала связи.
}

```

Заметим, что предварительная проверка готовности передатчика в этой программе не производится, так как при предыдущем вызове этой процедуры окончание асинхронной посылки было отслежено.

10.11.3. Передача пакета данных без прерывания

Пусть данные для передачи уже подготовлены в массиве *mBufT* размером *mPtrMax*:

```
void mSendPaket () {
    byte mPtrT;
    for (mPtrT = 0; mPtrT < mPtrMax; mPtrT++) {
        SBUF = mBufT [mPtrT];
        while (!TI);           // Ждем окончания последовательной посылки.
        TI = 0;                 // Сбрасываем флаг готовности
    }
}
```

Поскольку последовательная передача битов занимает определенное время, то программе приходится ожидать окончания передачи каждого байта, что не позволяет процессору производить другую работу. В подобном случае лучше использовать механизм прерывания.

10.11.4. Передача пакета по прерыванию

Процедуры прерывания не могут принимать параметры, поэтому пакет, как и прежде, должен быть подготовлен в буфере:

```
void mIntPS4 () interrupt SIO_VECTOR {
    if( RI ) { RI = 0; ... }    // Здесь обслуживаем приемник
    else {
        TI = 0;                // Очищаем запрос.
        if (mPtrT < mPtrMax) SBUF = mBufT [mPtrT++];
        // Передаем?
        // Не ожидая окончания посылки, завершаем процедуру.
    }
}
```

Заметим, что при системном сбросе или после передачи предыдущего пакета бит готовности *TI* будет обнулен, поэтому чтобы первый раз вызвать прерывание, передачу пакета можно начать вне процедуры прерывания командой засылки нулевого элемента массива в регистр данных передатчика и установкой начального значения переменной *mPtrT*:


```
SBUF = mBufT [0];
mPtrT = 1;
```

Следующий элемент массива будет передан процедурой прерывания после окончания асинхронной отправки и установки бита **TI**. Инициализация процедуры прерывания должна быть произведена заранее (см. п. 10.7.6).

При таком способе передачи окончание каждой отправки не отслеживается, и процессор в это время может выполнять необходимую работу, только временами прерываясь на отсылку очередного байта пакета в регистр данных передатчика.

Контроль состояния передачи пакета можно производить, анализируя значение переменной *mPtrT* или вводя дополнительный флажок (см. далее).

10.11.5. Передача строки символов по прерыванию

Размер символьной строки может изменяться по ходу программы, поэтому контроль окончания передачи пакета обычно производят по конечному коду *mKodEnd*, например, «возврат каретки» (код 13), завершающему строку. Получив конечный код, программа передает его и устанавливает флажок *mFlagPaketEnd*, означающий, что пакет передан:

```
void mIntPS5 () interrupt SI0_VECTOR {
    if ( RI ) { RI = 0; {...} }           // Здесь обслуживаем приемник
    else {
        TI = 0;                           // Очищаем запрос.
        if (!mFlagPaketEnd) {             // Все передано?
            SBUF = mBufT [mPtrT];
            if (mBufT [mPtrT++]==mKodEnd) mFlagPaketEnd = 1;
        }
    }
}
```

Первый байт пакета посылают вне процедуры прерывания и сопровождают сбросом флажка:

```
SBUF = mBufT [0];
mPtrT = 1;
mFlagPaketEnd = 0;
```

10.11.6. Примеры подготовки пакета для передачи

Если пакет имеет константные значения, то его можно оформить в кодовой области, например, подобно команде отключения эхо для модема:

```
code byte mMasT [ ] = {'ATE0',13};
```

Если составляющие пакета вычисляются по ходу программы, то в массив копируют необходимые переменные:

```
mBufT[0] = mAdr;           // Код адреса устройства.  
mBufT[1] = mKmd;          // Код команды.  
mBufT[2] = mData;         // Код данных.
```

11

КОМБИНАЦИОННЫЕ УЗЛЫ

Ниже рассмотрены основные особенности наиболее распространенных на практике базовых цифровых устройств, реализованных на логических элементах без памяти.

§11.1. Дешифраторы

11.1.1. Общие сведения

Дешифратор (*decoder*) выполняет преобразование логических сигналов – цифровой код на входе устройства преобразуется в выходной код, реализованный по другому правилу. Поэтому в некоторых случаях термином «дешифратор» именуют любой преобразователь кодов. Примером такого расширенного толкования является преобразователь двоичного кода в код управления семисегментным индикатором (см. §11.3).

Но чаще всего дешифратором называют комбинационную логическую схему, преобразующую входной двоичный код в **УНАРНЫЙ** («один из многих») [23]. В этом случае активный сигнал («0» или «1») присутствует только на одном из выходов, порядковый номер которого соответствует значению входного кода. На остальных выходах дешифратора будут пассивные сигналы.

Различают *полные* и *неполные* дешифраторы. Полные дешифраторы имеют количество выходов 2^N , где N – число входных разрядов. Например, если входов 3, то выходов $2^3 = 8$.

В некоторых случаях полный дешифратор может оказаться избыточным. Например, при построении устройств, связанных с микропроцессорной шиной, схема должна откликаться только на несколько своих номеров, включая базовую адресную двоичную комбинацию и смежные. Очевидно, что в этом случае применять полный дешифратор нецелесообразно, так как даже при

16-разрядном адресе количество выходов у полного дешифратора равно 65536, тогда как необходимы только несколько выходов (см. п. 11.2.5).

11.1.2. Иллюстративная модель дешифратора

Для наглядности дешифратор можно представить в виде контактного переключателя, который формирует активный сигнал только на одном из выходов (рис. 11.1), номер которого определяется входным кодом X_{BX} .

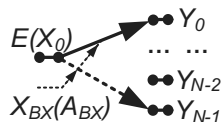


Рис. 11.1

Сигнал E является для дешифратора управляющим, то есть если он активен, то дешифратор работает и один из его выходов будет выбран. В противном случае, когда E пассивен, все выходы будут находиться в пассивном состоянии, и дешифратор работать не будет.

Такой переключатель иногда относят к одному из типов коммутаторов данных под названием демультиплексор (см. п. 11.8.1). Только информационным для коммутатора будет сигнал X_0 , а управляющим – адресный сигнал A_{BX} , показанные на рисунке в скобках. То есть при одинаковой структуре дешифратора и демультиплексора входной и управляющий сигналы меняются местами.

11.1.3. Таблица и уравнения для простейшего дешифратора

Логические функции, как известно [23], могут определяться формулами, таблицей и другими способами. В качестве примера для простейшего двухразрядного дешифратора в табл. 11.1 приведены значения входных и выходных сигналов.

Таблица 11.1

Входы			Выходы			
X_0	X_1	№	Y_0	Y_1	Y_2	Y_3
0	0	Y_0	1	0	0	0
0	1	Y_1	0	1	0	0
1	0	Y_2	0	0	1	0
1	1	Y_3	0	0	0	1

Два входных разряда X_0 и X_1 имеют четыре возможные комбинации N_k и, соответственно, четыре выходных сигнала от Y_0 до Y_3 . Причем активным будет выход, номер которого совпадает со значением двоичного кода на входе. Например, нулевому коду соответствует активный уровень на выходе Y_0 , тогда как остальные выходы будут находиться в пассивном состоянии.

Для каждого из выходов на основе этой таблицы можно записать логические уравнения. Проще всего это сделать для выхода Y_3 , который единичен, когда оба входных сигнала (и X_1 , и X_0) единичны. Из этого следует, что для реализации выходного сигнала Y_3 необходима функция логического умножения «И»:

$$Y_3 = X_0 X_1, \quad (11.1)$$

$$Y_0 = \bar{X}_0 \cdot \bar{X}_1; \quad (11.2)$$

$$Y_1 = \bar{X}_0 \cdot X_1; \quad (11.3)$$

$$Y_2 = X_0 \cdot \bar{X}_1. \quad (11.4)$$

Далее нетрудно заметить, что и для остальных выходов можно также использовать функцию «И» с учетом простого правила – если во входном двоичном коде записан ноль, то в уравнении для выхода нужно использовать инверсное значение переменной. Если единица, то – прямое.

Из уравнений видно, что двухразрядный дешифратор можно реализовать на основе 4 элементов логического умножения «И», причем для каждого выхода используется свой элемент.

Кроме того, надо отметить, что по принципу **дуальности** [23] дешифратор можно реализовать и на основе логических элементов «ИЛИ», только значения выходных и входных величин будут при этом инверсными по сравнению со значениями в базисе «И».

11.1.4. Схема линейного дешифратора

в соответствии с уравнениями (11.1)–(11.4) представляет набор (линейку) схем логических умножения, приведенную на рис. 11.2. Инверсные значения входных переменных образуются при помощи вспомогательных инверторов.

В большинстве микросхем дешифраторов используются элементы «И-НЕ» (см. п. 2.3.2), поэтому выходные сигналы таких схем дополнительно инвертированы и выбранный выход имеет в этом случае нулевое (активное) значение, а все остальные – единичное (пассивное). Такие выходы хорошо согласуются с ТТЛ стандартом, в котором при нулевом (активном) сигнале обеспечивается большой ток в нагрузке.

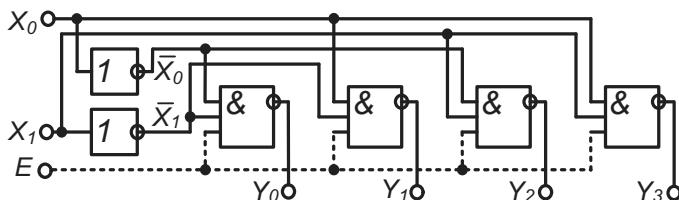


Рис. 11.2

Аналогично рассмотренному можно реализовать многоразрядный дешифратор, используя наборы логических элементов с большим числом входов. Причем нетрудно заметить, что при увеличении числа разрядов увеличивается число выходов и соответственно количество элементов в линейке. Но главным ограничением является то, что число входов логического элемента ограничено технологическими причинами. Поэтому при реализации много-разрядных дешифраторов используют, как правило, каскадный и матричный принципы построения (см. п.п. 11.1.6 и 11.1.7).

Кроме информационных входов, дешифратор имеет один или несколько входов разрешения или стробирования. На рис. 11.2 таким входом является вход *E* (от *Enable* – разрешить), который подается на все схемы совпадения (см. пунктирную линию на рисунке).

Если на входе *E* будет единичный уровень сигнала, то работа дешифратора разрешена (только один выход будет активным). В противном случае на всех выходах будет пассивный уровень, и дешифратор не будет воспринимать и преобразовывать входной код.

Управление дешифратором может понадобиться, например, при борьбе с ложными импульсами, помехами (см. п. 13.4.2). Вход управления позволяет также наращивать разрядность дешифратора, используя имеющиеся микросхемы малой разрядности (см. п. 11.1.6).

11.1.5. УГО и микросхемы дешифраторов

Вне зависимости от реализации дешифраторов в принципиальных схемах используют их общее условное обозначение. В качестве примера на рис. 11.3 приведено УГО для микросхемы трехразрядного дешифратора К555ИД7. Символ функции дешифратора *DC* (от *DeCoder*) указывается обычно в центре прямоугольника.

Выходы всегда располагаются справа, их обозначают чаще всего для простоты только цифрами, которые соответствуют порядковому номеру выхода, то есть

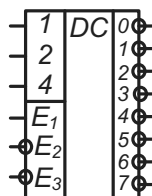


Рис. 11.3

десятичному номеру кода, на который откликается выход. Обратим внимание на то, выходы помечены кружком, то есть являются инверсными (активный уровень нулевой).

Входы расположены слева и могут иметь в обозначении, наряду с цифрами, буквы и нумероваться по порядку, как в табл. 1, или обозначаться весами двоичных разрядов (1, 2, 4), как на рис. 11.3.

Кроме информационных входов, которые указывают номер выбранного выхода, дешифратор может иметь один или несколько входов *разрешения*, обозначаемых буквами *E* или *V* (от *Valve* – клапан). На рис. 11.3 таких входов три, причем два из них инверсные, а один – прямой. Все они объединяются на внутренней логической схеме «И», которая вырабатывает общий сигнал *E*, управляющий работой дешифратора (как показано на рис. 11.2). Поэтому, чтобы разрешить работу дешифратора, нужно подать нулевые сигналы на инверсные управляющие входы и единичный на прямой вход.

На практике широко используются и другие микросхемы, например, К155ИД3 (четырёхразрядный дешифратор с 16 выходами), К155ИД10 (дешифратор двоично-десятичного кода) и подобные.

Отметим, что микросхема К155ИД10 является четырёхразрядным неполным дешифратором, реагирующим на значения от 0 до 9 (см. п. 11.4.1). Комбинации, которые не используются, могут подаваться на входы, но на выходе при этом активных сигналов не будет.

Разрядность микросхем дешифраторов обычно не превышает четырех. Для увеличения разрядности микросхемы используют каскадное включение (см. далее).

11.1.6. Каскадное соединение дешифраторов

При построении линейных дешифраторов с большим числом входов возникают проблемы, связанные с тем, что многовходовые элементы «И» технологически не выгодны. В этом случае можно использовать так называемый каскадный дешифратор.

В качестве примера рассмотрим реализацию 4-разрядного дешифратора на основе уже имеющихся 2-разрядных дешифраторов (рис. 11.4). Для этого входной код разбивают на 2 группы (X_0, X_1 и X_2, X_3), которые будут обрабатываться двумя каскадами дешифраторов. Старшая часть кода (X_2, X_3) обрабатывается управляющим дешифратором $ДС_4$, который формирует управляющие сигналы для каждого из дешифраторов младшего каскада.

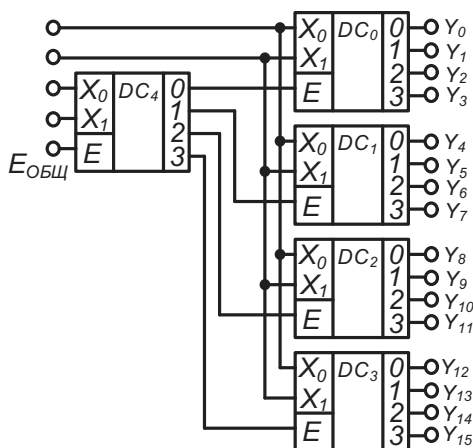


Рис. 11.4

Например, для нулевых значений X_2 и X_3 активным будет нулевой выход дешифратора DC_4 и будет разрешена работа DC_0 . Следовательно входной код, зависящий от комбинации X_0, X_1 , будет расшифрован и на одном из выходов $Y_0..Y_3$ появится активный сигнал. Остальные дешифраторы и их выходы будут находиться в пассивном состоянии.

Если старшая часть кода изменится, то по новому значению X_2, X_3 будет выбран соответствующий дешифратор и сигналы X_1 и X_0 будут управлять другими выходами каскадной схемы.

Нетрудно заметить, что быстродействие каскадной схемы хуже, чем линейной, так как информационные сигналы старшей группы последовательно распространяются через каскады, и время задержки выходных сигналов относительно входных соответственно увеличивается. Если в линейном дешифраторе задержка (без учета инверторов) определяется только одной схемой «И», то при наличии двух каскадов быстродействие в 2 раза хуже.

Каскадное соединение в неявном виде образуется при обслуживании микросхем памяти в микропроцессорных системах (см. п. 11.2.4) за счет дешифратора банка памяти (первая ступень) и встроенных дешифраторов запоминающих ячеек (вторая ступень).

Каскадный дешифратор можно реализовать и с большим числом каскадов, при этом входной код будет разбит на множество групп, но при такой структуре возрастут и задержки.

11.1.7. Прямоугольный (матричный) дешифратор

Другим, кроме каскадного, способом построения дешифратора на основе маловходовых элементов «И» является матричная организация схемы дешифратора с разделением на строки и столбцы.

Такая структура широко используется при адресации ячеек внутри микросхем памяти (см. п. 14.1.2).

Основой матричной схемы (рис. 11.5) являются два управляющих дешифратора (с прямыми выходами). Один анализирует старшие разряды входного кода, другой – младшие. Выходы этих дешифраторов подаются на входы матрицы, состоящей из 2-входовых элементов «И», причем один из дешифраторов управляет сигналами строк матрицы, другой – столбцов.

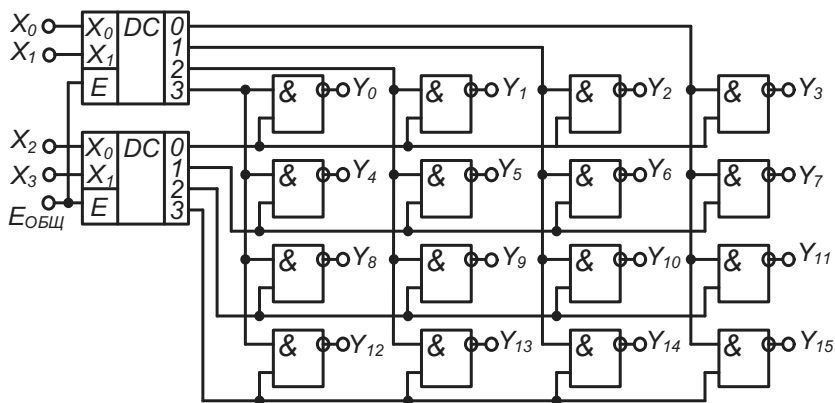


Рис. 11.5

У каждого управляющего дешифратора будет выбран только один выход, соответствующий части входного кода. Поэтому активный сигнал будет присутствовать только на одной строке и только на одном столбце, и сработает только один из 16 логических элементов «И», расположенный на пересечении выбранных столбца и строки. То есть нетрудно видеть, что данная схема будет выполнять функции 4-разрядного дешифратора.

Матричный дешифратор можно реализовать и с большим числом выходов, причем независимо от разрядности в матрице будут использоваться только двухвходовые элементы «И», что существенно упрощает схему и оказывается весьма полезным при реализации микросхем памяти (см. п. 14.1.2).

Матричный принцип, который в свое время широко использовался в вычислительной технике, не потерял своего значения и в настоящее время. например, матрицы элементов используются в программируемых логических микросхемах в виде наборов логических схем, программируя которые, то есть, устанавливая необходимые связи, можно получать требуемые комбинационные устройства (см. п. 14.3.3).

§11.2. Применение микросхем дешифраторов

Дешифраторы являются важной частью многих других узлов, входят в состав процессора и микросхем памяти (см. п. 14.1.2), используются в аналоговых и цифровых коммутаторах (см. п.п. 11.8.3 и 15.2.1), в схемах индикации данных (см. §11.3) и в распределителях импульсов (см. п. 13.4.2). В данном параграфе описаны лишь некоторые из возможных практических применений микросхем дешифраторов.

11.2.1. Дешифратор в микропроцессорной системе

На практике бывает необходимо включать в каком-то порядке одно из нескольких устройств, например, знакоместо в индикаторе (см. п. 11.4.1), фазу шагового двигателя, лампы иллюминации и т. д. В качестве примера особенностей использования микросхемы дешифратора в подобных случаях рассмотрим устройство для индикации одного из возможных режимов работы системы на основе линейного светодиодного индикатора (рис. 11.6).

Из схемы видно, что внешний дешифратор для управления 8-ю светодиодами позволяет использовать только три вывода ОВМ,

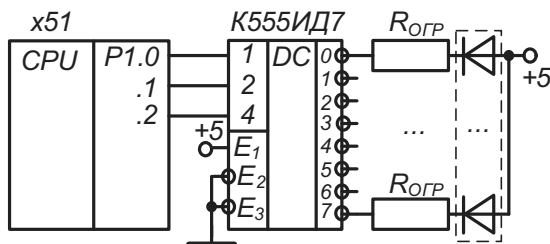


Рис. 11.6

то есть экономятся ее ресурсы. Кроме того дешифратор в большинстве случаев может коммутировать значительные токи в нагрузку, то есть выполнять вдобавок усилительные функции.

Пусть программа в переменной *mMode* получает двоичный номер режима, который необходимо индцировать, засветив соответствующий светодиод. Нужно установить этот номер в 3 младших битах порта *P1*, *сохранив* остальные биты. Простое копирование байтовой переменной в порт командой ассемблера

Mov P1, mBufInd

или оператором Си

P1 = mBufInd;

будет разрушать (обнулять) старшие биты порта, которые могут использоваться в системе для других целей, например, управлять двигателем или нагревателем. Поэтому прежние значения старших битов нужно обязательно сохранить.

Для этого необходимо предварительно читать все биты порта *P1*, очистить 3 младших бита, к которым подключен дешифратор, и поместить данные на их место при помощи операции логического сложения. На ассемблере подобный фрагмент может быть записан в следующем виде:

Mov A, P1	; Читать все биты порта <i>P1</i> .
Anl A, #11111000b	; Обнулить три младших бита.
Orl A, mMode	; Поместить номер режима.
Mov P1, A	; Вернуть <i>P1</i> с измененными битами

На языке Си тот же фрагмент будет записан короче:

P1 = (P1 & 0xF8) | mMode;

11.2.2. Программная реализация дешифратора

В некоторых случаях при наличии свободных выводов у ОВМ можно упростить схему (рис. 11.7), исключив внешний дешифратор и используя программу, в которой для большей наглядности применен оператор переключения **switch** [21], структура которого весьма похожа на иллюстративную модель дешифратора (см. рис. 11.1):

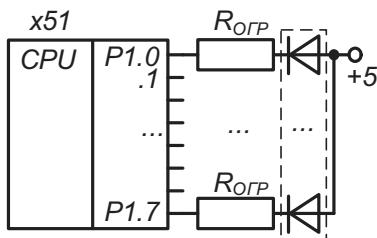


Рис. 11.7

```

switch (mMode) {
    case 0: P1 = ~1; break;
    case 1: P1 = ~2; break;
    case 2: P1 = ~4; break;
    case 3: P1 = ~8; break;
    case 4: P1 = ~16; break;
    case 5: P1 = ~32; break;
    case 6: P1 = ~64; break;
    case 7: P1 = ~128; break;
}

```

Нетрудно заметить, что значения выходных сигналов представлены весами установленных двоичных разрядов, а нулевой активный уровень обеспечивается операцией поразрядного отрицания “~”.

Однако на практике для программной реализации дешифратора пользуются более компактным фрагментом на основе оператора сдвига:

$P1 = \sim(1 \ll mMode);$

то есть единица сдвигается влево на место, соответствующее входной переменной, и инвертируется.

Отметим, что программа усложняется, если используются только часть битов порта, потому что незадействованные биты нужно сохранять, подобно тому, как это сделано в предыдущем пункте.

11.2.3. Реализация табличной функции

Пусть задана некая табличная функция с тремя входными переменными a , b , p и двумя выходными. Для примера рассмотрим табл. 11.2 для реализации полного сумматора (см. п. 11.9.3), который имеет два выходных сигнала, сумму S и перенос P_+ .

Таблица 11.2

Входы				Выходы	
№	a	b	p	S	P ₊
0	0	0	0	0	0
1	0	1	0	1	0
2	1	0	0	1	0
3	1	1	0	0	1
4	0	0	1	1	0
5	0	1	1	0	1
6	1	0	1	0	1
7	1	1	1	1	1

Для трех входных сигналов, имеющих *восемь* различных комбинаций, будем использовать трехразрядный дешифратор с восемью выходами.

Заметим далее, что выход S будет единичен, когда на входе действует **или** первая комбинация, **или** вторая, **или** четвертая, **или** седьмая. То есть вдобавок к дешифратору, различающему входные комбинации, нужен элемент «ИЛИ» (рис. 11.8), на входы которого поданы выходы дешифратора, соответствующие перечисленным комбинациям. Аналогично для выхода P_+ используется свой элемент «ИЛИ», показанный на рисунке пунктиром. Входы для него проставить самостоятельно с учетом значений таблицы.

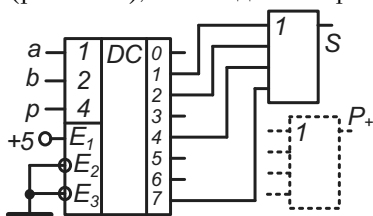


Рис. 11.8

Дешифратор формирует все возможные комбинации для входных переменных, распайкой (переключением) выводов дешифратора задаем значения выходного сигнала для каждой комбинации, и объединяющая схема «ИЛИ» формирует итоговый выходной сигнал. Подобная схема, состоящая из дешифратора и переключателя («распайки» и «объединителя» см. п. 11.8.3), является первоосновой программируемого комбинационного элемента (см. п.п. 11.8.7 и 14.3.2).

11.2.4. Дешифратор для управления банками памяти

Микросхемы памяти имеют ограниченную емкость, поэтому системы памяти микропроцессорных систем выполняют в виде набора микросхем (банков памяти). Все банки памяти имеют общую информационную магистраль, то есть все одноименные выводы данных подключаются к одним и тем же информационным выводам шины данных ШД. Такое общее использование магистрали может привести к нежелательной конкуренции, когда, например, два банка одновременно выставят свои данные для процессора. Подобная ситуация нарушает работу системы, поэтому для исключения конфликта применяют дешифраторы зоны (банка) памяти.

Предположим для примера, что микропроцессорная система имеет 16 битов адреса, то есть всего адресуются $2^{16} = 65536$ ячеек памяти. А в наличии имеются микросхемы памяти с ограниченной емкостью 8 кбайт. Следовательно, все адресное пространство 64 кбайта надо разделить на 8 банков по 8 кбайт каждый (рис. 11.9).



Рис. 11.9

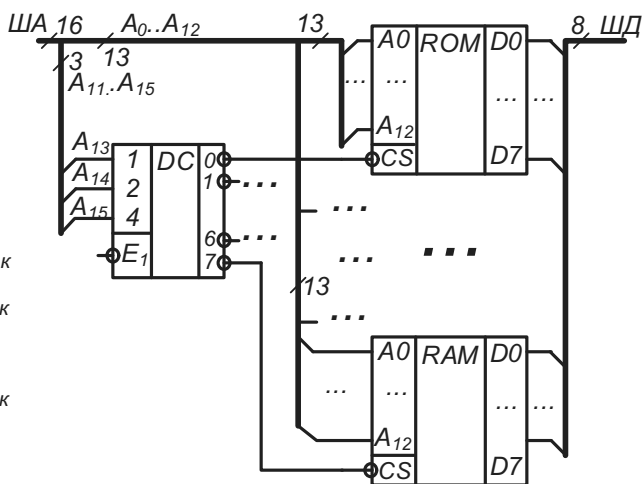


Рис. 11.10

Для этого адресные биты делятся на две группы: старшую и младшую. Старшие 3 бита подаются на дешифратор зоны (рис. 11.10), который разрешает работу одной из 8-ми микросхем по входу выбора микросхемы *CS* (*Chip Select*). Обязательно нужно, чтобы тип выхода дешифратора (прямой или инверсный) совпадал с типом входа *CS*. Некоторые выводы микросхем памяти на рисунке для простоты не показаны (см. п. 14.1.1).

Банки памяти могут быть выполнены как виде ОЗУ (для чтения и записи), так и ПЗУ, где хранятся редко изменяемые и системные процедуры. Особенности подключения микросхем памяти к микропроцессорной магистрали описаны в §14.5.

Процессор, обращаясь к ячейке памяти, устанавливает ее номер на шине адреса ША. Дешифратор различает зону памяти (старшую часть адреса) и разрешает работу только одной микросхеме, несмотря на то, что младшие адреса поданы параллельно на все микросхемы.

Иногда дешифратор банка памяти включают в состав БИС, подобным примером может служить процессор Intel 80386EX.

11.2.5. Дешифратор адреса для внешних устройств

В компьютере кроме памяти имеется множество так называемых внешних устройств (ВУ). Каждое ВУ имеет несколько внутренних

регистров, которым присвоены уникальные системные адреса. И соответственно ВУ должно иметь в своем составе дешифратор с несколькими выходами, реагирующими на свой двоичный адрес.

Подобные дешифраторы являются неполными и реализуются чаще всего по каскадной схеме. Первый каскад анализирует старшие биты ША и выделяет базовый адрес, который разрешает работу второго (младшего) дешифратора с несколькими выходами, реагирующими на младшие биты адреса.

В качестве примера рассмотрим дешифратор для двухбайтового адреса $0300h$, который можно использовать для вновь разрабатываемых вставных модулей компьютера [15].

Для простоты разбиваем адресный код на две байтовые группы, старшая ($A_{15}..A_8$) заданного кода в двоичном виде записывается как 00000011, а младшая ($A_7..A_0$) – 00000000. Используя правило, описанное при построении линейного дешифратора в п. 11.1.3, можно реализовать два неполных дешифратора (рис. 11.11).

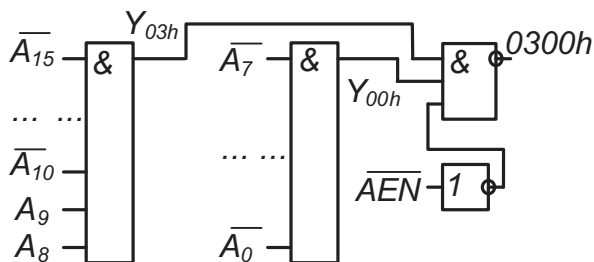


Рис. 11.11

Один, реагирующий на комбинацию $03h$ (для этого на вход элемента «И» поступают прямые сигналы A_8 и A_9 , остальные инверсные). Второй дешифратор, выделяющий нулевой младший байт адреса (на вход подаются инверсные значения $A_7..A_0$). Результирующий сигнал неполного дешифратора на адрес $0300h$ образуется при помощи элемента «И», объединяющего сигналы предварительных дешифраторов.

В реальной схеме при построении дешифратора адреса обязательно нужно использовать системный сигнал AEN (Address ENable) [15], который является индикатором режима ПДП.

Пассивный сигнал AEN устанавливается в режиме ПДП, когда контроллер формирует двоянный сигнал управления (см. п. 14.4.3) – MRD и IOWR (или MWR и IORD). На шине адреса толь-

ко адрес для памяти. Нужное ВУ включается отдельным разрешающим сигналом DACKi – разрешение ПДП. Поэтому, чтобы остальные ВУ не реагировали сигналы IORD или IOWR и на адрес, установленный для памяти, ввели индикатор AEN, запрещающий ВУ в режиме ПДП расшифровывать адрес. Особенности обмена данными при обработке прерывания и в режиме ПДП выходят за рамки пособия.

В настоящее время в качестве дешифратора адреса для ВУ применяют ППЗУ (см. п. 14.2.2) [15], цифровой компаратор (см. п. 11.10.1) или простейшую ПЛИС (см. п. 14.3.3), которые допускают настройку на любой требующийся адрес без изменения схемы.

§11.3. Дешифраторы для управления светодиодами индикаторами

11.3.1. Семисегментные индикаторы

Важной частью многих микропроцессорных систем является цифровой семисегментный индикатор (рис. 11.12), который может быть жидкокристаллическим или светодиодным. Сегменты таких индикаторов принято обозначать латинскими буквами от А до Н. В производственных системах чаще используют светодиодные индикаторы, каждый сегмент, которого выполнен в виде отдельного светодиода (или нескольких в крупных индикаторах).



Рис. 11.12

В связи с этим на практике возникает задача преобразования двоичного кода от процессора в код управления семисегментным индикатором (запятой Н управляют отдельно). Например, если на входе нулевой код, то нужно, чтобы светились (были активными) все сегменты, кроме G и H. Аналогично можно представить код управления и для остальных цифровых комбинаций от 1 до 9 (табл. 11.3).

Другие двоичные комбинации (для десятичных значений 10..15), предназначенные для индикации специальных знаков или букв, зависят от типа преобразователя и здесь не рассматриваются. Для большей наглядности клетки таблицы с нулевыми выходными значениями оставлены незаполненными.

Таблица 11.3

Входы					Выходы						
x_3	x_2	x_1	x_0	N_0	A	B	C	D	E	F	G
0	0	0	0	Y_0	1	1	1	1	1	1	
0	0	0	1	Y_1		1	1				
0	0	1	0	Y_2	1	1		1	1		1
0	0	1	1	Y_3	1	1	1	1			1
0	1	0	0	Y_4		1	1			1	1
0	1	0	1	Y_5	1		1	1		1	1
0	1	1	0	Y_6	1		1	1	1	1	1
0	1	1	1	Y_7	1	1	1				
1	0	0	0	Y_8	1	1	1	1	1	1	1
1	0	0	1	Y_9	1	1	1	1		1	1

Такой преобразователь кодов специфичен, но его часто называют дешифратором. При выборе типа подобного преобразователя для семисегментного индикатора нужно иметь в виду, что по внутреннему соединению светодиодов, образующих сегменты, различают две разновидности индикаторов.

11.3.2. Типы семисегментных светодиодных индикаторов

Чтобы уменьшить число внешних выводов индикатора светодиоды сегментов объединяют со стороны катодов или анодов. Чтобы различать эти две разновидности, принято кодировать индикаторы с объединенными катодами (рис. 11.13) буквой «А», а с объединенными анодами (рис. 11.14) буквой «Б». Например, АЛС324А или АЛС324Б.

Пунктиром на рисунках приведены возможные схемы включения, обязательным элементом которых являются токоограничивающие резисторы $R_{огр}$.

Управление отдельными сегментами обычно осуществляют при помощи внешних ключей. Например, в качестве верхних ключей K_B можно использовать схему на $p-n-p$ транзисторе (см. п. 7.2.6), а в качестве нижних ключей K_H – на $n-p-n$ транзисторе (см. п. 7.2.5).

Допустимо в некоторых случаях подключать сегменты напрямую к выводам микросхемы (см. п. 3.1.2) или процессора (см. п. 11.5.1).

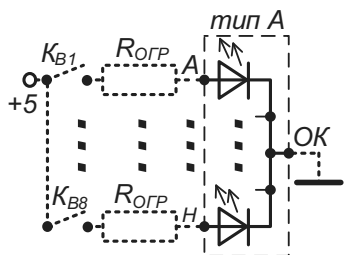


Рис. 11.13

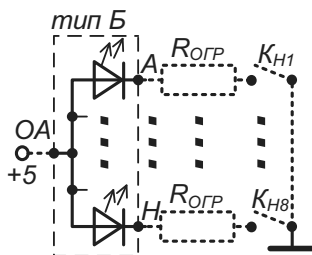


Рис. 11.14

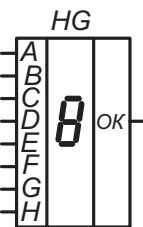


Рис. 11.15

На принципиальных схемах цифровые семисегментные индикаторы изображают в виде УГО (рис. 11.15) и с буквенным кодом *HG*.

Для управления подобными индикаторами в свое время был разработан ряд специализированных микросхем. Основные особенности их рассмотрим на примере преобразователей двоичного кода в семисегментный КР514ИД1 и КР514ИД2, которые отличаются типом выхода.

11.3.3. Дешифратор К514ИД1

предназначен для работы с индикаторами типа «А» (с общим катодом). Рассмотрим его особенности и УГО на примере схемы, приведенной на рис. 11.16.

В некоторых случаях вместо символа дешифратора *DC* используют символ функции преобразования кодов (*X/Y*).

Дешифратор К514ИД1 имеет 4 информационных входа и один вход для разрешения индикации *E*. Чтобы запретить работу дешифратора нужно подать нулевой (пассивный) сигнал, чтобы разрешить – единичный.

Семь выходов (по числу сегментов) выполнены по схеме с открытым (никуда внутри микросхемы не подключенным) эмиттером (рис. 11.17).

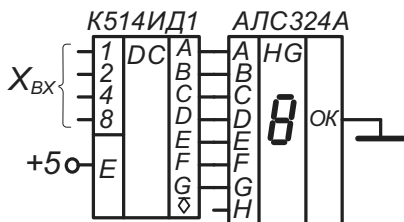


Рис. 11.16

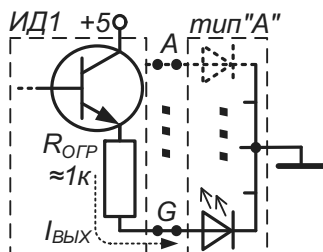


Рис. 11.17

Одним из достоинств микросхемы КР514ИД1 является наличие внутренних твердотельных резисторов $R_{огр}$, которые ограничивают токи через сегменты на уровне около 5 мА. Поэтому внешние резисторы включать не нужно, что упрощает схему.

Отметим также, что дешифратор служит своеобразным усилителем тока, что особенно важно для сравнительно старых типов ОВМ, имеющих недостаточную нагрузочную способность.

Однако следует помнить и о недостатках полупроводниковых резисторов, выполненных в толще кристалла. В первую очередь это существенные технологический разброс и температурный дрейф, что приводит к некоторому изменению яркости индикатора. Кроме того, величину тока сегментов нельзя изменять по своему усмотрению, например, в динамическом индикаторе (см. п. 11.4.1), когда необходимо в несколько раз увеличивать токи по сравнению со статическим режимом.

11.3.4. Дешифратор КР514ИД2,

предназначенный для индикаторов с общим анодом (типа «Б»), имеет такие же внешние выводы (рис. 11.18), но инверсный тип выходного каскада (рис. 11.19), обеспечивающий другое направление выходного тока, чего требуют подключенные к источнику питания общие аноды индикатора типа «Б».

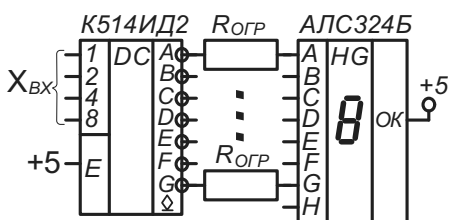


Рис. 11.18

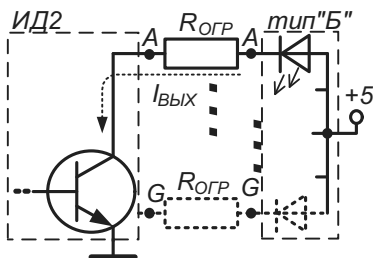


Рис. 11.19

Другой особенностью дешифратора является отсутствие внутренних ограничивающих резисторов, поэтому схема подключения, оставаясь по сути прежней, усложняется за счет дополнительных внешних резисторов $R_{огр}$. Но данный недостаток окупается возможностью получения повышенной яркости индикатора путем выбора величины сопротивления:

$$R_{огр} = U_{Rогр} / I_{сд} = (E - U_{сд} - U^0) / I_{сд} \quad (11.5)$$

Вход *E* можно использовать для управления яркостью, применяя широтно-импульсную модуляцию, то есть изменяя ширину импульса (коэффициент заполнения).

11.3.5. Обслуживание однодекадного индикатора

Пусть для примера индикатор совместно с дешифратором подключен к порту *P1* ОВМ (рис. 11.20) и необходимо индцировать цифровые данные (от 0 до 9), которые заранее были подготовлены в переменной *mBufInd*. Чтобы не исказить соседние биты, предварительно читаем порт *P1* (см. п. 11.2.1):

```
Mov A, P1
Anl A, #11110000b
Orl A, mBufInd
Mov P1, A
```

На языке Си та же процедура записывается одним оператором:

```
P1 = (P1 & 0xF0) | mBufInd;
```

Однако компилятор превращает такую краткую запись в набор машинных кодов, размер которых может превышать размер двоичных кодов после ассемблера.

11.3.6. Обслуживание двухдекадного индикатора

Рассмотрим схему (рис. 11.21) для вывода цифровой информации из ОВМ в десятичном виде, состоящую из двух дешифраторов типа КР514ИД1, управляемых тетрадами порта *P1*. В комплекте с дешифратором ИД1 нужно использовать индикаторы с объединенными катодами, например, АЛС324А.

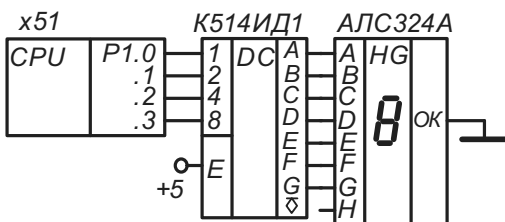


Рис. 11.20

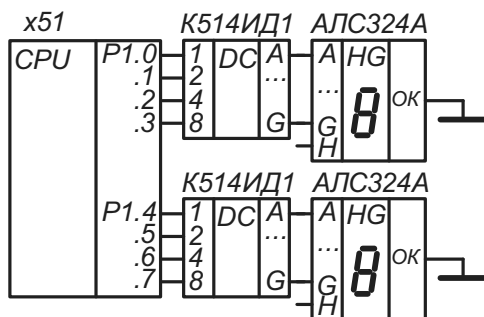


Рис. 11.21

Пусть двоично-десятичные данные для индикации двух декад хранятся в младших тетрадах двух байтовых (и обычно смежных) переменных *mBuflndBCD0* и *mBuflndBCD1*, старшие тетрады которых обнулены.

Программа должна извлечь два последовательных байта и объединить их младшие тетрады в одном байте:

```
Mov A, mBuflndBCD1      ; Код для старшего знакоместа
Swap A                  ; в старшей тетраде аккумулятора.
Orl A, mBuflndBCD0      ; Объединение тетрад.
Mov P1, A
```

Эта же программа на языке Си:

```
P1 = (mBuflndBCD0) | (mBuflndBCD1 << 4);
```

Количество машинных кодов в таком фрагменте зависит от компилятора. Хороший оптимизирующий компилятор дает около 10 байтов. Но если этот фрагмент оформить как процедуру, то количество кодов резко возрастет.

Реальные индикаторы, как правило, являются многозначными и рассмотренный статический метод индикации, когда каждый индикатор управляется своим дешифратором, становится невыгодным, поскольку схема становится сложной и от ОВМ требуется много внешних выводов. Например, для 4 десятичных знаков требуется 16 выводов и 4 внешних микросхемы дешифратора, что в ряде случаев не оставляет ресурсов для других внешних объектов.

Вместе с тем, инерционность человеческого зрения дает возможность использовать принцип динамической индикации, при которой, как в кино, поочередно индицируют отдельные знакоместа, сливающиеся в единое целое изображение (см. далее).

§11.4. Дешифраторы в динамических индикаторах

11.4.1. Схема динамического индикатора

Главным его отличием является то, что индикаторы светятся по очереди. Это позволяет использовать общий семисегментный преобразователь для всех одноименных сегментов, что упрощает схему и уменьшает количество внешних выводов ОВМ, необходимых для управления.

На рис. 11.22 приведена такая схема трехразрядного индикатора на основе встроенного светодиодного модуля типа «А» КИПЦ38-3/8А, одноименные сегменты в котором объединены. Для подключения отдельных

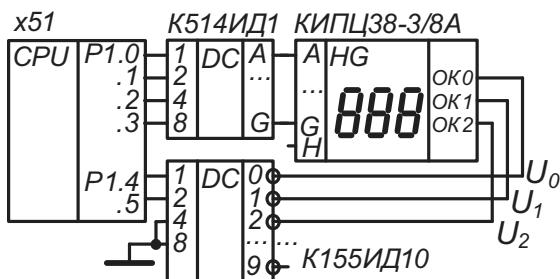


Рис. 11.22

индикаторов к общему проводу источника питания имеются, как и положено для типа «А», выводы общих катодов ОК0, ОК1 и ОК2 соответственно для единиц, десятков и сотен. Управляет подачей питания на декады дополнительный дешифратор типа К155ИД10, служащий и усилителем тока.

Дело в том, что через выводы общих катодов протекают суммарные токи сегментов, и поэтому микросхема К155ИД10, имеющая выходной ток около 80 мА [14], не только выполняет функции дешифратора, включая только одно знакоместо, но представляет набор сравнительно мощных ключей.

Необходимо отметить, что токи сегментов следует увеличивать во столько раз, сколько знаков в индикаторе, иначе будет тусклое свечение, поскольку средний ток через сегмент, определяющий яркость, уменьшается с ростом числа знаков. Но преобразователь КР514ИД1 имеет внутренние токозадающие резисторы, что не позволяет изменять ток сегментов. Поэтому при большем числе знакомест следует использовать микросхему КР514ИД2 с внешними резисторами, чтобы задать необходимый импульсный ток (см. п. 11.3.4).

Дешифратор К155ИД1 в принципе можно исключить, реализовав его функцию программно (см. п. 11.5.1). Но количество занятых внешних выводов ОВМ для выдачи информации при этом увеличится и иногда придется ставить внешние ключи для усиления тока (см. п. 11.5.1).

Примечание. В свое время были разработаны специализированные микросхемы, например, отечественная КР514ИД5, внутри которой были встроенные преобразователь семисегментного кода, дешифратор знакомест и счетчик указатель, обеспечивающий регенерацию изображения. Подобные специализированные микросхемы в настоящее время практически не применяются, так как

дешевле использовать для этой цели ОВМ, которая выполняет большинство указанных функций программно (см. п. 11.6.1).

11.4.2. Временные диаграммы динамического индикатора

Таким образом, дешифратор КР514ИД1 формирует информационные сигналы на сегментах всех трех индикаторов. Но светиться будет только тот индикатор, катод которого подключен к общей точке, то есть подано питание. Чтобы обеспечить слитное изображение всех индикаторов, необходимо организовать поочередное и периодическое их включение с высокой частотой (рис. 11.23).

Для простейшей оценки частоты «смены кадра» F_K можно использовать кинематографический стандарт и принять $F_K \geq 24 \text{ Гц}$ ($T_K \leq 42 \text{ мс}$). Отсюда можно найти время включения одного индикатора, которое для каждого из трех индикаторов должно быть в три раза меньше периода T_K , то есть $t_{\text{и}} \leq 42 / 3 = 14 \text{ мс}$.

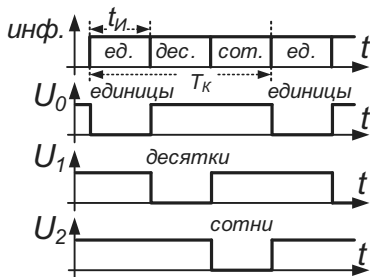


Рис. 11.23

11.4.3. Реализация периодического обслуживания динамического индикатора

Важную роль в приведенной схеме играет программа, которая должна установить информационный двоично-десятичный код одной из декад на входах дешифратора КР514ИД1 и подать питание на соответствующую декаду. Такой периодический цикл обслуживания очередного индикатора обычно реализуется при помощи процедуры прерывания по переполнению таймера ОВМ, например, через каждые 5–10 мс.

То есть подпрограмма индикации (*mIndZn3*), которая описана в п. 11.4.6, вызывается из ассемблерного прерывания (см. п. 8.7.5):

mProcIntT0:

... ; Загрузка на отсчет 10 мс (см. п. 8.7.2), счет «тиков» (см. п. 8.7.5).

Call mIndZn ; Индикация очередного знакоместа.

Retl

или из процедуры прерывания на языке Си (см. п. 10.6.4):

```

void mIntT0(void) interrupt TF0_VECTOR {
    ...           // Загрузка на отсчет 10 мс, счет системных «тиков».
    mIndZn();      // Индикация очередного знакоместа.
}

```

11.4.4. Подготовка данных для индикации

Для управления динамическим индикатором в начале программы должны быть объявлены:

- количество знакомест в виде константы, например, *mKvoZn*,
- массив для буфера индикации (*mBuflnd [mKvoZn]*), выполняющий роль своеобразной «игрушечной» видеопамати,
- переменная-указатель для индексации массива (*mPtrZn*).

На ассемблере указанные объявления записываются в следующем виде:

```

mKvoZn equ 3
.DATA
mBuflnd ds mKvoZn
mPtrZn   ds 1

```

На Си:

```

#define mKvoZn 3
data unsigned char mBuflnd[mKvoZn];
unsigned char mPtrZn=0;

```

11.4.5. Двоично-десятичное преобразование байта

В объявленном буфере для индикации должны храниться двоично-кодированных десятичные числа. Поэтому для преобразования значения байта (от 0 до 255) в три декады необходимо вычислить значения каждой из них, начиная со старшей. То есть, поделив байт на 100, получим число сотен, поделив остаток на 10, получим число десятков, а последний остаток будет числом единиц.

Пусть для ассемблерной программы преобразуемое двоичное число предварительно занесено в аккумулятор. Результатом работы программы будут три двоично-десятичных декады, размещенные в буфере индикации (сотни, десятки и единицы).

```

mAccToBCDtoBuflnd:
    Mov B, #100           ; Определяем
    Div AB                ;      число сотен
    Mov mBuflnd+2, A      ;      и сохраняем их.

```


Mov A, #10	<i>; Будем вычислять десятки</i>
Xch A, B	<i>; Остаток от сотен в A</i>
Div AB	<i>; Вычисляем и</i>
Mov mBufInd+1, A	<i>; сохраняем десятки.</i>
Mov mBufInd, B	<i>; Сохраняем остаток- единицы.</i>
Ret	

И далее в программе по мере готовности данных можно заносить двоично-десятичные данные для индикации, например, переменной *mBt*:

```
Mov A, mBt
Call mAccToBCDtoBufInd
```

На Си указанная процедура получает байт в виде аргумента:

```
void mByteToBCDtoBufInd (byte mBt) {
    mBufInd[2] = mBt / 100;           // сотни
    mBufInd[1] = (mBt % 100) / 10;    // десятки
    mBufInd[0] = (mBt % 100) % 10;    // единицы
}
```

11.4.6. Процедура индикации знакоместа mIndZn3

должна извлечь данные текущей декады из буфера индикации по указателю знакоместа в *младшую* тетраду временной переменной (старшая тетрада обнулена). Затем указатель знакоместа (в соответствии со схемой рис. 11.22) занести в пустую *старшую* тетраду переменной и копировать ее в порт *P1*. Это приведет к свечению только текущего индикатора.

После этого нужно подготовить (инкрементировать или декрементировать) указатель знакоместа для следующей цифры. Если указатель знакоместа выйдет за границы, то необходимо начать с начала, задав исходное значение указателя.

Подобные процедуры обычно оформляют в виде подпрограммы или макроса.

На Си

```
void mIndZn3 (void){
    P1 = mBufInd [mPtrZn] | (mPtrZn << 4);    // Включаем знакоместо.
    mPtrZn++;                                // Подготавливаем следующий адрес.
    If (mPtrZn == mKvoZn) mPtrZn = 0;        // Все ли знакоместа обслужили?
}
```

На Ассемблере:

```
mIndZn3:
```

```

Mov A, mPtrZn      ; Заносим адрес
Add A, #mBuferInd   ; очередной цифры
Mov R0, A           ; в регистр косвенной адресации.
Mov A, @R0          ; Извлекаем код очередной цифры в A.
Swap A             ; Объединяем информацию
                   ; с номером знакоместа
Mov P1, A           ;
Inc mPtrZn          ;
Mov A, mPtrZn       ;
CJNE A, #mKvoZn, in2 ; Все знакоместа обслужили?
Mov mPtrZn, #0      ; Начинаем сначала.
in2:
Ret

```

§11.5. Программное управление сегментами одиночного индикатора

11.5.1. Схема

В тех случаях, когда ОВМ имеет достаточную нагрузочную способность, можно исключить внешний преобразователь семисегментного кода, используя программное преобразование и управление отдельными сегментами. Это позволяет выводить на индикацию кроме цифр и некоторые буквы, а также управлять при необходимости запятой.

В качестве примера рассмотрим одиночный индикатор с общими анодами (АЛС324Б), подключенный к порту **P1** ОВМ типа *AT89C2051* (рис. 11.24), имеющей выходной ток внешнего вывода до 20 мА.

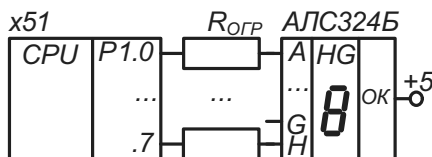


Рис. 11.24

При выборе ограничивающих резисторов нужно дополнительно иметь в виду, что ограничен и максимальный суммарный ток кристалла на уровне около 80 мА.

Опишем подключение сегментов индикатора к выводам порта **P1** при помощи именованных констант. Для наглядности пока не будем учитывать, что сегменты светятся при нуле и будем использовать единицу, как указатель, активного сегмента в двоичном числе.

mSgA EQU 00000001b ; Сегмент А подключен к 0-му разряду.

...

mSgH EQU 10000000b ; Сегмент Н подключен к 7-му разряду.

Для программы на языке Си, который не имеет двоичной записи чисел, опишем подключение отдельных сегментов индикатора к выводам порта для наглядности в виде десятичных двоично-взвешенных констант:

```
#define mSgA 1
```

...

```
#define mSgH 128
```

11.5.2. Таблицы для преобразования тетрады в код управления семисегментным индикатором

Оформим в ПЗУ таблицу для преобразования кодов с символическим именем (начальным адресом) *mTabIDC7* размером в 16 байт. Отдельные коды управления семисегментным индикатором, соответствующие табл. 11.3 (см. п. 11.3.1), будем формировать на основе символических имен включенных сегментов при помощи ассемблерной директивы **DB** (см. п. 7.12.8):

mTabIDC7:

```
db mSgA + mSgB + mSgC + mSgD + mSgE + mSgF ; Цифра 0
```

```
db mSgB + mSgC ; 1
```

```
db mSgA + mSgB + mSgD + mSgE + mSgG ; 2
```

.....

```
db mSgA + mSgD + mSgE + mSgF ; Буква С
```

```
db mSgA + mSgD + mSgE + mSgF + mSgG ; Буква Е
```

Таблица на Си:

```
code unsigned char mTabIDC7[16] = {
```

```
    mSgA | mSgB | mSgC | mSgD | mSgE | mSgF, // Цифра 0
```

```
    mSgB | mSgC, // Цифра 1
```

.....

```
    // 2..9, A..F
```

```
};
```

Иногда полезно ввести константу для сегментов «восьмерки», когда на все сегменты, кроме H, поступают единицы, то есть *mSg8 = 127*. Тогда некоторые символические коды будут проще, например, код для нуля будет записан, как «восьмерка без сегмента G», то есть *mSg8 – mSgG*.

11.5.3. Ассемблерная программа для табличного преобразования кодов

основана на команде извлечения табличного элемента по заданному в аккумуляторе числу-индексу (см. п. 7.8.6):

```
mAkkToDC7_n:  
    Mov DPTR, mTablDC7  
    MovC A, @A+DPTR  
    Ret
```

Таблицу можно поместить в любом месте ПЗУ. Но эта программа не перемещается, поскольку при переносе кодов таблицы нужно пересчитывать ее адрес в подпрограмме.

Особенностью перемещаемой программы является слитная запись программы с таблицей и использование адресации относительно программного счетчика (см. п. 7.8.6), что позволяет перемещать кодовый фрагмент без пересчета адресов.

```
mAkkToDC7_p:  
    Inc A      ; Таблица находится за командой Ret – добавляем 1.  
    MovC A, @A + PC  
    Ret  
mTablDC7:  
    ; здесь размещается 16-байтовая таблица.
```

На Си табличное преобразование упрощено и сводится к обычной индексации массива, как будет показано ниже.

11.5.4. Программа управления сегментами индикатора

Поскольку из схемы рис. 11.22 следует, что сегмент светится при нуле, поэтому извлеченный из таблицы код управления должен быть инвертирован.

Программа на ассемблере:

```
Mov A, mBufInd  
Call mAkkToDC7      ; Извлекаем табличный элемент  
Cpl A               ; Сегмент светится при нуле.  
Mov P1, A
```

Программа на Си:

```
P1 = ~ mTablDC7[mBufInd & 0x0F];
```

Повторно отмечаем, что значения *mBufInd* от 0 до 9 – это цифры, а от 11 до 15 – это коды букв.

Обнуление старшей тетрады при помощи операции & сделано для страховки и эту операцию можно исключить.

§11.6. Динамический индикатор с программно-управляемыми сегментами

11.6.1. Схема четырехзначного динамического индикатора,

приведенная на рис. 11.25, имеет минимум внешних микросхем, функции которых реализуются программно.

Одноименные сегменты всех индикаторов типа «Б» подключены, как и ранее (см. п. 11.5.1), напрямую в порт *P1* без преобразователя семисегментного кода КР514ИД2.

Напомним, что схема иллюстрационная и поскольку ограничен максимальный суммарный ток кристалла, то могут понадобиться внешние ключевые усилители.

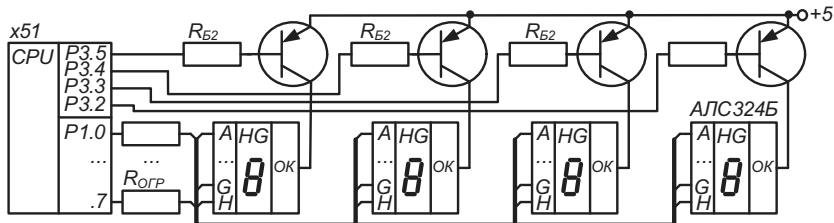


Рис. 11.25

В отличие от предыдущей схемы динамического индикатора (см. п. 11.4.1), в которой для выбора знакоместа использовался внешний дешифратор, в данном случае функция дешифратора реализуется программно (см. п. 11.2.2). Подачей питания на одно из знакомест управляют 4 бита порта *P3*, при помощи внешних транзисторных ключей *p-n-p* типа (см. п. 7.2.6), обеспечивающих достаточно большой ток индикатора, равный сумме токов отдельных сегментов.

Поэтому в данном случае к основному алгоритму обслуживания динамического индикатора (см. п. 11.4.6) – установить данные на сегменты и включить знакоместо, необходимо добавить программное преобразование данных в семисегментный код (см. §11.5) и программную реализацию дешифратора знакоместа (см. п. 11.2.2).

Программа на ассемблере для тренировки предлагается написать самостоятельно.

Кроме рассмотренных светодиодных индикаторов существуют и жидкокристаллические индикаторы (ЖКИ). Их особенность – меньшая потребляемая мощность.

Но при управлении ими есть сложности: они не допускают постоянного уровня напряжения на сегментах, следовательно, необходимо подавать сигнал с нулевой постоянной составляющей. Поэтому на практике чаще используются так называемые модули ЖКИ с уже встроенными схемами управления, простеньким процессором, «видеопамятью» и стандартным интерфейсом.

§11.7. Шифраторы

11.7.1. Общие сведения

Шифратор (кодер) – кодовый преобразователь, который выполняет функцию, обратную дешифратору, и преобразует номер одного активного входа в двоичный код на выходе. То есть на входе действует *унарный* код, а на выходе – двоичный код номера активного входа.

Чаще всего шифраторы выполняют полными с количеством входов равным 2^N , где N – число выходов. Для иллюстрации принципа действия рассмотрим табл. 11.4 для реализации шифратора с 8 входами и тремя двоичными выходами. Для наглядности нулевые входные значения не показаны.

Таблица 11.4

Входы								Выходы		
Y_0	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6	Y_7	Z_2	Z_1	Z_0
1								0	0	0
	1							0	0	1
		1						0	1	0
			1					0	1	1
				1				1	0	0
					1			1	0	1
						1		1	1	0
							1	1	1	1

Анализируя таблицу, можно увидеть, что выходной сигнал Z_0 будет единичным, когда единичны входы **или** Y_1 , **или** Y_3 , **или** Y_5 , **или** Y_7 . Поэтому можно записать Y_0 через логическую сумму с четырьмя указанными переменными:

$$Z_0 = Y_1 + Y_3 + Y_5 + Y_7 \quad (11.6)$$

Остальные выражения нетрудно написать по аналогии, выделяя соответствующие переменные, образующие активный выходной сигнал:

$$Z_1 = Y_2 + Y_3 + Y_6 + Y_7 \quad (11.7)$$

$$Z_2 = Y_4 + Y_5 + Y_6 + Y_7 \quad (11.8)$$

11.7.2. Схема линейного шифратора на 8 входов

Из уравнений (11.6)–(11.8) видно, что для реализации шифратора с тремя двоичными выходами нужны три элемента «ИЛИ» (рис. 11.26). Такой шифратор называют линейным, так как он представляет набор (линейку) однотипных элементов.

Шифратор можно построить и на элементах «И», используя принцип дуальности [23]. При этом говорят, что кодер реализуется в инверсном базисе, при котором входные и выходные сигналы будут проинвертированы.

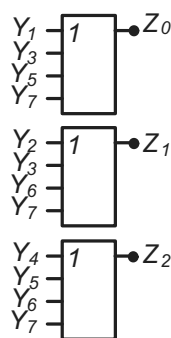


Рис. 11.26

При разработке многоразрядных кодеров возникает технологическая сложность, связанная с увеличением число входов логических элементов. Поэтому, чтобы использовать маловходовые логические элементы, применяют каскадный принцип, как и в дешифраторе (см. п. 11.1.6), то есть, сначала обрабатывают одну группу сигналов, а потом результат объединяют с результатом обработки другой группы.

11.7.3. Каскадный шифратор с 16 входами

имеет уже четыре выхода (табл. 11.5). Используем его для иллюстрации каскадного принципа построения шифратора [22].

Для этого разделим выходные разряды на две группы, младшую Z_0, Z_1 и старшую Z_2, Z_3 . Коды в этих группах из двух битов могут принимать четыре значения: «00», «01», «10», «11». Комбинация «00» является для функции «ИЛИ» неинформативной и в дальнейшем не учитывается.

Таблица 11.5

№	Z_3	Z_2	Z_1	Z_0
Y_0	0	0	0	0
Y_1	0	0	0	1
Y_2	0	0	1	0
Y_3	0	0	1	1
Y_4	0	1	0	0
Y_5	0	1	0	1
Y_6	0	1	1	0
Y_7	0	1	1	1
Y_8	1	0	0	0
Y_9	1	0	0	1
Y_{10}	1	0	1	0
Y_{11}	1	0	1	1
Y_{12}	1	1	0	0
Y_{13}	1	1	0	1
Y_{14}	1	1	1	0
Y_{15}	1	1	1	1

Вначале реализуем схему анализа входных сигналов для выходных комбинаций только младших разрядов Z_0, Z_1 .

Комбинация «01», выделенная в таблице рамкой, возможна в четырех случаях – когда активен Y_1 или Y_5 , или Y_9 , или Y_{13} . Поэтому можем записать логическое уравнение для комбинации «01»:

$$Z_{01} = Y_1 + Y_5 + Y_9 + Y_{13} \quad (11.9)$$

По аналогии можно записать уравнения и для комбинаций «10» и «11»:

$$Z_{10} = Y_2 + Y_6 + Y_{10} + Y_{14} \quad (11.10)$$

$$Z_{11} = Y_3 + Y_7 + Y_{11} + Y_{15} \quad (11.11)$$

Затем, используя промежуточные выходные переменные Z_{01}, Z_{10}, Z_{11} можно вычислить и значения отдельных разрядов Z_0 и Z_1 . Например, младший бит Z_0 будет единичным в комбинациях «01» или «11», а старший бит Z_1 – в комбинациях «10» или «11», то есть

$$Z_0 = Z_{01} + Z_{11} \quad (11.12)$$

$$Z_1 = Z_{10} + Z_{11} \quad (11.13)$$

На основе приведенных рассуждений и уравнений (11.9) – (11.13) можно нарисовать принципиальную схему одного каскада шифратора для получения младших выходных разрядов Z_0 и Z_1 (рис. 11.27).

Схему формирования старших разрядов Z_2 и Z_3 каскадного шифратора рекомендуется для тренировки реализовать самостоятельно.

11.7.4. Микросхема КР155ИБ1

является *приоритетным* шифратором на 8 инверсных входов, то есть активный входной уровень – нулевой.

Приоритет старшего входа означает, что активными могут быть одновременно два и более сигналов, но закодируется только номер старшего из них.

На рис. 11.28 показано УГО этой микросхемы на примере использования шифратора для кодирования номера нажатой клавиши. Для простоты на рисунке не показаны подтягивающие резисторы, фиксирующие единичный сигнал при непроводящем ключе (см. п. 2.5.4).

Дополнительный входной инверсный сигнал E разрешает работу шифратора и используется обычно для наращивания разрядности шифратора.

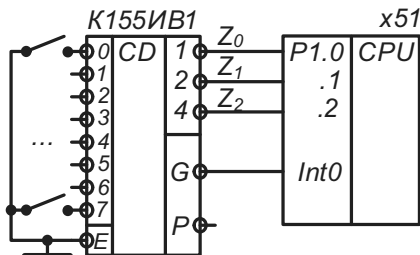


Рис. 11.28

Выходы помечены на рисунке весами двоичных разрядов (1, 2, 4), они тоже инверсные.

Кроме того имеются два специфичных инверсных выхода. Один из них P (выход переноса), используется для увеличения разрядности шифратора [5]. Второй выход G является индикатором активности входов. Он активен, когда возбужден хотя бы один из входов, и позволяет устранить особую ситуацию при считывании выходного кода, возникающую при пассивности всех входов. Этот выход может использоваться, например, системой прерывания ОВМ.

Нетрудно видеть, что использование микросхемы шифратора экономит внешние выводы ОВМ. Самим написать программу для чтения кода.

11.7.5. Программная реализация функции шифратора

В некоторых случаях при считывании сигнала с линейки ключей можно исключить внешнюю микросхему и реализовать функцию шифратора программно (рис. 11.29). В этой схеме при нажатии одной клавиши формируется нулевой активный уровень (см. п. 7.2.2). Поэтому после чтения порта *P1* и дополнительного инвертирования (см. п. 10.1.3) получается двоичный код с одной единицей. Чтобы определить порядковый номер единичного разряда обычно сдвигают код вправо до исчезновения единицы и считают число сдвигов. Ниже представлена соответствующая процедура для этого:

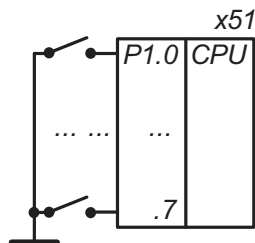


Рис. 11.29

```
unsigned char mCoder(unsigned char mY) {
    unsigned char mCount = 0;
    while (mY != 0) {
        mY = mY >> 1;
        mCount++;
    }
    return mCount;
}
```

Самостоятельно выяснить, является ли рассмотренный программный шифратор приоритетным.

Кроме клавишных пультов для формирования скан-кода нажатой клавиши шифраторы применяются в составе параллельных АЦП (см. п. 18.1.2), контроллерах прерываний для преобразования сигнала запроса прерывания в двоичный код вектора прерывания и в других устройствах. Например, связку кодер-декодер (рис. 11.30) можно использовать с целью защиты данных для перестановки комбинаций двоичного кода.

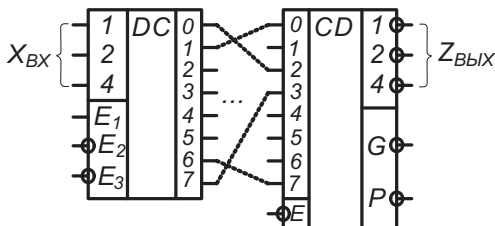


Рис. 11.30

§11.8. Коммутаторы

11.8.1. Общие сведения

Коммутатор переключает поток данных с одной линии (канала) на другую. На практике различают два типа коммутаторов (переключателей), которые в переводной литературе определены терминами *демультиплексор* и *мультиплексор*.

Первый из них, который распределяет информацию от одного источника на много направлений (рис. 11.31), можно реализовать на основе дешифратора (см. п. 11.1.2). Поэтому такой тип коммутатора относят иногда к распределителям (см. п. 13.4.2). В неявном виде демультиплексор образуется в микропроцессорной системе, в которой множество внешних устройств получают данные от единой информационной магистрали (см. п. 14.5.1).

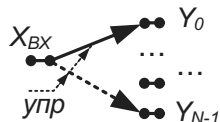


Рис. 11.31

Под коммутатором (или без перевода *мультиплексором*) обычно понимают устройство переключющее информацию от многих локальных источников на одно общее направление (рис. 11.32). Коммутаторы (явные или неявные) обычно реализуют на основе ключей, аналоговых или логических.

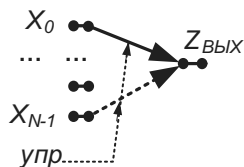


Рис. 11.32

11.8.2. Аналоговый и логический ключи

Аналоговый ключ должен максимально точно передавать входной сигнал на выход (рис. 11.33), поэтому он по своим параметрам близок к идеальному, контактному ключу (см. п. 15.1.1). На основе набора аналоговых ключей можно реализовать коммутаторы обоих типов, меняя местами входы и выходы. Это свойство коммутатора называют *обратимостью*.

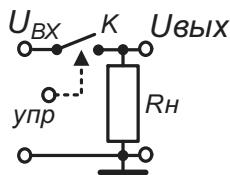


Рис. 11.33

Логический ключ этим свойством не обладает, поскольку выполнен на основе *однонаправленного* логического элемента «И» (рис. 11.34). Если управляющий сигнал Y будет нулевым, то и выходной также будет нулевым. Если уп-

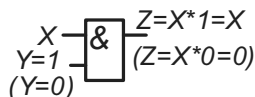


Рис. 11.34

правляющий сигнал будет единичным, то

$$Z = X \cdot Y = X \cdot 1 = X. \quad (11.14)$$

Говорят, что в этом случае логический ключ пропускает входной сигнал X на выход.

Поскольку на выход логического элемента нельзя подавать входной сигнал, поэтому коммутаторы на основе логических ключей являются необратимыми переключателями цифровых сигналов.

11.8.3. Реализация коммутатора

Коммутатор на логических ключах представляет набор логических элементов «И» и нуждается в схеме объединяющей их выходные сигналы, в качестве которой выступает схема логического сложения «ИЛИ» (рис. 11.35).

Чтобы выходной сигнала коммутатора определялся только одним логическим ключом, управляющие сигналы формируются на выходах дешифратора, который преобразует двоичный номер выбранного канала в соответствующий активный управляющий сигнал.

То есть сигнал будет пропускать только один элемент «И», остальные логические ключи своим нулевым сигналом влиять не будут.

Отметим, что вспомогательный дешифратор управления обычно реализуется на дополнительных входах этих же элементов «И» подобно рассмотренному линейному дешифратору (см. п. 11.1.4).

УГО коммутатора (на примере микросхемы К555КП7) приведено на рис. 11.36 и имеет знак функции MX . Особенностью данной микросхемы является наличие второго инверсного выхода.

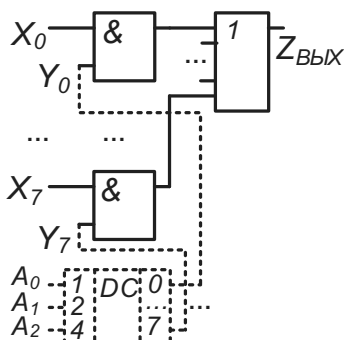


Рис. 11.35

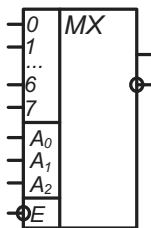


Рис. 11.36

11.8.4. Каскадное соединение коммутаторов

Для наращивания разрядности (числа входов) используется каскадное соединение коммутаторов. Пусть на основе 4-входовых коммутаторов надо реализовать 16-входовой (рис. 11.37).

Как и в каскадном дешифраторе (см. п. 11.1.6) выделяется главное устройство, работающее со старшей частью управляющего адресного кода (A_2 и A_3). Выходы подчиненных коммутаторов, работающих с младшей частью адреса (A_0 и A_1), коммутируются главным коммутатором, который пропускает код только от одного подчиненного коммутатора, адресуемого старшей частью адреса.

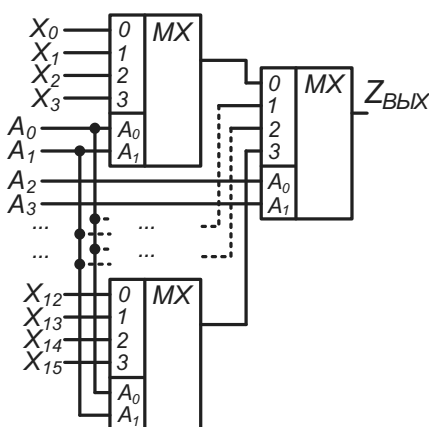


Рис. 11.37

11.8.5. Неявный

коммутатор на основе элементов с открытым коллектором

Коммутатор часто применяют для подключения к общим информационным линиям. Но использование коммутаторов возможно тогда, когда не предполагается наращивание числа входов. Поэтому при организации доступа к общим линиям кроме явно выделенного коммутатора могут использоваться и другие переключающие цепи неявного вида.

В этих схемах выходной элемент коммутатора «ИЛИ», воспринимающий сигналы с логических ключей реализован распределенным в пространстве «монтажным ИЛИ». В качестве примера на рис. 11.38 приведена схема неявного коммутатора на основе логических ключей с открытым коллектором.

Если управляющий сигнал Y пассивен и элемент «И» не выбран, то внутренний ключевой транзистор не проводит и никак не влияет на выходной сигнал «монтажного ИЛИ». И только одна схема,

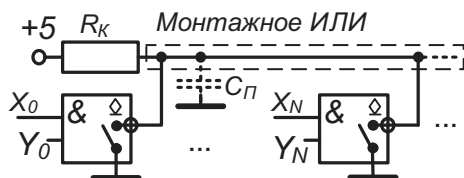


Рис. 11.38

управляющий сигнал которой единичен, будет управлять транзистором, устанавливая нулевой и единичный уровни.

Управляющий сигнал может формироваться, как и ранее, при помощи дешифратора или вычисляться программным путем,

например, на основе адресного кода, полученного по информационной линии в последовательном виде (см. п. 8.1.4).

Преимуществом такого неявного коммутатора является его распределенность в пространстве (нет необходимости в дополнительных проводах, кроме информационного) и возможность подсоединения дополнительных источников сигнала путем монтажа без изменения схемы в целом.

Однако, невысокое быстродействие подобных ключевых схем с резисторной нагрузкой из-за сравнительно долгого заряда паразитной емкости C_{Π} нагрузки через резистор R_K (см. п. 1.2.1) ограничивает число входов и протяженность выходной линии. Поэтому такие неявные коммутаторы применяют в основном для низкочастотных межмикросхемных соединений, когда протяженность выходной линии не превышает нескольких десятков сантиметров, например, в последовательных интерфейсах типа клавиатурного (см. §12.7), I²C [25], и в других подобных случаях.

11.8.6. Неявный коммутатор на основе элементов с третьим состоянием,

приведенный на рис. 11.39 и реализованный по тому же принципу, имеет существенно лучшее быстродействие. Когда ни одно из устройств не выбрано, все схемы находятся в третьем состоянии, которое отражено на рисунке средним положением иллюстративного перекидного ключа. То есть в пассивном состоянии источники информации не влияют на выходной сигнал.

И только один элемент, получивший активный уровень управляющего сигнала Y (от дешифратора или программы) будет формировать при помощи своего перекидного ключа (двухтранзисторного выходного каскада) двоичные данные.

Более высокое быстродействие в этой схеме обеспечивается форсированным зарядом и разрядом паразитной емкости C_{Π} через транзисторы (биполярные или полевые) двухключевого выходного каскада (см. п. 2.2.2). Поэтому такой тип неявного коммутатора широко используется при подключении устройств к микропроцессорным шинам (см. §14.5).

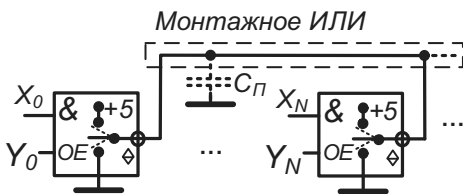


Рис. 11.39

11.8.7. Реализация логической табличной функции

Пусть для примера задана логическая функция произвольного вида с тремя входами (табл. 11.6), которые имеют 8 возможных комбинаций $Y_0..Y_7$. Для конкретности используем логические значения мажоритарного элемента (см. п. 11.10.5).

Таблица 11.6

Входы				Выход
X_2	X_1	X_0	№	$Z_{ВЫХ}$
0	0	0	Y_0	0
0	0	1	Y_1	0
0	1	0	Y_2	1
3	1	1	Y_3	0
4	1	0	Y_4	1
5	1	0	Y_5	1
6	1	1	Y_6	1
7	1	1	Y_7	0

Используя входные переменные для адресации коммутатора можно реализовать таблично заданную функцию, устанавливая необходимые значения $Z_{ВЫХ}$ на входах коммутатора, соответствующих номеру логической комбинации (рис. 11.40).

Например, нулевой комбинации входных сигналов Y_0 соответствует значение $Z_{ВЫХ} = 0$, поэтому нулевой вход коммутатора подключают к общей точке. Тогда как на третий вход коммутатора необходимо подать единичный уровень, так как для третьей комбинации (Y_3) $Z_{ВЫХ} = 1$.

Таким образом, меняя распыку входных каналов можно задавать необходимые значения табличной функции. Если же входы коммутатора подключить к запоминающим ячейкам (см. п. 14.1.3), то получится прообраз программируемого логического элемента (см. п. 14.3.2).

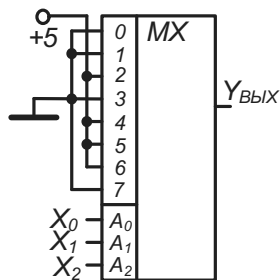


Рис. 11.40

11.8.8. Управление внешним коммутатором

Чтобы уменьшить число внешних выводов, необходимых микропроцессору для чтения информации с нескольких цифровых датчиков,

можно использовать коммутатор. Например, для чтения состояния 16 датчиков (рис. 11.41) можно обойтись только 5 выводами ОВМ.

Отметим, что при будущей разрядности коммутатора экономия будет значительней, но нужно иметь в виду, что реагирование программы на изменение состояния датчика будет замедленной и зависеть от алгоритма опроса.

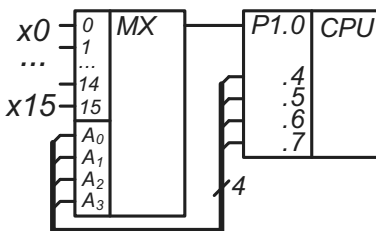


Рис. 11.41

В качестве примера рассмотрим ассемблерную процедуру для чтения состояния датчика, номер которого задан в переменной *mNomerK*. Программа должна установить этот номер в 4 старших битах порта *P1*, не затрагивая остальные биты. Затем нужно копировать выходной сигнал коммутатора, например, в бит переноса:

```
Mov A, P1           ; Читать биты порта P1.
Anl A, #00001111b   ; Обнулить четыре старших бита.
Swap A             ; Обменять тетрады.
Orl A, mNomerK      ; Поместить номер канала в младшую тетраду.
Swap A             ; Номер канала в старшей тетраде.
Mov P1, A          ; Вернуть P1 с измененными битами.
Mov C, P1.0
```

Функция на Си:

```
bool mSensor (byte mNomerK) {
    P1 = (P1 & 0x0F) | (mNomerK << 4);
    return P1_0;
}
```

Коммутаторы широко используются в составе более сложных устройств например, в реверсивных регистрах (см. п. 12.5.3) и счетчиках (см. п. 13.1.5). Они переключают потоки между узлами процессоров, перераспределяют информацию между локальными шинами, обеспечивают доступ к общим информационным ресурсам.

В настоящее время коммутаторы используются даже в составе скоростных сетей типа ISDN.

§11.9. Сумматоры

Суммирование двоичных кодов реализуется на основе одноразрядных схем – полусумматора и полного сумматора.

11.9.1. Полусумматор (сумматор по модулю 2)

не учитывает переполнения (переносов из младшего в старший разряд), имеет два входа, которые ниже обозначены буквами a и b , и один выход суммы S .

Входные сигналы имеют четыре возможные комбинации (табл. 11.7), для которых нетрудно вычислить значения выходного сигнала S , суммируя двоичные входные значения.

Таблица 11.7

Входы			Выход
a	b	N_{Σ}	S
0	0	Y_0	0
0	1	Y_1	1
1	0	Y_2	1
1	1	Y_3	0

Из таблицы видно, что переменная S единична для двух комбинаций (**или** для второй, обозначаемой как Y_2 , **или** для третьей Y_3). Основываясь на этом, можно записать формулу для описания работы полусумматора:

$$S = Y_2 + Y_3 = \bar{a} \cdot b + a \cdot \bar{b}. \quad (11.15)$$

Эта логическая функция, осуществляющая так называемое сложение по модулю два, настолько широко используется в вычислительной технике, что получила специальное название «*исключающее ИЛИ*» (*Exclusive Or* или сокращенно *XOR*) и реализуется почти во всех наборах команд микропроцессоров. Поэтому УГО полусумматора можно встретить в двух вариантах, одно со знаком *HS* (от *Half* – половина), которое приведено на рис. 11.42а, а другое со знаком «=1» для «исключающего ИЛИ», которое приведено на рис. 11.42б.

В простейшем случае полусумматор реализуется при помощи элементов «И», «ИЛИ», «НЕ» в соответствии с уравнением (11.15), как показано на рис. 11.43. Эту исходную комбинационную схему можно реализовать и в других базисах («И-НЕ», «ИЛИ-

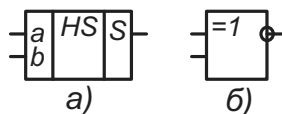


Рис. 11.42

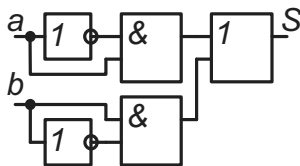


Рис. 11.43

НЕ»), но чаще всего «исключающее ИЛИ» воспринимается как базовый элемент без внутренних подробностей.

11.9.2. «Исключающее ИЛИ», как управляемый инвертор

Одним из возможных применений «исключающего ИЛИ» является управление инверсией выходного сигнала (рис. 11.44). Если управляющий сигнал равен нулю, то, как видно из табл. 11.5, выходной сигнал будет повторять входной. Иначе, на выходе получается инверсия входного сигнала. Эту особенность «исключающего ИЛИ» используют в ПЛИС (см. п. 14.3.2) при выборе типа логической переменной.

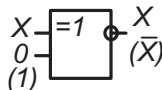


Рис. 11.44

В схеме контроля по четности (см. п. 11.10.3) приведен еще один пример использования полусумматора.

11.9.3. Полный сумматор

имеет уже три входа (a , b и перенос из предыдущего разряда p) и два выхода (сумма S и перенос в следующий разряд P_+). Поэтому число состояний (см. табл. 11.8) расширяется до восьми возможных входных комбинаций – по четыре для $p = 0$ и $p = 1$.

Таблица 11.8

Входы				Выходы	
a	b	p	№	S	P_+
0	0	0	Y_0	0	0
0	1	0	Y_1	1	0
1	0	0	Y_2	1	0
1	1	0	Y_3	1	1
0	0	1	Y_4	1	0
0	1	1	Y_5	1	1
1	0	1	Y_6	0	1
1	1	1	Y_7	1	1

В таблице можно выделить комбинации, когда выходные сигналы единичны, и записать для них логические соотношения входных переменных:

$$S = Y_1 + Y_2 + Y_3 + Y_7 = \bar{a} \cdot b \cdot \bar{p} + a \cdot \bar{b} \cdot \bar{p} + \bar{a} \cdot \bar{b} \cdot p + a \cdot b \cdot p \quad (11.16)$$

$$P_+ = Y_3 + Y_5 + Y_6 + Y_7 = a \cdot b \cdot \bar{p} + \bar{a} \cdot b \cdot p + a \cdot \bar{b} \cdot p + a \cdot b \cdot p \quad (11.17)$$

На основе этих уравнений можно составить комбинационную не минимизированную схему для полного сумматора. Для тренировки ее рекомендуется нарисовать самостоятельно.

УГО полного сумматора приведено на рис. 11.45 и отличается от полусумматора числом выводов и знаком функции SM .

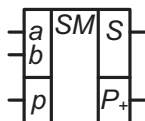


Рис. 11.45

11.9.4. Многоразрядные сумматоры

Параллельный многоразрядный сумматор производит вычисления одновременно над всеми разрядами двух двоичных чисел и реализуется в виде цепочки одноразрядных сумматоров. Схема 4-разрядного сумматора приведена на рис. 11.46.

Основной проблемой при построении сумматоров большой разрядности является их низкое быстродействие. На первый взгляд кажется, что его быстродействие определяется задержкой одноразрядного сумматора, но это не так. Основное запаздывание вносит сигнал переноса, который должен последовательно пройти всю цепочку сумматоров от младших к старшим, непрерывно корректируя сумму, пока не установятся последняя сумма и перенос.

Поэтому цепи переноса стараются делать более быстродействующими, например, уменьшая число элементов в этих цепях, вычисляя перенос не в одном разряде, а сразу в нескольких с учетом всех входных переменных. Такие цепи называют цепями ускоренного переноса.

УГО 4-разрядного сумматора приведено на рис. 11.47, но нужно отметить, что отдельные микросхемы сумматоров на практике используются редко, поскольку подобные операции обычно реализуют при помощи процессоров.

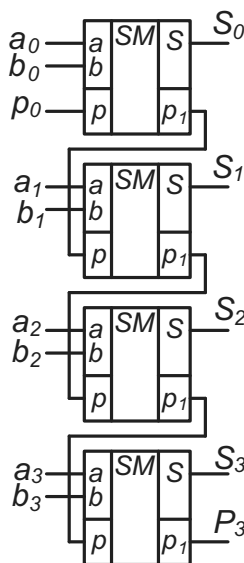


Рис. 11.46

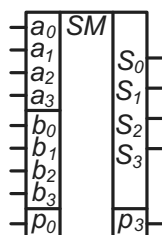


Рис. 11.47

11.9.5. Арифметико-логическое устройство

выполняет арифметические и логические операции обычно над двумя многоразрядными числами, причем тип операции определяется управляющим кодом.

К числу таких операций, кроме суммирования, относится вычитание, (которое обычно реализуется через суммирование числа с обратным или дополнительным кодом другого [24]), инкремент-декремент операнда, поразрядные логические операции и др.

По аналогии с ранее записанными для различных комбинационных устройств логическими уравнениями (11.1) – (11.17), представляющими логические суммы логических произведений над входными переменными можно записать обобщенную функцию АЛУ [18] в виде:

$$F = \sum a_i \cdot b_i \cdot U_i \quad (11.18)$$

где a_i, b_i – разряды двоичных чисел, над которыми выполняют операцию, а U – управляющие сигналы, задающие тип операции.

Разнообразные АЛУ, оптимизированные для конкретных машинных команд, обычно входят в состав процессорных БИС.

§11.10. Узлы контроля

11.10.1. Цифровой компаратор

служит для сравнения двух чисел на равенство. Двоичный разряд имеет всего два значения, «0» и «1», что упрощает реализацию однобитового компаратора, так как равны **или** нулевые значения ($0 = 0$), **или** единичные ($1 = 1$). Причем равенство значений легко зафиксировать при помощи двух логических элементов «И», анализирующих отдельно прямые и инверсные значения сравниваемых разрядов (рис. 11.48).

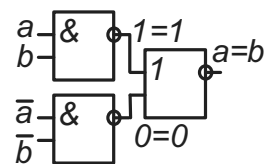


Рис. 11.48

Два двоичных числа A и B равны, если попарно равны значения всех одноименных разрядов a_i и b_i . Поэтому схему компаратора можно реализовать на основе одноразрядных сравнивающих схем, дополненных схемой совпадения «И», вырабатывающей

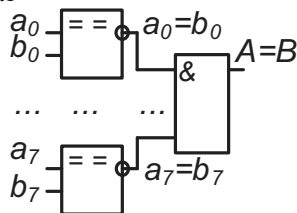


Рис. 11.49

активный сигнал, когда на всех ее входах будет зафиксировано равенство (**и** на первом, **и** на втором...). Для примера на рис. 11.49 приведена схема 8-разрядного компаратора.

Реальные комбинационные схемы компараторов не только анализируют двоичные числа на равенство, но и выявляют какое из чисел больше.

11.10.2. УГО микросхемы цифрового компаратора

В качестве примера такого компаратора рассмотрим УГО (рис. 11.50) микросхемы КР561ИП2, имеющей два входных канала по 4 разряда и три выхода – *равно, больше, меньше*. Очевидно, что только один из выходов может находиться в активном состоянии, так как только одно из условий будет выполняться.

Кроме того, микросхема имеет три входа разрешения для наращивания разрядности (каскадирования), которые должны быть соединены с одноименными выходами младших каскадов. В самой старшей микросхеме активный сигнал подается на вход $E=$, а на два других – пассивные.

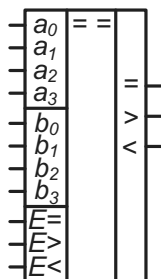


Рис. 11.50

Схемы компараторов входят в состав большинства процессоров. Например, они обеспечивают работу команд условных переходов, сравнение признаков (тэгов) в кэш-памяти, определение границ страниц памяти. Кроме того, в некоторых процессорах компараторы работают в отладочных режимах, контролируя точки останова по заданному адресу.

Во внешних устройствах компараторы могут использоваться при построении программируемых дешифраторов адреса ВУ (см. п. 11.2.5). В этом случае на один вход компаратора подаются сигналы с адресной магистрали, на другой – сигнал с контактного переключателя или энергонезависимого регистра, задающих необходимый код.

Применяются компараторы и в таймерных устройствах ОВМ (см. п. 13.2.5), например, для подключения внешних устройств в заданные моменты времени (простейший пример – будильник).

11.10.3. Схемы контроля по четности

Многие устройства вычислительной техники (процессоры, память, сетевые приемопередатчики) используют схемы встроенного конт-

роля, позволяющие обнаруживать ошибки в работе. Простейшими из них являются схемы контроля по четности, которые подсчитывают количество единиц во всех разрядах контролируемого двоичного числе по модулю два (без учета переносов) и сохраняют результат (чет или нечет) в контрольном разряде P (см. п. 7.4.3)

Например, для двоичного числа «01101110» нетрудно получить $P = 1+1+1+1+1 = 1$.

После передачи числа по линии связи или долговременного хранения контролируемые информационные разряды снова суммируются друг с другом, а сумма по модулю два сравнивается с записанным в контрольном разряде P . По результату сравнения судят об ошибке.

Сумматор по модулю 2 (полусумматор, «ИСКЛЮЧАЮЩЕЕ ИЛИ») был рассмотрен в п. 11.9.1, поэтому нетрудно нарисовать две схемы для суммирования, например, 8 двоичных разрядов:

Первая схема (рис. 11.51), в которой разряды числа суммируются последовательно друг с другом, не только имеет большее число элементов, но и худшее быстродействие, поскольку при изменении кода сигнал частичных сумм проходит через все восемь полусумматоров последовательно.

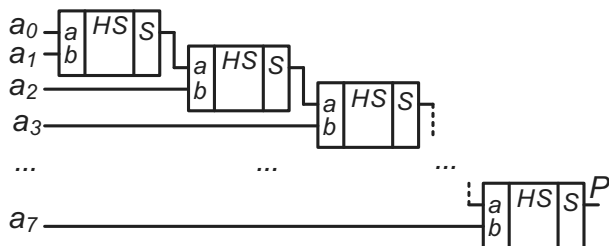


Рис. 11.51

Во второй схеме (рис. 11.52) суммирование осуществляется в группах разрядов, и частичные суммы получаются быстрее, так как сигнал проходит меньшее количество каскадов.

Контроль по четности не обнаруживает двойные ошибки, поэтому более совершенные схемы используют суммирование по модулю три и выше. Развитием подобных схем является метод контрольных сумм, когда массив двоичных чисел снабжается дополнительным числом – контрольной суммой всех чисел определенного формата. При выходе суммы за рамки формата сложение чисел массива продолжается без учета переполнения.

Но в настоящее время чаще используется более сложный и надежный метод контроля, имеющий аббревиатуру CRC (Cyclic

Redundancy Check – циклическая избыточная проверка), который реализуется с использованием регистра сдвига [16] или программным путем.

11.10.4. Искатели старшей единицы

применяются, например, в устройствах прерывания работы процессора для определения номера устройства с наивысшим приоритетом, в арифметических устройствах с плавающей запятой для определения числа сдвигов при нормализации результата и для других целей.

Схему подобного искателя можно реализовать на элементах «И-НЕ» (рис. 11.53) [22]. Выходы искателя – инверсные, то есть старшая единица индицируется нулевым уровнем и служит запретом для младших каскадов. Тогда как на всех остальных выходах будут единичные, пассивные для данного случая, уровни.

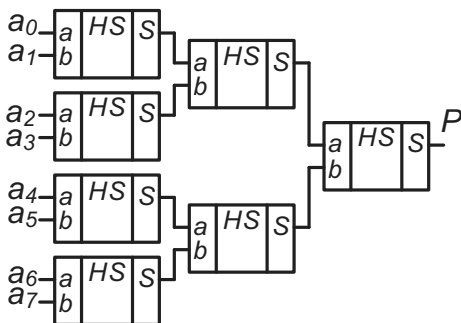


Рис. 11.52

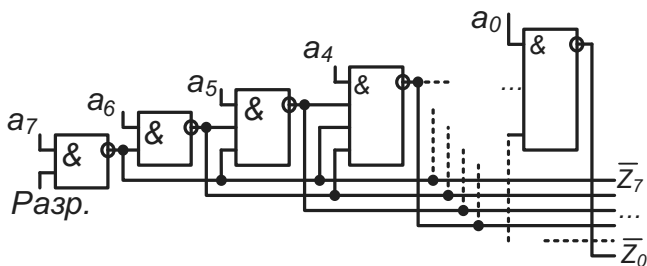


Рис. 11.53

Пусть для примера имеем двоичный код «01100000». Учитывая, что логика работы элементов «И-НЕ» описывается правилом «хотя бы один ноль на входе, на выходе будет единица», можно определить, что сигнал для старшего бита Z_7 будет единичным и пассивным. Тогда на входе логического элемента для шестого разряда встретятся две единицы, то есть $Z_6 = 0$ будет активным и своим нулевым сигналом будет удерживать все младшие элементы в единичном (пассивном) состоянии.

Недостатком рассмотренной схемы является то, что при большой разрядности контролируемого числа логические схемы, анализирующие младшие разряды, должны иметь недопустимо много входов, что затрудняет практическую реализацию. В этом случае применяют каскадное построение искателя старшей единицы, разбивая разряды числа на группы.

11.10.5. Мажоритарный элемент

В ряде случаев бывает необходимо вычислить выходную величину на основании нескольких противоречащих друг другу входных величин. В этом случае применяют принцип «голосования».

Например, на вход последовательного приемника поступает сигнал с помехой. Чтобы повысить надежность, аппаратура приемника осуществляет прием одного последовательного бита несколько раз за период (число опросов обязательно нечетное). Часть отсчетов будет искажена помехой, но на основе остальных можно вычислить правильное значение бита.

Для сведения укажем, что последовательный приемник ОВМ х51 значение каждого принимаемого бита определяет тремя выборками за период длительности бита и вычисляет «голосованием» два из трех (см. п. 8.8.1). Тогда как в ОВМ типа AVR [31] число таких опросов увеличено до 15.

В табл. 11.9 приведен случай трех входных значений, несущих информацию о некоторой переменной, причем одно из значений может быть искажено помехой.

Таблица 11.9

Входы			Выход
F0	F1	F2	F
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Для тренировки можно, используя таблицу, написать для единичных значений выходного сигнала логическую функцию и разработать схему мажоритарного элемента.

УГО мажоритарного элемента для 3 входов, приведенное на рис. 11.54, помечается знаком « ≥ 2 », причем числовое значение символа функции зависит от числа входов и определяет порог голосования.



Рис. 11.54

В некоторых случаях мажоритарный элемент может иметь дополнительные выходы, которые кодируют двоичный номер отказавшей комбинации [30].

12 ТРИГГЕРЫ И РЕГИСТРЫ

§12.1. Триггеры RS-типа

12.1.1. Общие сведения

Давно было замечено, что два усилительных инвертора, соединенных в кольцо (рис. 12.1а), могут долго находиться в одном из двух устойчивых состояниях (то есть выполнять функцию запоминающей ячейки) и переходить из одного состояния в другое под действием входных сигналом лавинообразно (необратимо). Подобное логическое устройство с положительной ОС, имеющее два устойчивых состояния, нулевое и единичное, называют *триггером* (от *trigger* – курок) и относят к *последовательным* схемам с памятью, в которых выходные сигналы определяются не только входными переменными, но и тем, в каком состоянии схема была ранее.

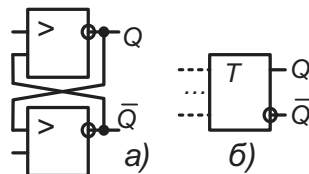


Рис. 12.1

Два инвертора (а значит и сам триггер) имеют два выхода, сигналы которых противофазны. Поэтому у триггера обычно различают два выхода, прямой и инверсный, причем инверсный помечают на УГО кружком (рис. 12.1б).

Установить триггер в «0» или «1» можно, подав входные сигналы на инверторы. В зависимости от логики реагирования на эти сигналы можно скомбинировать множество различных типов триггеров, но на практике используется лишь несколько основных разновидностей триггеров.

12.1.2. Асинхронный RS-триггер

имеет два информационных входа (R и S) и два выхода. Вход S (от *Set* – установка) называют входом установки триггера в единичное состояние, когда $Q=1$, $/Q=0$.

Вход R (*Reset* – предустановка, сброс) выполняет обратную операцию и сбрасывает триггер в начальное, нулевое состояние.

В зависимости от значений двух входных сигналов возможны четыре режима работы RS-триггера:

$S=1, R=0$, режим установки единицы,	$Q=1$
$S=0, R=1$, режим установки нуля,	$Q=0$
$S=0, R=0$, режим хранения информации	$Q=1$ или 0
$S=1, R=1$, недопустимый режим, когда активны оба сигнала.	

RS-триггер с прямыми входами, УГО которого приведено на рис. 12.2, реализуется на основе двух элементов «ИЛИ-НЕ», как показано на рис. 12.3.

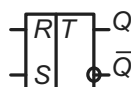


Рис. 12.2

Для пояснения принципа действия проверим режим установки триггера в единицу ($S=1, R=0$). Напомним, что для элемента «ИЛИ-НЕ» можно пользоваться правилом – «если на входе хотя бы одна единица, то на выходе будет ноль, иначе – единица». Поэтому при $S=1$ сигнал $/Q$ будет нулевым, и на входах верхнего элемента встретятся два нуля ($R=0$ и $/Q=0$), что приведет к установке $Q=1$ (значения этих сигналов приведены на рисунке без скобок)

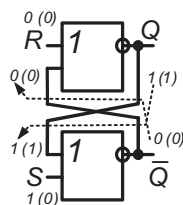


Рис. 12.3

Отметим далее, что на входах нижнего элемента «ИЛИ-НЕ» будут действовать две единицы ($S=1$ и $Q=1$), поэтому, если сигнал S сделать равным нулю (пассивным), то состояние нижнего элемента не изменится. Схема сама будет поддерживать себя в прежнем состоянии (перейдет в режим хранения). Значения сигналов в этом случае приведены на рисунке в скобках.

Запрещенная комбинация на входах триггера, когда оба входа активны, не приводит к трагическим последствиям. Просто в этом случае выходной сигнал триггера становится неопределенным. И прямой, и инверсный выходы одновременно будут иметь нулевой уровень.

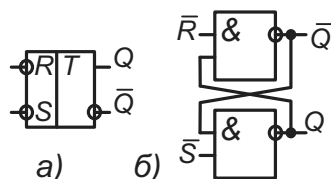


Рис. 12.4

УГО RS-триггера с инверсными входами и его реализация на элементах «И-НЕ» приведены на рис. 12.4.

Рассмотренные элементарные RS-триггеры широко используются на практике как в составе более сложных триггеров, регистров, элементов памяти, так и автономно, например, для старт-стопных операций (см. п. 13.4.3) или предварительной обработки сигнала с дребезгом от контактного ключа.

12.1.3. Применение RS-триггера для подавления дребезга

При использовании контактных ключей обязательно нужно помнить про дребезг контактов. Это явление заключается в том, что при переключении ключа (рис. 12.5а) из-за колебательной реакции упругого элемента или изменений контактного сопротивления происходит некий переходный процесс (рис. 12.5б), длящийся примерно 5-20 мс и нарушающий нормальное восприятие сигнала.

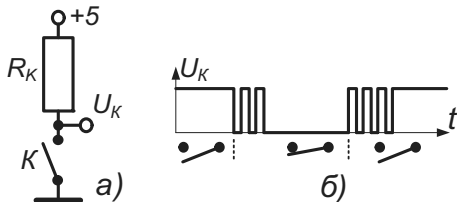


Рис. 12.5

Игнорирование дребезга контакта осуществляют как программно (см. п. 10.1.4), так и аппаратно, например, подключая к выходу интегрирующую цепочку (см. п. 1.2.2) для сглаживания переходного процесса. Но самым надежным аппаратным способом подавления дребезга является использование RS-триггера, например с инверсными входами. Но в этом случае вместо обычного ключа приходится использовать более сложный перекидной ключ (рис. 12.6).

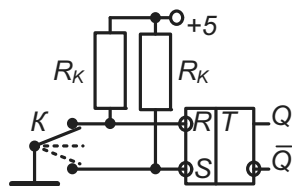


Рис. 12.6

Для пояснения воспользуемся временными диаграммами, приведенными на рис. 12.7 и отражающими наличие переходных процессов на входах триггера из-за дребезга контактов. Активными для входов триггера являются нулевые уровни. Возникающий переходный процесс, например, по входу S, когда сигнал имеет то активный уровень, то пассивный (перекидной ключ в среднем положении), приводит к тому, что триггер переключается из режима установки «1» в режим хранения и выходной сигнал при переключении не изменяется.

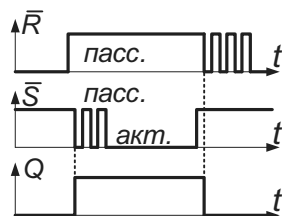


Рис. 12.7

12.1.4. Тактируемый уровнем RS-триггер

имеет (рис. 12.8) кроме входов R и S тактирующий вход C . Если синхросигнал C активен (единичен в данном случае), то триггер работает, как описано ранее, то есть может устанавливаться в «0» и «1» или хранить прежнее состояние.

Если сигнал C пассивен, то триггер не реагирует на входные сигналы и работает в режиме хранения. Временные диаграммы работы тактируемого уровнем триггера приведены на рис. 12.9. Отметим, что при пассивном синхросигнале, входные сигналы могут быть одновременно активными, но на работе это не скажется.

Вход C может быть и инверсным, тогда его вход на УГО помечен кружком, что означает восприятие входной информации триггером при $C=0$.

Тактируемый уровнем триггер реализуется на основе асинхронного триггера и двух логических ключей (см. п. 11.8.2), управляемых синхросигналом (рис. 12.10).

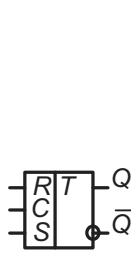


Рис. 12.8

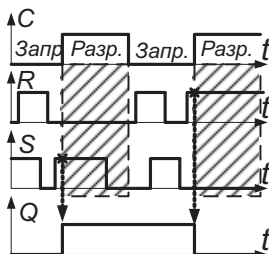


Рис. 12.9

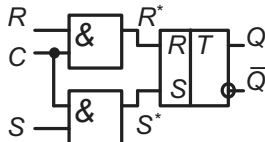


Рис. 12.10

12.1.5. Тактируемый фронтом (перепадом) RS-триггер

обозначение которого приведено на рис. 12.11, реагирует на входную информацию только в момент тактирующего перепада, положительного или отрицательного, остальное время триггер хранит ранее записанное состояние.

Временные диаграммы для RS-триггера, тактируемого положительным перепадом, приведены на рис. 12.12.

Реализовать такой триггер можно, например, на основе двух RS-триггеров, тактируемых противофазными уровнями (рис. 12.13). В момент перепада происходит перенос состояния Q , входного триггера в выходной триггер. В остальное время независимо от значения C изменение состояния выходного триггера не происходит.

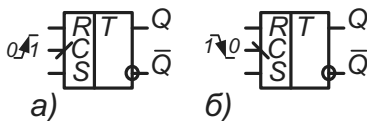


Рис. 12.11

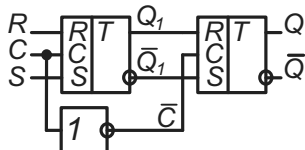


Рис. 12.13

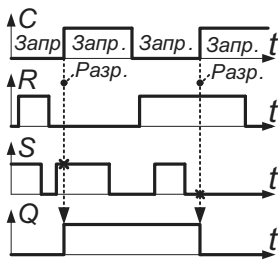


Рис. 12.12

12.1.6. RS-триггеры с приоритетными входами

не имеют запрещенной комбинации входных сигналов.

S-триггер в основном работает как обычный RS-триггер, то есть устанавливается по входам R и S в ноль и единицу. Однако, он работоспособен в режиме, недопустимом для обычного триггера, когда активны оба входа. В этом случае приоритет имеет вход S, то есть, когда активны оба входа, триггер устанавливается в «1».

R-триггер работает с точностью до наоборот, то есть при обоих активных входах, устанавливается в ноль.

E-триггер (Exclusive) в подобном случае не изменяет своего состояния.

Еще один триггер (JK-типа, см. п. 12.3.3), не имеющий запрещенной комбинации на входах, подобных входам R и S, будет рассмотрен далее.

§12.2. Триггеры D-типа

12.2.1. Тактируемый уровнем D-триггер

имеет один информационный вход данных D, причем сигнал на выходе триггера повторяет сигнал на входе D, то есть при $D=0$, триггер устанавливается в ноль, при $D=1$ – в единицу.

Сразу отметим, что асинхронные D-триггеры на практике не используются, поэтому D-триггер обязательно имеет и тактирующий вход C.

Если разрешения по входу C нет, то триггер хранит ранее запомненное состояние. После прихода на вход C разрешающего уровня

или перепада триггер воспринимает значение D , то есть устанавливается в «0» или «1».

Обозначения D -триггеров с прямым и инверсным входами тактирования приведены на рис. 12.14.

Временные диаграммы для D -триггера, тактируемого единичным уровнем, приведены на рис. 12.15,

Триггеры D -типа легко реализуются на основе RS -триггеров, тактируемых уровнем, и инвертора, образующего сигнал R для сброса в ноль (рис. 12.16).

Триггер D -типа является, пожалуй, самым распространенным триггером регистровых устройств памяти, поскольку он имеет только один информационный вход, что упрощает топологию схем по сравнению с RS -триггерами, имеющими двойной (парафазный) информационный вход.

Выпускаются D -триггеры и в виде микросхем, например, КР155ТМ5 и ТМ7, которые имеют по четыре триггера в одном корпусе.

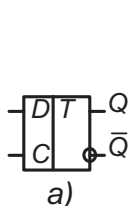


Рис. 12.14

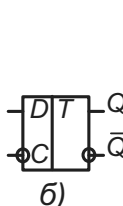


Рис. 12.15

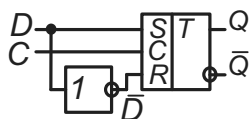
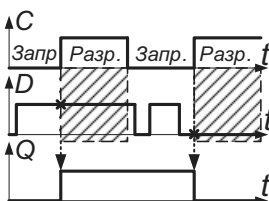


Рис. 12.16

12.2.2. Тактируемый фронтом триггер D -типа

Он также имеет два входа, информационный D и тактирующий C (рис. 12.17), но переброс триггера возможен только в момент разрешающего перепада тактирующего сигнала. Временные диаграммы для триггера с положительным разрешающим перепадом приведены на рис. 12.18.

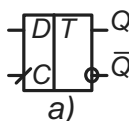


Рис. 12.17

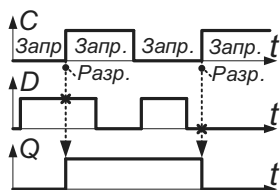
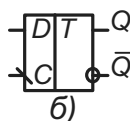


Рис. 12.18

Подобные *D*-триггеры могут реализоваться, либо, как и прежде, на основе *RS*-триггера, тактируемого фронтом (см. рис. 12.16), либо по другим схемам.

12.2.3. Микросхема триггера *D*-типа, тактируемого фронтом

Одной из до сих пор применяемых на практике микросхем является КР155ТМ2, содержащая в одном корпусе два *D*-триггера, тактируемых положительным перепадом.

Обозначение микросхемы показано на рис. 12.19 и отличается от рассмотренных ранее наличием *инверсных* установочных входов R_y и S_y , которые действуют независимо от информационного и тактирующего сигналов.

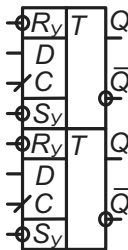


Рис. 12.19

§12.3. Триггеры Т- и JK-типов

12.3.1. Асинхронный триггер Т-типа

имеет подобно *D*-триггеру один информационный вход *T*, но другую логику работы. Он реализует так называемый счетный принцип и перебрасывается в противоположное состояние при каждом разрешающем перепаде сигнала на входе *T* (от слова *Toggle* – переключатель). Тактирующий перепад, как и прежде, может быть положительным (рис. 12.20а) или отрицательным (рис. 12.20б).

Из временных диаграмм, приведенных на рис. 12.21 для счетного триггера с отрицательным разрешающим перепадом, нетрудно видеть, что период выходного сигнала получается в два раза больше входного периода. Поэтому для выходной частоты справедливо соотношение

$$f_{\text{ВЫХ}} = 1 / T_{\text{ВЫХ}} = 1 / 2T_{\text{ВХ}} = f_{\text{ВХ}} / 2, \quad (12.1)$$

из которого следует, что счетный триггер делит входную частоту ровно пополам. Это замечательное свойство широко используется

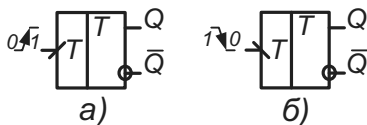


Рис. 12.20

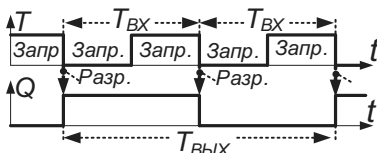


Рис. 12.21

на практике, например, в делителях частоты, в том числе и программируемых (см. §13.2). Кроме того счетный триггер является естественной основой суммирующих и вычитающих счетчиков (см. §13.1).

Асинхронный T -триггер может быть реализован на базе RS - или D -триггеров, тактируемых фронтом (рис. 12.22), путем введения перекрестных обратных связей для реализации T – алгоритма.

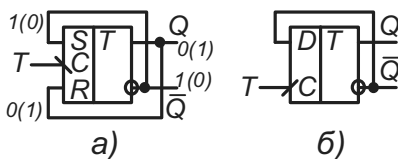


Рис. 12.22

Для примера поясним работу T -триггера на основе триггера RS -типа (рис. 12.22а). Пусть триггер находится в исходном состоянии, то есть $Q=0$, а $\bar{Q}=1$. Эти сигналы, показанные на рисунке без скобок, через перекрестную обратную связь формируют входные сигналы $S=\bar{Q}=1$, $R=Q=0$. Следовательно, триггер по входам R и S будет подготовлен к установке в единицу, поэтому при отрицательном перепаде на входе T триггер будет установлен в «1», то есть перейдет в противоположное состояние. При этом значения сигналов на входах S и R показанные в скобках, как бы поменяются местами. Поэтому триггер опять будет готов к переходу в противоположное состояние. Аналогично реализуется счетный триггер на триггерах D -типа (рис. 12.22б).

Рассмотренный T -триггер является, по сути, асинхронным, хотя использует тактируемые фронтом синхронные RS - или D -триггеры, поскольку его переброс происходит в момент изменения информационного сигнала (асинхронно) и не связан с синхронным импульсом.

12.3.2. Синхронный счетный триггер

имеет управляемую перекрестную связь (рис. 12.23), работа которой разрешается сигналом на входе T . А момент переброса триггера будет определяться сигналом синхронизации, то есть рассмотренный триггер по принципу действия является синхронным.

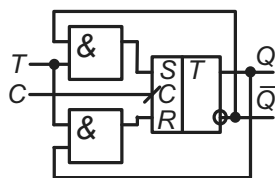


Рис. 12.23

Если $T = 0$, то логические ключи не будут пропускать сигналов перекрестной обратной связи и сигналы на входах S и R будут пассивными. Поэтому триггер будет хранить прежнее состояние, не реагируя на тактовые импульсы.

Если $T = 1$, то логические ключи восстановят действие перекрестной обратной связи, подготавливая триггер к переходу в противоположное состояние. Момент переброса определяется только тактирующим перепадом C .

12.3.3. JK-триггер

имеет два информационных входа J и K (рис. 12.24) и чаще всего выполняется тактируемым отрицательным или положительным перепадом.

Этот триггер в определенной степени является универсальным, так как может моделировать работу ранее рассмотренных триггеров и в первую очередь RS -триггера. Например, вход J соответствует входу установки в единицу S , вход K – входу сброса R .

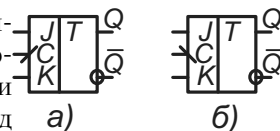


Рис. 12.24

Однако RS -триггер имеет запрещенную входную комбинацию, когда на обоих входах действуют активные уровни сигналов. JK -триггер в этом случае при $J = 1$, $K = 1$ перебрасывается в противоположное состояние, то есть работает подобно счетному T -триггеру.

Реализуется JK -триггер на основе синхронного счетного триггера, приведенного на рис. 12.24, в котором счетный вход T расщепляется и образует два входа J и K .

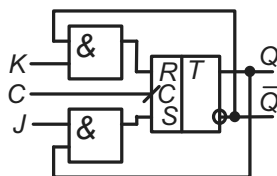


Рис. 12.25

Таким образом, из полученной схемы (рис. 12.25) видно, что вход J воздействует на вход S , вход K на вход R . Когда $J = 1$ и $K = 1$ (входы как бы объединены), реализуется схема, подобная синхронному счетному триггеру. Самостоятельно проверить на конкретных кодах алгоритм работы JK -триггера.

12.3.4. Микросхема JK триггера

Реальные JK -триггеры обычно имеют несколько дополнительных входов. Например, микросхема JK -триггера КР155ТВ1 (рис. 12.26) имеет три входа J и три входа K , поступающих на внутренние схемы логического умножения (говорят, что сигналы J и K объединяются по «И», то есть сигнал $J = 1$ (или $K = 1$) для триггера формируется, когда на всех трех входах J (или K) действуют единичные сигналы.

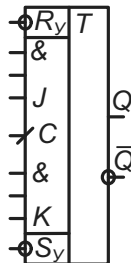


Рис. 12.26

Такое «излишество» позволяет в ряде случаев упростить практическую схему устройств, используя внутренние возможности триггера.

Кроме информационных и тактирующих входов микросхема триггера имеет два установочных входа S_y и R_y . Они инверсные и асинхронные (действуют независимо от тактирующих и информационных сигналов), поскольку воздействуют сразу на выходную триггерную ячейку.

Триггер JK-типа, как и счетный, может использоваться в делителях частоты и счетчиках, причем он в силу своей универсальности позволяет реализовать устройства без дополнительных внешних элементов. Например, JK-триггер можно остановить (заставить только хранить информацию), можно установить в «1» или в «0», или перебросить в противоположное состояние.

§12.4. Регистры памяти

12.4.1. Общие сведения

Регистры памяти служат для хранения двоичных кодов, выполнения некоторых операций над ними и представляют собой группу триггеров с общим сигналом синхронизации, число которых и определяет разрядность хранимого числа. В дальнейших примерах для простоты будем ограничиваться 4 или 8 разрядами.

При реализации регистров памяти можно применять любые тактируемые триггеры, кроме счетных T -типа. Но для уменьшения потребляемой мощности и задержек стараются использовать самые простые – тактируемые уровнем триггеры RS - и D -типа. Причем чаще всего регистр памяти строят на триггерах D -типа (рис. 12.27), которые имеют только один информационный сигнал, что упрощает разводку схем.

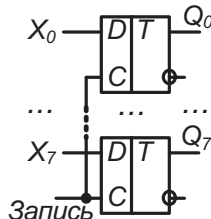


Рис. 12.27

Если использовать в регистре двухходовой RS -триггер, то для записи двоичного числа понадобится в два раза больше информационных линий, так как для установки триггера в «0» или в «1» необходимы отдельные сигналы R и S .

Сигнал «ЗАПИСЬ» управляет приемом информации в триггеры. Когда он пассивен (в нашем случае нулевой), триггеры не реагируют на входные сигналы и хранят ранее записанный двоичный код.

Когда сигнал «ЗАПИСЬ» на короткое время становится активным, все триггеры принимают двоичные значения входных переменных ($Q_0 = X_0, \dots, Q_7 = X_7$) для последующего хранения. Такой режим часто называют параллельным занесением информации.

Простейшие микросхемы регистров памяти могут предлагаться в виде наборов D-триггеров с общими или частично объединенными цепями синхронизации. Например, микросхемы КР155ТМ5 и ТМ7 [14] представляют собой набор из четырех триггеров (4-разрядный регистр) с попарно объединенными цепями синхронизации. Но такие малоразрядные регистры в настоящее время почти не применяются на практике.

В регистрах можно использовать и более сложные триггеры, тактируемые перепадом (см. п. 12.2.2), что удобно в некоторых случаях, поскольку не нужно формировать короткий сигнал записи. Однако нужно помнить, что такие триггеры по своей структуре являются двухступенчатыми и поэтому имеют меньшее быстродействие и большее энергопотребление.

12.4.2. Обозначение регистра памяти

оформляется, как всегда, в виде прямоугольника с полями (рис. 12.28). В левом поле отмечаются входы, в правом – выходы. В центральном поле помещают знак регистра RG .

Тактирующий вход C , который иногда называют WR (от $WRIte$), бывает прямой или инверсный. Если используется тактирование фронтом (положительным или отрицательным), то такой синхровход помечают наклонной чертой. Отметим, в справочниках не всегда соблюдается это требование, поэтому тип синхронизации следует уточнять в тексте.

Регистр памяти имеет, как правило, и некоторые дополнительные входы управляющие приемом или выдачей информации.

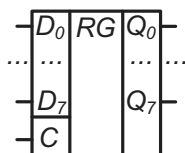


Рис. 12.28

12.4.3. Управление приемом информации

Прием кода от общей информационной шины может происходить, например, по какому-то условию. Поэтому помимо собственно тактирующего входа некоторые регистры имеют один или несколько разрешающих сигналов (рис. 12.29). Эти сигналы объединяются с тактирующим на внутренних элементах «И» (говорят – объединяются по «И») и обычно помечаются буквами EI ($Enable Input$) или

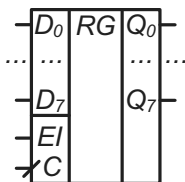


Рис. 12.29

другими. Таким образом, запись кода в регистр будет возможна при наличии активных разрешающих сигналов и в момент прихода синхросигнала.

12.4.4. Управление выдачей информации

рассмотрим на примере 8-разрядного регистра КР580ИР82 (рис. 12.30) из микропроцессорного комплекта серии КР580. Кроме синхровхода C , определяющего момент приема кода, регистр ИР82 имеет *инверсный* вход OE (*Output Enable* – выход разрешен), который управляет только выходными каскадами регистра и не влияет на информацию, записанную в триггерах. Заметим, что этот сигнал иногда называют RD (от *ReaD*).

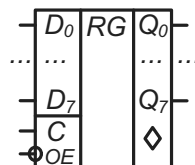


Рис. 12.30

Следует отметить, что регистр ИР82 тактируется уровнем, что не всегда приемлемо. Альтернативой подобному регистру являются регистры с синхронизацией фронтом.

Чтобы пояснить управление выдачей информации, воспользуемся структурой регистра, приведенной на рис. 12.31. Между триггерами и выходным зажимами включены буферные повторители сигнала с третьим состоянием, о чем говорит знак $\diamond$ на их обозначении. Когда сигнал OE активен (в данном случае равен нулю), на выходные зажимы регистра пропускаются сигналы с выходов триггеров. Когда сигнал OE пассивен (единичен), повторители переводят выходы регистра в третье состояние (см. п. 2.3.7), как бы обрывая выходные выводы регистра.

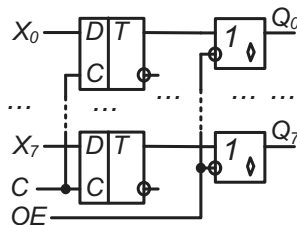


Рис. 12.31

Наличие третьего состояния позволяет, используя «монтажное ИЛИ» (см. п. 11.8.6), подключать выходы регистра к микропроцессорной магистрали и в нужные моменты времени, когда формируется активный сигнал OE , передавать процессору свою информацию (см. п. 14.5.1).

*Некоторые регистры имеют расширенный набор управляющих сигналов, включающий, кроме сигналов разрешения приема или выдачи кода, например, вход R , позволяющий производить асинхронную установку всех триггеров регистра в нулевое состояние, вход CS (*Chip Select* – выбор микросхемы), разрешающий действие информационных и тактовых сигналов, и ряд других управляющих входов.*

§12.5. Регистры сдвига

12.5.1. Общие сведения

Операция сдвига заключается в перемещении в регистре всех разрядов двоичного кода на одинаковое число разрядов влево или вправо. При этом информация, которая выходит за пределы разрядной сетки пропадает. На первое освободившееся место заносится информация с последовательного входа регистра.

Сдвиг используется в процессорных умножителях (для сдвига частичных произведения), в сетевых адаптерах (для преобразования параллельного кода в последовательный) и во множестве других устройств (см. §§12.6–12.8).

Для построения регистра сдвига обязательно нужно использовать триггеры, тактируемые перепадом (фронтом).

12.5.2. Регистр сдвига вправо

реализуется на основе *тактируемых фронтом D-триггеров*, входы которых соединены с выходами предыдущих. Далее для простоты будут рассмотрены 4-разрядные варианты регистров (рис. 12.32).

При этом вход самого первого триггера часто называют последовательным входом регистра *SI* (*Serial Input*) или входом для сдвига вправо *DR* (*Data Right*). Выход самого последнего триггера иногда именуют последовательным выходом *SO* (*Serial Output*). Этот выход также может быть использован для увеличения разрядности регистра сдвига.

В момент разрешающего перепада (в нашем случае положительного) сдвигового сигнала *C* каждый триггер воспринимает сигнал с предыдущего триггера и происходит *одновременный* перенос (сдвиг) вправо записанного ранее кода. Причем значение, хранившееся в последнем триггере, при сдвиге выходит за пределы регистра и пропадает, если не принять мер к его сохранению. С другой стороны в первый триггер, освободившийся от предыдущего значения, записывается сигнал с входа *SI*.

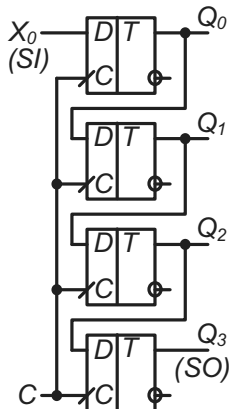


Рис. 12.32

Обозначение регистра сдвига вправо показано на рис. 12.33 и имеет, кроме символов *RG*, знак выполняемой функции в виде стрелки, направленной вправо.

Изменив порядок следования триггеров в цепочке, можно реализовать регистр для сдвига влево, соединив входы с выходами последующих триггеров.

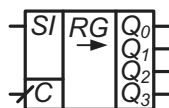


Рис. 12.33

Нужно отметить, что пользователям не приходится самим соединять триггеры для регистров сдвига. Большинство серий логических элементов имеют в своем составе много разновидностей регистров сдвига, из которых можно подобрать подходящий. При необходимости можно использовать и реверсивный регистр, в котором сдвиг может производиться влево или вправо в зависимости от управляющих сигналов.

12.5.3. Реверсивный регистр сдвига

также представляет собой цепочку триггеров (рис. 12.34). Только на вход каждого триггера информация поступает от логического коммутатора на элементах «2И-ИЛИ» (см. п. 11.8.3), который управляется противофазными сигналами *L* (от *Left*), *R* (от *Right*).

Один из входов коммутатора подсоединен к выходу *предыдущего* триггера, другой – к выходу *последующего* триггера. Поэтому, формируя сигналы управления коммутатором, можно организовать передачу информации от предыдущего триггера к последующему (сдвиг вправо), а можно и в обратном направлении (сдвиг влево).

Последовательных входа в реверсивном регистре два. Один *DR* служит для приема информации при сдвиге вправо (сигнал с предыдущего каскада), другой *DL* воспринимает данные при сдвиге влево (сигнал с последующего каскада). Эти входы используются при наращивании разрядности за счет каскадного включения регистров.

В некоторых реверсивных регистрах коммутатор имеет допол-

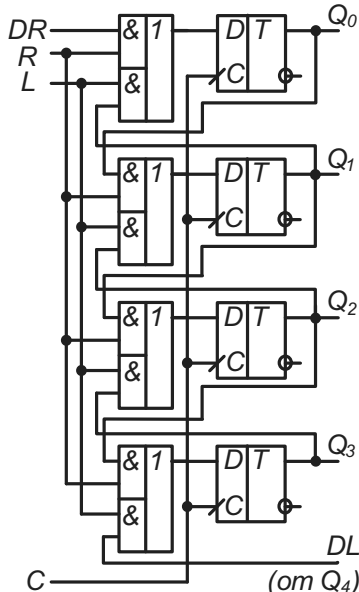


Рис. 12.34

нительный третий вход так называемого параллельного занесения (как в регистре памяти), для начальной установки регистра от внешнего источника двоичного кода. Поэтому для формирования только одного из управляющих сигналов коммутатора регистр может иметь внутренний дешифратор, тогда режим работы регистра будет определяться двоичным управляющим кодом (см. далее).

12.5.4. Микросхема реверсивного регистра КР155ИР13,

которую рассмотрим в качестве примера, является 8-разрядной. На УГО реверсивного регистра (рис. 12.35), помеченное двусторонней стрелкой, можно выделить вход тактирования C , вход общего сброса триггеров R , информационные входы $D_0 \dots D_7$ для начальной установки произвольного кода.

Два режимных входа M_0, M_1 , управляющие (через встроенный дешифратор) внутренними 3-ходовыми коммутаторами перед триггерами, служат для задания одного из 4 режимов работы регистра (сдвиг влево, сдвиг вправо, параллельное занесение сигналов $D_0 \dots D_7$, хранение информации).

На рис. 12.36 приведен пример каскадного включения микросхем реверсивного регистра для наращивания разрядности. Получившееся 16-разрядное устройство по-прежнему может воспринимать параллельные данные и работать в режиме сдвига, как влево, так и вправо.

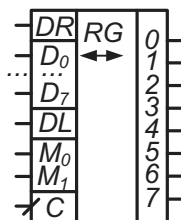


Рис. 12.35

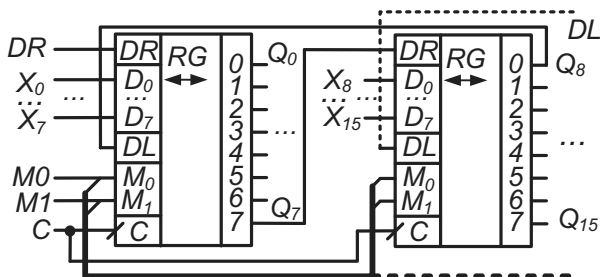


Рис. 12.36

12.5.5. Регистровая память типа очередь

обычно обозначается аббревиатурой *FIFO* (*First In First Out* – первым вошел, первым вышел). Код, который первым был записан, первым появляется на выходе.

Принцип действия FIFO знаком каждому студенту по поговорке «В одно ухо влетает, в другое вылетает»...

Применение они находят, например, в быстрых сетях в качестве буфера для приема-передачи данных. При этом упрощается взаимодействие с памятью. Дело в том, что для обычной памяти процессор должен установить адрес, прочитать данные, вычислить следующий адрес, проверить, не выходит ли он за пределы буфера.

Память *FIFO* является для внешних устройств по принципу действия безадресной, поскольку данные заносятся и извлекаются в порядке очереди, что повышает скорость обмена.

Кроме того, буфер типа FIFO удобно использовать для согласования разноскоростных сетевых устройств. Например, высокоскоростное устройство без задержек заносит принятый пакет в буфер, а низкоскоростное по мере надобности извлекает данные, не снижая общей производительности. Или наоборот, программа, не торопясь, подготавливает данные в буфере, а затем сетевой адаптер с высокой скоростью передает их в линию связи.

Подобный алгоритм работы буфера обычно реализуется аппаратными средствами на основе различных запоминающих устройств и схем управления в виде счетчиков-указателей [23, 30]. Могут применяться для оперативного буферирования данных и регистры сдвига.

Например, на рис. 12.37 показана подобная иллюстративная регистровая структура на 16 байтов. После 16 импульсов сдвига порция информации с последовательных входов X будет занесена в регистры. Подавая следующие импульсы сдвига в необходимом темпе можно получить эти данные на выходах Q .

Программная реализация принципа очереди (буфер со сдвигом) описана в п. 9.9.2.

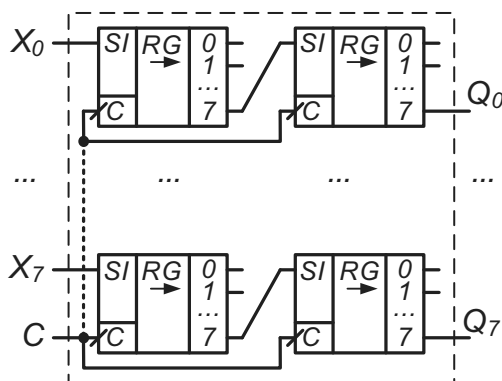


Рис. 12.37

Области применения регистров сдвига очень обширны. Кроме приемопередатчиков последовательного кода, имеющих во всех сетевых устройствах, регистры сдвига также используют-

ся в вычислителях контрольной последовательности CRC, микропрограммных автоматах [30], в генераторах псевдослучайного двоичного кода [16] и т. д.

§12.6. Регистры сдвига в асинхронном приемопередатчике

Асинхронный последовательный формат (см. п. 8.1.4) широко используется на практике. Поэтому асинхронные приемопередатчики на основе регистров сдвига, выполняющих преобразование параллельного кода в последовательный и обратно, входят в состав большинства процессорных семейств.

12.6.1. Передатчик асинхронного последовательного порта

На рис. 12.38 приведена упрощенная схема передатчика асинхронного последовательного порта на основе 10-разрядного регистра сдвига влево, поскольку в асинхронной посылке биты идут, начиная с младшего.

Информационный байтовый код X_{BX} обычно предварительно записывается при помощи сигнала $WR1$ во вспомогательный регистр хранения, где данные ожидают готовности (опустошения) регистра сдвига. Затем из регистра хранения двоичный код копируется в регистр сдвига для передачи при помощи сигнала $WR2$.

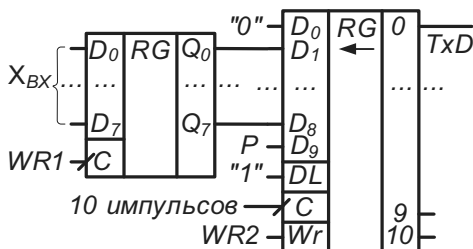


Рис. 12.38

Обратим внимание на то, что информационные данные (и при необходимости бит четности P) в регистре помещаются между нулевым стартовым и единичным стоповым битами, причем стартовый бит D_0 появляется на выходе в момент подачи $WR2$. Далее не показанный на схеме автомат управления формирует 10 импульсов сдвига, в результате чего на выходе TxD поочередно появятся все биты двоичного кода (см. рис. 8.1). С последним сдвиговым импульсом на выходе (и во всех разрядах регистра) установится единичный стоповый бит, определяющий исходное состояние линии связи.

12.6.2. Программная передача асинхронной посылки

В некоторых случаях аппаратный последовательный порт ОВМ может быть занят другим связным процессом и приходится реализовать последовательную передачу битов программным путем. Поскольку ранее подобная процедура была оформлена на ассемблере (см. п. 8.1.4), ниже приводится ее вариант на языке Си.

Вначале программа должна передать нулевой стартовый бит, сделать паузу длительностью $T_{\text{в}}$, затем в цикле поочередно, начиная с младшего, передать с паузой восемь информационных битов и в заключение установить единичный стоповый бит, который определяет исходное состояние.

Для большей компактности выходного бинарного кода при организации цикла в нижеприведенной подпрограмме для примера использован оператор **do-while** с проверкой условия в конце цикла:

```
void mProcSendA (byte mDat) {
    byte mCount = 8;           // Инициализация счетчика битов.
    EA = 0;                     // Запрет прерываний, чтобы не удлинялись паузы.
    mPinTx = 0;                 // Старт-бит.
    mDelay (mTb);               // Пауза для стандартной частоты
    do {                         // Цикл приема 8-ми битов.
        mPinTx = mDat & 1;      // Передача бита, младший – первый.
        mDat >>= 1;             // Готовим следующий бит.
        mDelay(mTb);
    } while(--mCount);          // Все биты приняты?
    mPinTx = 1;                 // Стоп-бит.
    mDelay(mTb);
    EA = 1;                     // Разрешение прерываний.
}
```

Обратить внимание на то, что при передаче происходит преобразование байта в бит (см. п. 9.6.5).

12.6.3. Приемник асинхронного последовательного порта

также может быть реализован на основе регистра сдвига влево (рис. 12.39), на последовательный вход *DL* которого поступают биты асинхронной посылки, начиная со стартово-

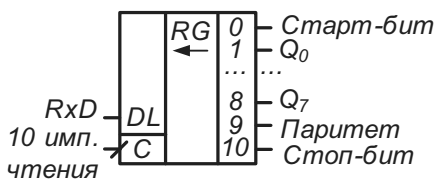


Рис. 12.39

го. Устройство управления приемником, рассмотрение которого выходит за рамки пособия, реагирует на стартовый перепад и, подстраиваясь под середину периода T_b , формирует 10 импульсов сдвига. В результате этого все последовательные биты, включая служебные, оказываются сохраненными в сдвиговом регистре. Служебные биты анализируются на предмет ошибок, а информационные могут быть скопированы обслуживающей программой.

12.6.4. Программный прием асинхронной посылки

Ниже приведен вариант ассемблерной программы (см. п. 8.2.4), оформленный на языке Си. Программа должна ожидать стартовый перепад, периодически опрашивая вывод порта *mPinRx*, затем сделать паузу для выхода на середину периода T_b и в цикле принимать последовательные информационные биты:

```
byte mProcReadA (void) {
    byte mCount = 8, mR = 0; // Счетчики битов и результата.
    while (mPinRx);          // Ждем стартовый перепад.
    EA = 0;                  // Запрет прерываний, чтобы они не изменяли паузы.
    mDelay (mTb / 2);        // Подстройка на середину периода.
    do {                      // Цикл приема 8 битов.
        mDelay (mT0);
        if (mPinRx) mR = mR | 128; // Прием на место старшего
        mR = mR >> 1;           // бита с учетом сдвига вправо.
    } while (--mCount);        // Все биты приняты?
    EA = 1;                   // Разрешение прерываний.
    return mR;
}
```

Данная процедура является иллюстративной, реальные подпрограммы используют механизм и выводы внешнего прерывания (см. §10.4), и поэтому текст реальной программы будет несколько отличаться от приведенной ниже, например, отсутствием команды ожидания стартового перепада, на который должна реагировать система прерываний.

Как уже отмечалось, рассмотренные процедуры чаще всего выполняют аппаратные передатчик и приемник последовательного порта, обслуживание которых рассмотрено в §10.10 и §10.11.

§12.7. Регистр сдвига в синхронном клавиатурном интерфейсе

12.7.1. Передача битов в синхронном виде

Асинхронный формат предъявляет определенные требования к одинаковости частоты приемника и передатчика. В некоторых случаях, чтобы освободить удаленное устройство от необходимости отсчитывать интервалы времени и упростить аппаратуру, используют формируемый одним из устройств дополнительный синхронизирующий сигнал, который сопровождает последовательные информационные данные и разрешает чтение бита. Однако определенной платой за это является большее число соединяющих проводов.

Синхронные последовательные интерфейсы широко используются для связи ОВМ с внешними микросхемами (ПЗУ, АЦП, цифровыми часами, карточками памяти) и миниатюрными модулями (радиомодемами, сетевыми адаптерами). В настоящее время наиболее распространенными интерфейсами синхронного типа являются интерфейсы SPI (см. п. 12.8.1) и I²C [25].

До сих пор на практике используется и один из первых синхронных интерфейсов – клавиатурный, который имеет тактирующий *Clock* и информационный *Data* сигналы (рис. 12.40). Каждый информационный бит в этом интерфейсе сопровождается синхросигналом, причем тактирующий перепад подается примерно в середине передаваемого бита, что позволяет отстроиться от переходных процессов в линии связи и сделать схему независимой от изменения частоты сигналов. Отметим также, что сигналы клавиатурного интерфейса инверсные.

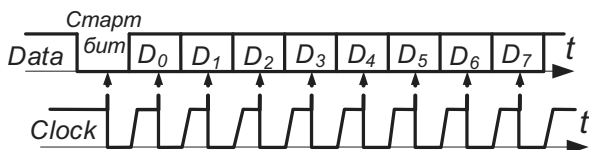


Рис. 12.40

12.7.2. Приемник сигналов от клавиатуры

В компьютере необходимо преобразовать последовательную синхронную посылку от клавиатуры в параллельный формат и обработать ее. В свое время для этой цели в первых компьютерах типа *IBM XT* использовались две микросхемы, 8-разрядного регистра сдвига и D-

триггера, которые образовывали регистр для приема стартового и 8 информационных битов. На рис. 12.41 приведен аналог такой цепочки в виде единого 9-разрядного регистра, который сбрасывается в исходное нулевое состояние сигналом по входу *R* от центрального процессора.

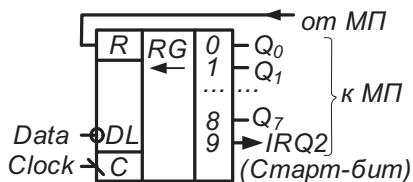


Рис. 12.41

Информационные сигналы *Data* поступают на инверсный последовательный вход регистра сдвига и под действием тактирующего сигнала *Clock* заполняют все разряды регистра. Причем, как только стартовый бит заносится в 9 разряд, являющийся запросом на прерывание с системным номером *IRQ2*, начинает работать программа-драйвер компьютера, которая читает принятый байт, затем обнуляет регистр и выполняет другие действия.

Отметим, что клавиатурный интерфейс является двунаправленным [6] и может использоваться для передачи команд от компьютера, например, для управления индикаторами и установки параметров клавиатуры.

12.7.3. Программная передача синхронной посылки

осуществляется процессором клавиатуры. Как видно из временных диаграмм (рис. 12.40) изменения информационного сигнала могут происходить при нулевом тактирующем сигнале.

При единичном тактирующем сигнале данные должны оставаться неизменными. Кроме того, отметим, что в синхронном интерфейсе старший бит иногда передается первым, хотя в некоторых случаях очередность может быть и обратной.

Программа в цикле устанавливает очередной бит и формирует синхросигнал, длительность которого должна быть больше времени заряда паразитной емкости линии связи:

```
void mProcSendK (byte mOut) {
    byte mCount = 8;
    mPinData = 0;
    mDelay5mks;
    mPinClock = 0;
    mDelay5mks;
    do {
        // Стам-бит.
```

```

    mPinData = mOut & 1; // Младший бит передается первым.
    mOut = mOut >> 1;    // Готовим следующий бит.
    mPinClock = 1;
    mDelay5mks;          // Пауза для заряда паразитной
                        // емкости на линии.

    mPinClock = 0;
    mDelay5mks;
} while (--mCount);
mPinClock = 1;          // Исходное состояние
}

```

12.7.4. Программный прием синхронной посылки

Иногда сигнал от клавиатуры может обрабатываться программно. В этом случае необходимо дождаться стартового перепада на линии данных, а затем в цикле, отслеживая синхроимпульсы, читать очередной бит:

```

byte mProcReadK () {
    byte mCount = 8, mBt = 0;
    while (mPinData); // Ждем стартовый перепад.
    while (mPinClock); // Пропускаем старт бит
    do {
        while (!mPinClock); // Ждем начало синхросигнала
        mBt = mBt >> 1;     // Готовим место для бита.
        If (mPinData) mBt = mBt | 128; // На старшее место
                                    // с учетом сдвига.
        while (!mPinClock); // Ждем окончание синхросигнала
    } while (--mCount);
    return mBt;
}

```

§12.8. Регистр сдвига в дуплексном синхронном интерфейсе типа SPI

При синхронной обмене можно одновременно производить прием данных в передающий регистр, то есть работать в дуплексном режиме.

12.8.1. Приемник и передатчик интерфейса SPI

Как и в клавиатурном интерфейсе, момент приема информационного бита определяет синхросигнал (рис. 12.42), который действует одновременно на регистры сдвига двух устройств, главного (*Master*)

и подчиненного (*Slave*), которые соединены в кольцо и тактируются общим генератором (рис. 12.43). В этом случае каждый из регистров одновременно выполняет функции и передатчика и приемника последовательного кода.

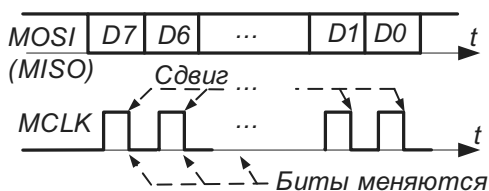


Рис. 12.42

Выходной сигнал главного устройства обозначается, как *MOSI* (**M**aster **O**utput **S**lave **I**nterface), и из расшифровки аббревиатуры видно, что он является одновременно входным сигналом для подчиненного. В описаниях микросхем могут встречаться альтернативные названия этого сигнала – *DO*, *SDO*, *DOUT*.

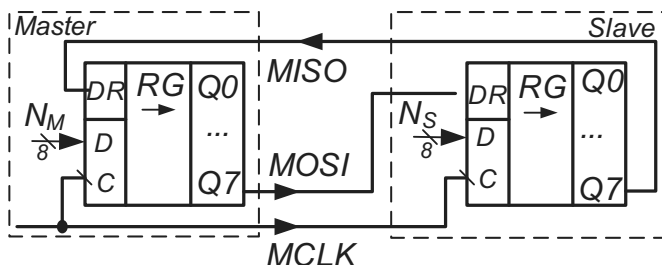


Рис. 12.43

Для подчиненного устройства в обозначении его выходного сигнала *MISO* (**M**aster **I**nterface **S**lave **O**utput) кодируется одновременно тип сигнала для главного. Альтернативные названия этого сигнала – (*DI*, *SDI*, *DIn*) не столь конкретны и на практике возможна путаница.

Главное устройство в принципе может работать с несколькими подчиненными, для этого используются дополнительные сигналы разрешения. Кроме того отметим, что стартового импульса во временной диаграмме нет.

Из рис. 12.43 видно, что последовательные выходы (от старших разрядов) регистров подчиненного подключаются к последовательным входам, принимающим сигналы для младших триггеров регистров. Таким образом, образуется единый 16-разрядный регистр сдвига, в котором под действием тактовых импульсов происходит сдвиг исходного кода главного устройства через линию *MOSI* в регистр подчиненного. Одновременно по линии *MISO* идет прием кода из подчиненного. Таким

образом, после 8 тактовых импульсов происходит обмен байтами между регистрами.

Главным преимуществом интерфейса *SPI* является то, что он может работать в дуплексном режиме, то есть передача и прием битов осуществляются одновременно и под управление общих синхроимпульсов.

Отметим, что некоторые процессоры имеют аппаратно реализованные и программно управляемые (через регистры области SFR) синхронные интерфейсы типа SPI [31], разновидности которого широко используются на практике в различных устройствах (ЦАП, АЦП, сетевых микросхемах), в мультимедиа картах памяти типа MMC или SD.

Отметим также, что подобный механизм дуплексного последовательного обмена на основе кольцевого многоразрядного регистра сдвига использован в специализированном интерфейсе JTAG [30], который позволяет производить внутрисистемное тестирование и отладку.

12.8.2. Пример программы приема-передачи для SPI

В тех случаях, когда ОВМ не имеет аппаратного интерфейса типа *SPI*, его приходится реализовывать программным путем. Ниже приведена дуплексная процедура одновременной передачи и приема для главного устройства из руководства для компилятора *SDCC* [32]:

```
byte mProcOutInSPI(byte mOut) {
    byte mI = 8;
    do {
        mPinMOSI = mOut & 128;    // Передача очередного бита
        mOut = mOut << 1;
        mPinMCLK = 1;
        mDelay5mks;                // Для заряда паразитной емкости
        if (mPinMISO) mOut = mOut + 1; // Прием очередного бита
        mPinMCLK = 0;
    } while(--mI);
    return mOut;
}
```

Следует обратить внимание на другой метод приема бита – не логическое, а арифметическое сложение.

13 СЧЕТЧИКИ

Счетчик под действием входных импульсов перебирает свои состояния в определенном порядке. Чаще всего перебор соответствует нормальному двоичному коду, но необязательно. Счетчик, у которого код возрастает, называют суммирующим, у которых убывает – вычитающим. Если направление счета можно изменить, то это реверсивный счетчик.

§13.1. Двоичные счетчики

13.1.1. Асинхронный двоичный счетчик

представляет собой цепочку последовательно соединенных триггеров T -типа (рис. 13.1), то есть входным сигналом для триггера является выходной предыдущего триггера.

Рассмотрим работу такой схемы, построив временные диаграммы для случая четырех разрядов (рис. 13.2) с учетом того, что T -триггер перебрасывается в противоположное состояние при каждом тактирующем перепаде (в нашем случае отрицательном) входного сигнала. При этом, как было отмечено ранее, счетный триггер делит частоту в два раза (см. п. 12.3.1), что приводит к постепенному понижению частоты работы каждого последующего триггера.

Таким образом, чем дальше триггер от входа счетчика, тем реже он перебрасывается и как бы больше вес (значимость) его разряда в счетчике. Это рассуждение можно подтвердить, если использовать формулу для перевода двоичного кода в десятичный:

$$N = Q_0 * 1 + Q_1 * 2 + Q_2 * 4 + Q_3 * 8 + \dots \quad (13.1).$$

Подставим сюда значения двоичных разрядов из временных диаграмм в определенный момент времени, например, после прихода 12 счетных импульсов, когда $Q_0 = 0$, $Q_1 = 0$, $Q_2 = 1$, $Q_3 = 1$. Для этого случая можно получить $N = 0 * 1 + 0 * 2 + 1 * 4 + 1 * 8 = 12$, то есть двоичный код счетчика точно соответствует числу входных импульсов.

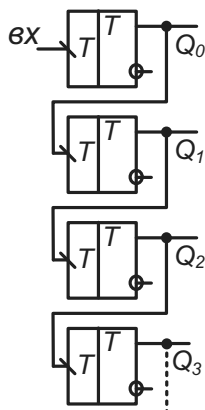


Рис. 13.1

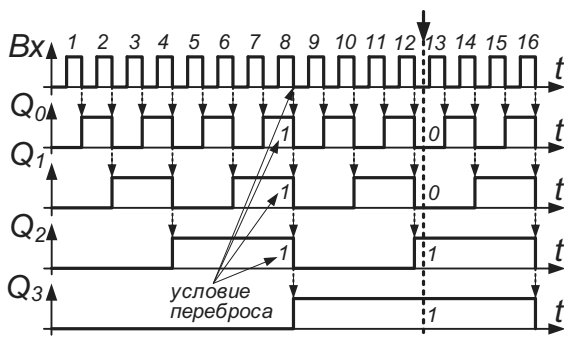


Рис. 13.2

Рассмотренная схема двоичного счетчика является по своему принципу асинхронной, так как триггеры в ней перебрасываются асинхронно и последовательно друг за другом. Новый счетный импульс на вход можно подать только после того, как перебросится последний триггер цепочки.

Эта особенность снижает частотный диапазон для входного сигнала и делает его зависящим от числа разрядов счетчика. Кроме того, при таком последовательном перебросе триггеров возникают кратковременные двоичные комбинации, не соответствующие количеству сосчитанных импульсов.

Например, читается код счетчика после прихода 12 импульса в момент, когда перебросились три первых триггера ($Q_0 = 0$, $Q_1 = 0$, $Q_2 = 1$), а четвертый — нет ($Q_3 = 0$), то есть вместо кода «12» будет прочитан ошибочный код «4».

От этих недостатков свободны синхронные счетчики, в которых счетный импульс воздействует одновременно на все триггеры (см. п. 13.1.3).

Примечание. Чтобы получить вычитающий счетчик, также нужна цепочка счетных триггеров, только используются уже не прямые, а инверсные выходы триггеров (см. п. 13.1.4).

13.1.2. Обозначение асинхронного счетчика

приведено на примере 4-разрядной микросхемы K555IE5 (рис. 13.3). Отличием этого счетчика от приведенного на рис. 13.1 является на-

личие двух счетных входов. Это связано с тем, что выход первого триггера микросхемы не имеет связи с остальными, что позволяет при необходимости изменять коэффициент пересчета. Например, для организации 4-разрядного счетчика нужно соединить выход Q_0 с входом второго триггера C_2 .

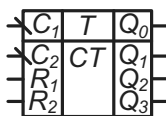


Рис. 13.3

Входы R_1 и R_2 , объединенные внутри микросхемы по «И», служат для сброса всех триггеров в нулевое состояние, для чего нужно подать на оба входа единичные сигналы. Для работы микросхемы в режиме счета импульсов нужно, чтобы хотя бы один из сигналов сброса был нулевым.

Для наращивания емкости счетчика применяют каскадное соединение микросхем, когда выход младшей соединяют с входом старшей. Но нужно помнить, что при увеличении числа разрядов асинхронного счетчика снижается максимальная частота входного сигнала. Этого недостатка лишены синхронные счетчики.

13.1.3. Синхронный двоичный счетчик

Для пояснения принципа действия синхронного счетчика необходимо обратить внимание на условие переброса, указанное стрелками на временных диаграммах рис. 13.2. Триггер перебрасывается в противоположное состояние, когда все предыдущие триггеры находятся в единичном состоянии, и происходит отрицательный перепад счетного импульса.

Поэтому синхронные счетчики имеют дополнительные цепи анализа условия переброса (переноса). Эти цепи реализуются на основе логических элементов «И» (рис 13.4), выходы которых управляют входами JK -триггеров. Если на всех предыдущих триггерах будет «1», то $J = K = 1$ и с приходом счетного импульса будет разрешен переброс триггера в противоположное состояние. Иначе – триггер будет хранить прежнее состояние.

В этой схеме счетный импульс подается на все синхровходы одновременно. Поэтому триггеры, которым

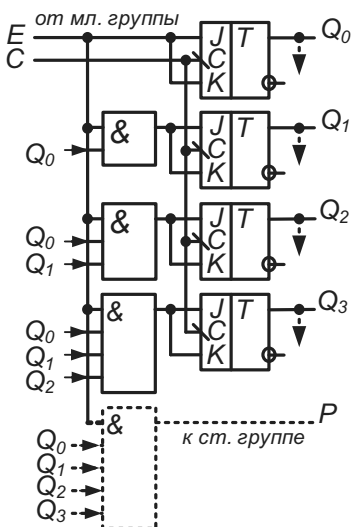


Рис. 13.4

разрешен переброс, переходят в противоположное состояние также одновременно, то есть отсутствует основной недостаток асинхронного счетчика – последовательное переключение триггеров в цепочке. Благодаря этому синхронные счетчики имеют максимально возможное быстродействие.

Однако рассмотренная схема с так называемым параллельным переносом (анализом) при построении многоразрядных счетчиков имеет и недостаток, связанный с тем, что цепи анализа старших разрядов должны быть многоходовыми.

Чтобы уменьшить число входов элемента «И», можно использовать результат предыдущего элемента, то есть перейти к счетчику с так называемым последовательным переносом. Но при этом снижается быстродействие – новый счетный импульс можно подавать, когда сигнал переноса пройдет через всю цепочку для анализа.

Компромиссным вариантом являются устройства с групповым переносом, в которых счетчики делятся на группы триггеров. В каждой из групп реализуется схема с параллельным анализом условий переброса и имеется разрешающий вход от младшей группы. А разрешающий выход P на старшую группу триггеров образуется на отдельной логической схеме анализа, которая показана пунктиром на рис. 13.4.

13.1.4. Асинхронный реверсивный счетчик

Реверсивные счетчики могут работать как на суммирование, так и на вычитание. Режим суммирования обеспечивается, как уже показано, путем соединения прямого выхода триггера со счетным входом. А режим вычитания получается, если использовать для счета инверсные выходы цепочки триггеров.

В реверсивных счетчиках для смены направления счета используют, как и в реверсивных регистрах сдвига, коммутаторы на входе триггеров. На один из входов коммутатора поступает прямой выход предыдущего триггера, используемый в режиме суммирования. На другой – инверсный выход, обеспечивающий режим вычитания. Но при смене направления счета в асинхронном реверсивном счетчике происходит неинформационное изменение кода. Поэтому реверсивные счетчики строятся только по синхронному принципу.

13.1.5. Синхронный реверсивный счетчик

с одной стороны должен, как синхронный, иметь на входе каждого триггера цепи анализа переноса, а с другой, как реверсивный – ком-

мутатор для изменения направления счета, управляемый двумя счетными сигналами.

На рис. 13.5 приведен 4-разрядный вариант такого счетчика с использованием JK -триггеров. На информационные входы триггеров поданы непоказанные единичные сигналы, разрешающие перебор в противоположное состояние при наличии разрешающего перепада синхроимпульса управляемого цепью анализа.

Двухканальный коммутатор на входах каждого триггера выполнен на элементах «И-ИЛИ» (см. п. 11.8.3), логические элементы «И» которого одновременно используются для анализа состояния предыдущих триггеров (условия переброса триггера), как для прямых, так и инверсных выходов.

Управляющие сигналы коммутатора (+1, -1) в то же время являются счетными, то есть импульсы поступают на синхровходы триггеров *почти* одновременно, поскольку быстродействие коммутаторов несколько различается. Поэтому синхронность данной схемы в определенной степени условная, но с этим мирятся.

При увеличении разрядности счетчика число входов логических элементов «И» увеличивается, поэтому в этом случае, как и ранее, используют разделение счетчика на отдельные каскады. Для каскадирования предназначены элементы «И», показанные на рисунке пунктиром. Их инверсные выходные сигналы $P+$ и $P-$ выделяют два состояния счетчика. Первое состояние, когда инверсный сигнал $P+$ активен (все триггеры установлены в «1» – код равен 15) разрешает работу старшей группы счетчика при суммировании. Второе состояние, когда активен $P-$ (единичны все инверсные выходы – код равен нулю), разрешает работу при вычитании.

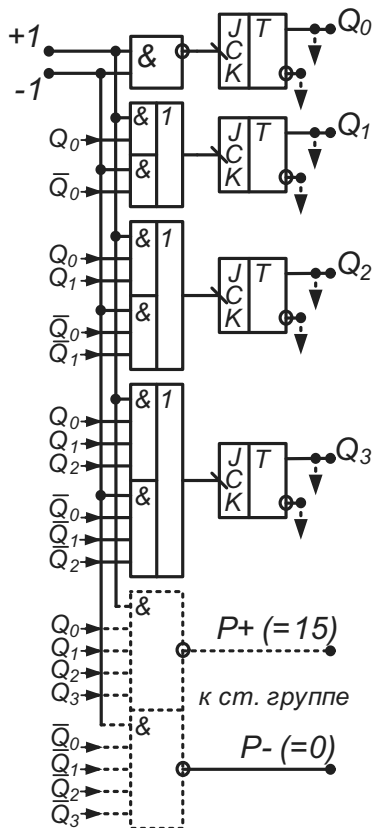


Рис. 13.5

13.1.6. УГО синхронного реверсивного счетчика

рассмотрим на примере микросхемы КР155ИЕ7 (рис. 13.6). В отличие от схемы, приведенной на рис. 13.5, счетчик имеет дополнительные входы $D_0 \dots D_3$ для параллельного занесения кода в триггеры, как в регистре памяти. Поэтому коммутаторы на входах триггеров выполнены трехканальными, причем третий канал управляется инверсным входным сигналом WR , который действует независимо от счетных входов и в режиме счета должен быть пассивным.

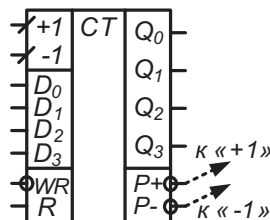


Рис. 13.6

Кроме того микросхема имеет вход R для общего сброса всех триггеров.

Инверсные выходы группового переноса $P+$ и $P-$ обычно используются при каскадировании. Пунктиром показаны соединения для подключения другой микросхемы при наращивании разрядности. Но в некоторых случаях выходы переноса могут использоваться при построении счетчиков с программируемым коэффициентом пересчета (см. п.п. 13.2.3 и 13.2.4).

§13.2. Счетчики с программируемым коэффициентом пересчета

13.2.1. Счетчики с двоично-взвешенным коэффициентом пересчета

в виде цепочки счетных триггеров (осуществляющие деление входной частоты в 2, 4, 8 и т. д. раз) широко используются в предварительных делителях частоты для тактирования цифровых таймеров. В качестве примера рассмотрим предделитель подобный, используемому в процессорном семействе AVR [31], на основе 10-разрядного двоичного счетчика с коммутатором (рис. 13.7).

На вход счетчика поступает сигнал F_{KB} от внутреннего генератора процессора, на выходах счетчика образуются сигналы пониженной частоты, поскольку каждый из счетных триггеров делит частоту ровно в два раза. Временные диаграммы подобны приведенным на рис. 13.2.

Но не все выходы счетчика используются, поскольку 8-канальный коммутатор, управляемый тремя битами **CSi** (**C**lock **S**elect) одного из

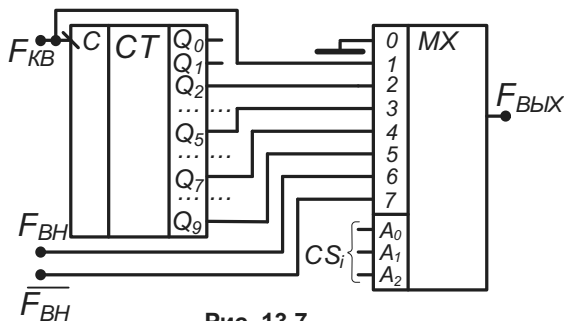


Рис. 13.7

специальных регистров ОВМ, позволяет получить только восемь возможных комбинаций. В предделителе процессора AVR такими комбинациями являются – останов (тактового сигнала нет), F_{KB} , $F_{KB}/8$, $F_{KB}/64$, $F_{KB}/256$ и $F_{KB}/1024$, внешний сигнал F_{BH} прямой и инверсный. Необходимая частота тактирования таймеров задается битами **CSi** при программной инициализации.

Но в большинстве практических случаев требуется более плавное управление коэффициентом пересчета.

13.2.2. Программируемый инкрементный счетчик с асинхронным сбросом

имеет устройство для непрерывного контроля текущего состояния двоичного счетчика, в качестве которого можно использовать, например, дешифратор (рис. 13.8). Под действием входного сигнала F_{BX} счетчик перебирает свои состояния в порядке возрастания кодов, то есть работает в режиме суммирования (инкремента или прямого счета). На соответствующих выходах дешифратора будут появляться

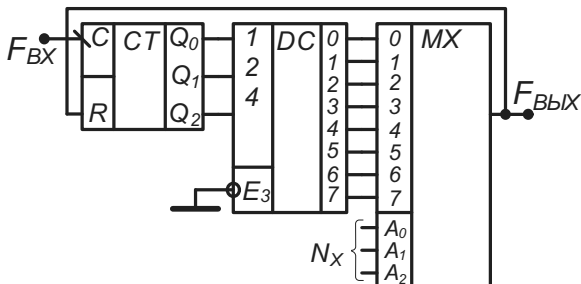


Рис. 13.8

активные сигналы. Но при помощи программирующего коммутатора будет выбран только один канал, адресуемый кодом N_x .

Поэтому, когда состояние счетчика станет равно заданному N_x , появится сигнал на выходе коммутатора, который сбросит счетчик в исходное нулевое состояние (рис. 13.9).

Таким образом, при помощи дешифратора и коммутатора ограничивается число возможных состояний схемы, то есть коэффициент пересчета (емкость счетчика) можно изменять. Выбор коэффициента осуществляется при помощи выбора соответствующего выхода дешифратора. Если не требуется программно управлять коэффициентом деления, то можно исключить коммутатор и использовать неполный дешифратор, как например, в двоично-десятичных счетчиках (см. п. 13.3.1).

Следует отметить, что в рассмотренной схеме нужно использовать синхронный счетчик, потому что асинхронный имеет в моменты переброса ложные кодовые состояния (см. п. 13.1.1).

Из временной диаграммы, приведенной на рис. 13.9, следует, что выходной период пропорционален коду, что бывает удобно устройств, формирующих интервал времени. Однако выходная частота

$$F_{\text{ВЫХ}} = 1 / T_{\text{ВЫХ}} = 1 / (N_x \cdot T_{\text{ВХ}}) = F_{\text{ВХ}} / N_x \quad (13.2),$$

оказывается обратно пропорциональной коду N_x , что не всегда желательно.

13.2.3. Программируемый инкрементный счетчик с предустановкой

позволяет получить примерно прямую зависимость частоты от заданного кода. В этом случае прямой счет идет до максимального возможного значения, после чего в счетчик загружается код N_x (рис. 13.10).

Подобную схему реализуют на основе счетчиков (рис. 13.11), имеющих входы для параллельного занесения информации, например, можно

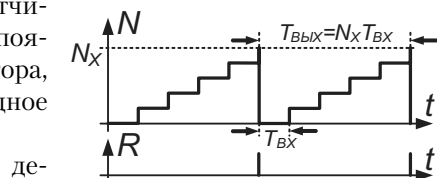


Рис. 13.9

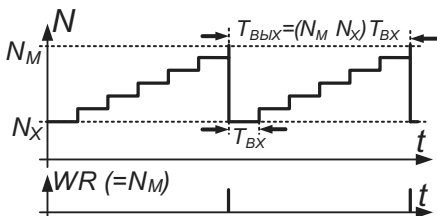


Рис. 13.10

использовать ранее рассмотренную микросхему ИЕ7 (см. п. 13.1.6), которая является синхронной (то есть без ложных состояний при перебросе) и имеет выход неполного дешифратора $P+$ для наращивания разрядности, являющийся индикатором максимального значения кода ($=15$ – все единицы). Отметим, что счетный вход для режима вычитания в этом случае не используется.

Для данной схемы

$$F_{\text{ВЫХ}} = 1 / T_{\text{ВЫХ}} = 1 / ((N_M - N_X) \cdot T_{\text{ВХ}}) = F_{\text{ВХ}} / ((N_M - N_X)) \quad (13.3),$$

то есть при увеличении кода N_X выходная частота тоже увеличивается. Нелинейность же этой зависимости на практике почти не мешает, так как нужный код можно вычислить предварительно или в самой программе.

Примечания. Вход WR может быть синхронным, тогда период выходных импульсов увеличится на один такт. Подобный принцип используется в таймере T1 OBM x51 для тактирования последовательного порта (см. п. 8.9.2).

13.2.4. Программируемый декрементный счетчик с предустановкой

использует *вычитающий* режим (обратного счета) реверсивного счетчика типа ИЕ7 с параллельным занесением данных (рис. 13.12).

В исходном состоянии с входов $D_0 \dots D_3$ в счетчик занесен код N_X . Затем под действием счетных импульсов значение кода начинает уменьшаться (рис. 13.13). И в момент, когда содержимое счетчика станет нулевым, сигнал отрицательного переполнения $P-$ (являющийся выходом неполного дешифратора, реагирующего на нулевое состояние), воздействуя на вход WR , вызовет повторную установку кода N_X в триггеры счетчика.

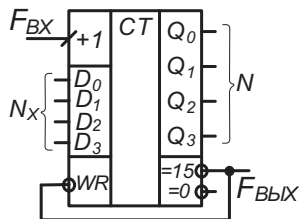


Рис. 13.11

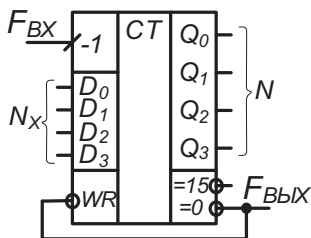


Рис. 13.12

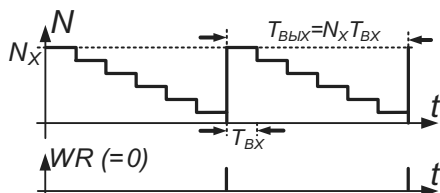


Рис. 13.13

Таким образом, как видно из временных диаграмм, приведенных на рис. 13.13, можно изменять коэффициент деления рассмотренной схемы. Период выходных импульсов в этом случае пропорционален коду, а выходная частота обратно пропорциональна и по-прежнему описывается формулой (13.2).

Отметим, что подобную структуру имели декрементные синхронные счетчики каждого из трех каналов программируемого таймера Intel 8253 (аналог КР580ВИ53), который использовался в качестве системного таймера в первых персональных компьютерах.

13.2.5. Программируемый счетчик с цифровым компаратором

В настоящее время более универсальным устройством для контроля текущего кода счетчиков является цифровой компаратор (см. п. 11.10.1), который позволяет использовать режим, как прямого, так и обратного счета без дополнительных дешифратора или коммутатора.

Обратный счет, как уже отмечалось, удобен для формирования временных интервалов, пропорциональных входному коду. Прямой счет позволяет использовать таймер для измерения длительности импульсов, но при формировании интервалов зависимость не пропорциональная. Но это обходят программно.

Упрощенная схема приведена на рис. 13.14. Временные диаграммы для прямого и обратного счета в основном похожи на рассмотренные ранее. Подобные устройства обычно реализуют в составе таймеров внутри ОВМ, которые имеют так называемый режим «компарирования», позволяющий формировать импульсы с программируемой длительностью [31].

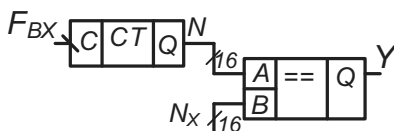


Рис. 13.14

§13.3. Двоично-десятичные и часовые счетчики

13.3.1. Двоично-десятичные счетчики

Иногда на практике необходимо обеспечивать счет в двоично-десятичном формате, более удобном для восприятия человеком. Счет при

этом происходит на основе декадных счетчиков, которые имеют 10 двоичных состояний. Нетрудно заметить, что для реализации декады можно использовать 4-разрядные двоичные счетчики (состояния от 0 до 15), подобные рассмотренным выше, приняв меры для исключения лишних в этой ситуации 6 состояний.

Чаще всего исключают последние по порядку комбинации (рис. 13.15), чтобы сохранить нормальную двоичную кодировку. С этой целью к счетным триггерам добавляют неполный дешифратор с выходом Y_{10} , при помощи которого декада обнуляется после прихода 10 импульса подобно счетчику с асинхронным сбросом (см. п. 13.2.2).



Рис. 13.15

13.3.2. Микросхема асинхронного двоично-десятичного счетчика КР155ИЕ2,

обозначение которой приведено на рис. 13.16, выполнена на основе четырех счетных триггеров. Причем первый триггер имеет отдельный вход C_1 и выход Q_0 , не соединенный с остальными. Это подчеркивается графически и отдельным знаком функции CT_2 , означающим, что его коэффициент пересчета равен двум.

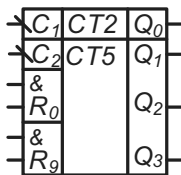


Рис. 13.16

Три других триггера образуют *асинхронную* счетную цепочку со встроенным неполным дешифратором, который обнуляет эти триггеры после прихода 5 импульса. Таким образом, коэффициент пересчета этой части схемы равен 5 (состояния триггеров от 0 до 4), что отражено в знаке функции CT_5 .

Соединив выход Q_0 с входом C_2 можно получить двоично-десятичную декаду, имеющую десять состояний (от 0 до 9).

Другой особенностью микросхемы ИЕ2 является наличие, кроме обычных входов общего сброса R_0 , входов R_9 – установки в состояние 9.

13.3.3. Микросхема синхронного реверсивного двоично-десятичного счетчика КР155ИЕ6

имеет УГО (рис. 13.17) полностью совпадающее с обозначением двоичного счетчика ИЕ7 (см. п. 13.1.6).

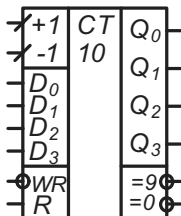


Рис. 13.17

Счетчик обладает двумя *синхронными* счетными входами $+1$ и -1 , четырьмя входами $D0..D3$ для параллельной установки, входом управления записью WR и входом R общего сброса триггеров.

Только знак выполняемой функции $CT10$ показывает, что число его состояний укорочено до 10. Поэтому выход переноса для режима суммирования соответствует состоянию 9.

13.3.4. Микросхема асинхронного часового счетчика КР155ИЕ4

в свое время была самой простой и универсальной основой для построения цифровых часов. Дело в том, что для реализации цифровых часов требуются, счетчики секунд, минут и часов с емкостью 60 (счет секунд или минут) и 24 (счет часов).

Как видно из УГО, приведенного на рис. 13.18, микросхема ИЕ4 имеет 4 триггера, причем первый триггер ($CT2$) цепочки (см п. 13.1.1) не связан с остальными, что позволяет при необходимости изменять коэффициент пересчета.

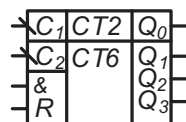


Рис. 13.18

Три последних триггера имеют, что показано на УГО знаком $CT6$, укороченный коэффициент пересчета, равный шести (состояния от 0 до 5). Это позволяет реализовать, привлекая минимум внешних схем, счет часов ($2 \cdot 6 = 12$ или $2 \cdot 2 \cdot 6 = 24$) и счет секунд или минут ($6 \cdot 10 = 60$).

13.3.5. Схема счетчика секунд или минут,

имеющая состояния от 0 до 59, реализуется с использованием двух счетчиков с емкостью 6 и 10, в качестве которых можно использовать микросхемы ИЕ4 и ИЕ2 (рис. 13.19). На вход двоично-десятичной микросхемы ИЕ2 с емкостью счета 10, выполняющей роль счетчика единиц, подаются секундные (или минутные) импульсы.

Включенная затем микросхема ИЕ4 считает уже десятки секунд, причем использует только часть ее с коэффициентом пересчета 6. Оставшийся триггер может пригодиться,

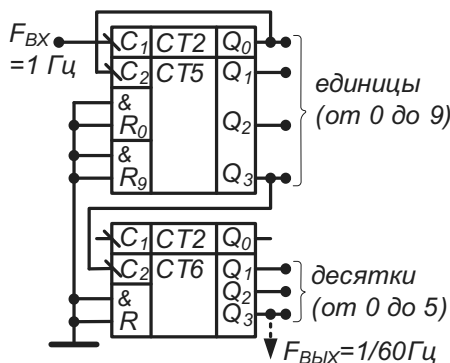


Рис. 13.19

например, в счетчике часов, в котором также может использоваться микросхема ИЕ4 (12·2=24).

Таким образом, схема цифровых часов должна иметь генератор секундных импульсов, счетчики секунд, минут и часов, снабженные при необходимости индикацией. В настоящее время цифровые часы реального времени (RTC – Real Time Clock) реализуются как в виде отдельных микросхем, так и в составе процессорных БИС. Иногда используется и программная реализация цифровых часов (см. п. 10.4.7).

13.3.6. Микросхема цифровых часов с параллельным интерфейсом

KP512ВИ1 (аналог *МС14818*), УГО которой показано на рис. 13.20, предназначена для работы в качестве цифровых часов в составе микропроцессорных систем и использовалась в самых первых персональных компьютерах.

Секундный генератор в микросхеме выполнен на основе встроенного генератора с внешним кварцем (обычно с так называемой часовой частотой 32768 Гц), подключаемым к выводам X_{IN} , X_{OUT} (см. п. 6.8.3) и триггерной счетной цепочки, понижающей частоту до 1 Гц.

Для программы микросхема доступна через параллельный интерфейс, подключаемый к процессорным шинам (см. п. 14.7.3) и представляется набором пронумерованных регистров, которые хранят секунды, минуты, часы, дни, месяцы и годы.

Проблема 2000 года, кстати, была связана с тем, что для счета годов был выделен один байт – сэкономили память...

Параллельный интерфейс имеет совмещенные (мультиплексированные) линии адреса/данных $AD_0..AD_7$, причем установка адреса должна сопровождаться синхросигналом AS , а данных – DS . Сигнал R/W служит для указания типа обмена (чтение или запись), а CS является инверсным входом выбора микросхемы, который нужно подключать к дешифратору банка памяти (см. п. 11.2.4).

Сигнал на инверсном выходе IRQ формируется автоматически при обновлении информации в ячейках или при срабатывании будильника и используется системой для временных отметок в качестве запроса на прерывание. Отметим также, что некоторые выводы микросхемы для простоты не показаны.

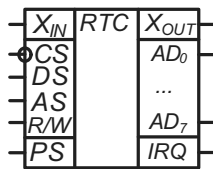


Рис. 13.20

13.3.7. Микросхема цифровых часов с последовательным интерфейсом

В настоящее время в микропроцессорных системах используют в основном часы с более экономичным последовательным интерфейсом. В качестве примера таких часов рассмотрим 8-ножечную микросхему *INA8583* (аналог *PCF8583*), выпускаемую в Белоруссии, УГО которой приведено на рис. 13.21.

Эта микросхема представляет собой цифровые часы – счетчики секунд, минут часов и календарь, доступные для программиста в виде последовательности байтов в ОЗУ размером 256 байтов.

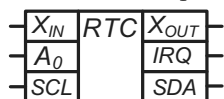


Рис. 13.21

Адреса и данные передаются в последовательном виде через интерфейс *I²C*, который имеет линию данных *SDA* и линию тактирования *SCL* [25]. Для задания «сетевого» номера микросхемы может использоваться вход *A0*, который позволяет различать одинаковые устройства при последовательном обмене.

Выходы X_{IN} , X_{OUT} являются входом и выходом внутреннего генератора и служат для подключения внешнего кварцевого резонатора (см. п. 6.8.3).

Выход *IRQ* используется в качестве запроса прерывания при обновлении счетчиков времени или при срабатывании будильника.

§13.4. Некоторые применения счетчиков

13.4.1. Принцип измерения длительности импульсов и частоты

Основой традиционной схемы измерения (рис. 13.22) является счетчик и логический элемент «И», на который подается две импульсных сигнала. Один из сигналов U_T назовем временным, поскольку для него определено время t_n , а другой U_F – частотным, он характеризуется частотой $F = 1/T_F$. Причем величина t_n должна быть много больше периода T_F . Временные диаграммы для этого случая приведены на рис. 13.23. Асинхронностью сигналов для простоты пренебрежем.

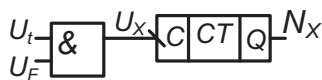


Рис. 13.22

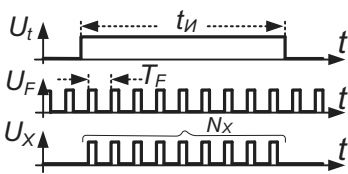


Рис. 13.23

Нетрудно заметить, что на выходе элемента «И» импульсы U_F как бы заполняют длительность $t_{\text{и}}$ (как деления мерной линейки), то есть

$$t_{\text{и}} = N_x T_F = N_x / F \quad (13.3)$$

Отсюда код, зафиксированный в счетчике

$$N_x = t_{\text{и}} \cdot F \quad (13.4)$$

окажется пропорциональным как длительности $t_{\text{и}}$, так и частоте F .

Если выбрать длительность $t_{\text{и}}$ равной 1 секунде, то код N_x , как следует из (13.4), будет численно равен частоте. Если наоборот взять частоту эталонной, например, с периодом $T_F = 1$ мкс, то получим код N_x , численно равный длительности $t_{\text{и}}$ в микросекундах.

Подобные измерения можно проводить также при помощи внутренних таймеров ОВМ (см. §10.7) или программно (см. п. 10.4.5).

13.4.2. Распределитель тактов

используют на практике для управления поочередно работающими устройствами, которые включаются и выключаются, строго друг за другом по порядку. Простейшим распределителем является двоичный счетчик и дешифратор (рис. 13.24).

Тактовые импульсы поступают на счетчик, и тот поочередно перебирает свои состояния, а дешифратор их расшифровывает (индицирует). Поэтому на выходе дешифратора появляются последовательные импульсы (рис. 13.25), которые могут использоваться для управления работой необходимых устройств.

В данной схеме нельзя использовать асинхронные счетчики, поскольку триггеры в них перебрасываются последовательно друг за другом (см. п. 13.1.1) и на выходе дешифратора могут появиться короткие ложные импульсы, не соответствующие очередности. Эти короткие импульсы, тем не менее, могут спровоцировать запуск схем на внеочередную работу, то есть в устройстве возникнет сбой. Поэтому в качестве счетчика рекомендуется использовать синхронные двоич-

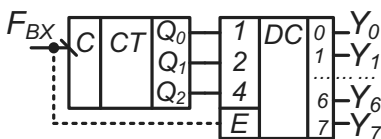


Рис. 13.24

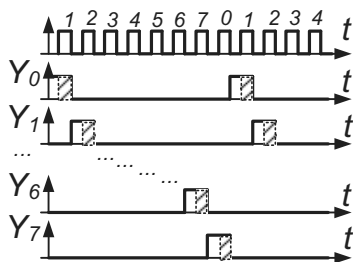


Рис. 13.25

ные счетчики, в которых триггеры перебрасываются одновременно (см. п. 13.1.3), и ложных состояний на выходе дешифратора нет.

В некоторых случаях дешифраторы стробируют (на рис. 13.24 показано пунктиром) тактирующим сигналом. То есть работа дешифратора разрешается только в момент действия единицы на тактовом входе. Когда единица исчезает, происходит переброс триггеров счетчика по отрицательному перепаду, то есть начинаются переходные процессы. Но дешифратор их уже не чувствует и не реагирует на них, он отключен нулевым сигналом стробирования.

При этом импульсы на выходе дешифратора укорачиваются. Если раньше по длительности они были равны периоду тактовых импульсов, то теперь они становятся равны половине, а точнее длительности единицы на входе (показано штриховкой на рис. 13.25).

Выходные импульсы при этом становятся не перекрываемыми во времени, они четко разнесены. Во многих случаях на практике это условие является важным, например один из первых процессоров Intel 8080 тактировался двумя не перекрывающимися сигналами.

13.4.3. Командный аппарат

или сокращенно называемый *командоаппаратом* – это устройство управления различными объектами, подающее команды «включить – выключить» с определенными, заранее заданными временными соотношениями, не обязательно последовательными.

Очередность подачи управляющих импульсов в командоаппарате может быть произвольной, поэтому при разработке, как правило, используют временную сетку, задающую цикличность и определенные моменты переключения объекта. В качестве устройства, генерирующего метки времени можно использовать ранее рассмотренный распределитель импульсов на основе счетчика и дешифратора (см. п. 13.4.2).

В качестве примера рассмотрим простейший командный аппарат с двумя выходными сигналами, временные соотношения которых заданы на рис. 13.26. Эти сигналы относительно друг друга произвольны, но привязаны к временной сетке, которую пусть для примера создадут 3-разрядные счетчик и дешифратор (рис. 13.27).

Поскольку сигналы временной сетки Y определяют только начало или окончание процесса, то для хранения управляющих сигналов будем использовать *RS*-триггеры (см. п. 12.1.2) в так называемом старте/

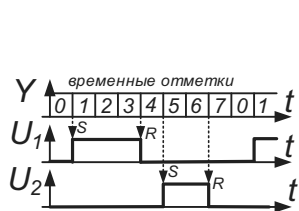


Рис. 13.26

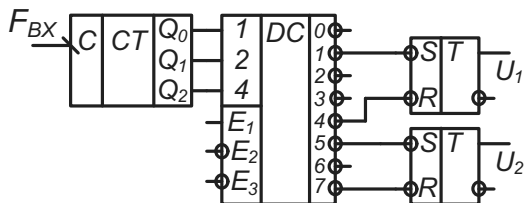


Рис. 13.27

стопном режиме. Сигнал «старт», определяющий момент включения устройства, будем подавать на вход S триггера, а «стоп» – на вход R . Схема на рис. 13.27 реализует заданную последовательность импульсов U_1 и U_2 .

Меняя подключение входов триггера в соответствии с техническим заданием, можно, как иногда говорят, «программировать» командоаппарат распайкой выводов. Но более удобным является настоящее программирование, когда в качестве элемента памяти используют не триггер, а программируемую микросхему ПЗУ (см. п. 14.2.2), которая, кстати, имеет встроенный дешифратор, что упрощает схему.

Для реализации командоаппаратов можно также использовать ОВМ с соответствующей программой (см. п. 10.2.2).

14

МИКРОСХЕМЫ ПАМЯТИ И ИХ ИСПОЛЬЗОВАНИЕ

§14.1. Микросхемы памяти

14.1.1. Общие сведения

Существует множество разновидностей запоминающих устройств, но традиционно различают два основных типа, ПЗУ и ОЗУ, которые в переводной литературе имеют аббревиатуру *ROM* (от *Read Only Memory*) и *RAM* – (*Random Access Memory*).

Граница между ПЗУ и ОЗУ в настоящее время достаточно размыта. Появились энергонезависимые ОЗУ (иногда с батареей питания), которые можно отнести к устройствам, постоянно хранящим информацию. В перспективе появятся ПЗУ, в которые можно записывать как будто в ОЗУ [30].

При использовании микросхем памяти в первую очередь обращают внимание на тип их интерфейса, параллельный или последовательный. Поскольку микросхемы с медленным последовательным интерфейсом (см. п. 14.1.6) специфичны и пригодны только для хранения редко изменяющихся цифровых данных, в дальнейшем будем в основном рассматривать чаще применяемые на практике микросхемы памяти с быстрым параллельным интерфейсом.

Такие микросхемы имеют три группы внешних выводов – адресные, управляющие и данные. Для примера на рис. 14.1 приведено УГО микросхемы ОЗУ типа КР537РУ10. Количество адресных выводов определяет емкость памяти, которая определяется подобно числу выводов для полного дешифратора (см. п. 11.1.1). Например, для данной микросхемы РУ10, имеющей 11 адресных выводов, общее количество ячеек равно $2^{11} = 2048$ кбайт.

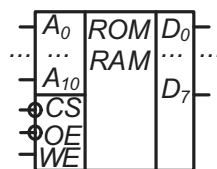


Рис. 14.1

Отметим, что адресные ($A_0 \dots A_{10}$) и управляющие выходы (CS , OE , WR) являются входами (однонаправлены), тогда как выходы данных – входами/выходами, то есть двунаправлены и имеют третье состояние, зависящее от управляющего сигнала OE (см. п. 2.3.7).

Управляющий сигнал CS , подключаемый обычно к выходу дешифратора банка памяти (см. п. 11.2.4), обеспечивает выбор микросхемы и разрешает работу всех внутренних узлов микросхемы (см. далее).

Сигнал WE (*Write Enable*), переводящий микросхему из режима чтения в режим записи, имеется только у ОЗУ или перепрограммируемых ПЗУ (ППЗУ). Если он активен, то информация может записываться в ОЗУ, иначе, данные только читаются.

Особенности подключения микросхем памяти к микропроцессорным шинам рассмотрены в §§14.5 и 14.6.

14.1.2. Внутренняя структура микросхем памяти

ОЗУ и ПЗУ имеют схожую структуру на основе полного дешифратора, обеспечивающего доступ к любой из ячеек ЗУ. Дешифратор реализуют обычно по матричной схеме, причем матрица может быть выполнена, как на элементах «И-НЕ» (*NAND*), так и на «ИЛИ-НЕ» (*NOR*) [30].

Кроме матричного дешифратора микросхема памяти имеет массив запоминающих ячеек, обозначенных на рис. 14.2 буквами «Я», и выходной каскад ВК. Условно одиночная ячейка памяти, подобно ключу, может находиться или в непроводящем состоянии, или в проводящем, формируя в выбранном состоянии соответствующий выходной сигнал. Для получения однобитного сигнала, например, на линии D_0 , требуется набор ячеек необходимой емкости.

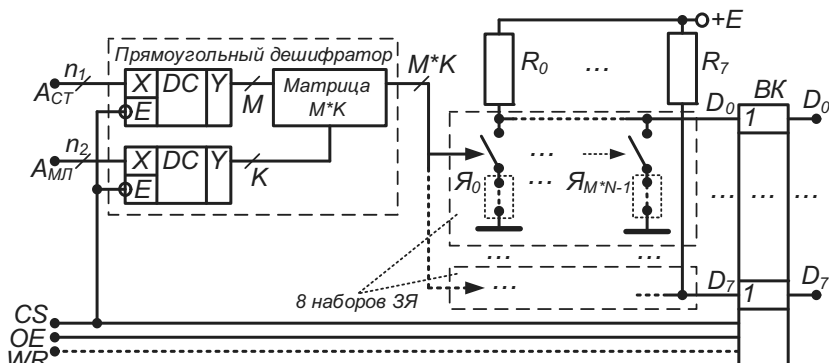


Рис. 14.2

В качестве коммутатора для выбора только одной ячейки используется неявный коммутатор на основе монтажного «ИЛИ» (см. п. 11.8.5). В некоторых случаях в качестве ключа коммутатора используется сама ячейка памяти, снабженная электродом выбора.

Вместо иллюстративного нагрузочного резистора R_o обычно используется токозадающий транзистор (см. п. 2.6.4).

Выходной каскад ВК в ОЗУ и ППЗУ является двунаправленным, обеспечивающим под управлением сигнала WE одно из нужных направлений передачи данных.

14.1.3. Разновидности запоминающих ячеек ПЗУ

Принципы действия ячеек памяти, которые могут хранить записанные данные достаточно долго, могут быть различными. Но вне зависимости от механизма хранения данных микросхемы ПЗУ принято подразделять на два основных типа, непрограммируемые (неизменяемые) и программируемые пользователем (ППЗУ или РПЗУ).

К первому типу относятся так называемые масочные ПЗУ, в которых программирование (металлизация перемычки или нет) осуществляется фотосаблоном (маской) [30] и выполняется на заводе производителем микросхем в ходе изготовления кристалла. Использование таких ПЗУ экономически целесообразно только при массовом производстве вычислительных устройств, программы которых изменяются очень редко.

Второй тип ПЗУ, программируемых пользователем, в свою очередь делится на однократно программируемые (английская аббревиатура *OTP*) и многократно. Одни из первых микросхем типа *OTP* использовали в качестве запоминающего элемента плавкие перемычки [30], которые прожигались при необходимости специальным программатором.

Но в настоящее время основным элементом ПЗУ являются различные варианты полевого транзистора с плавающим затвором [5, 30], который допускает многократное электрическое программирование. Причем в зависимости от структуры программирующего автомата внутри кристалла принято подразделять подобные микросхемы на три основных типа, обозначаемые аббревиатурами EPROM, FLASH, EEPROM, особенности которых, рассмотрены далее.

14.1.4. Типы электрически программируемых ПЗУ

Микросхемы типа *EPROM* (*Electrical Programmable ROM*), были исторически первыми массовыми изделиями с электрическим програм-

мированием. Стирание таких ППЗУ (удаление заряда с поверхности плавающего затвора) обычно производится за счет ультрафиолетового облучения, поэтому в отечественных справочниках для таких *перепрограммируемых* микросхем (например, КР573РФ5) иногда используется сокращение РПЗУ–УФ.

В двух других типах электрически программируемых ПЗУ (*FLASH* и *EEPROM*) стирание старых данных электрическое, что более удобно на практике. Причем стирание и соответственно программирование флэш-памяти осуществляется только блоками (секторами, как в магнитном диске). Это упрощает программирующий автомат и позволяет увеличить емкость микросхемы. Флэш-память широко используется, как в виде автономных устройств, например, всем известных «флэшек», так и в составе ОВМ для хранения программных кодов или сравнительно больших массивов информации.

Но для работы с одиночными изменяющимися данными флэш-память подходит плохо, потому что при перезаписи, например, одного байта, необходимо предварительно прочитать весь сектор, изменить один байт, а затем стереть и переписать сектор. Это не только усложняет обмен, но и приводит к уменьшению ресурса работы всего блока с данными (см. п. 14.1.5).

Внутренняя структура микросхем типа *EEPROM* (*Erased Electrical Programmable ROM* – электрически стираемое ППЗУ, то есть ЭСППЗУ) допускает перепрограммирование отдельных ячеек, не затрагивая остальные. Платой за это усовершенствование программирующего автомата является усложнение топологии микросхемы и соответственно меньшие плотность упаковки и количество ячеек на кристалле. Память типа *EEPROM* (отечественный термин ЭСППЗУ сравнительно редко используется в литературе) входит в состав ОВМ в качестве энергонезависимой памяти данных [31] или выполняется в виде отдельных микросхем, например, с последовательным интерфейсом (см. п. 14.1.6).

14.1.5. Ресурс работы программируемых ПЗУ

Основным параметром ППЗУ, характеризующим возможный ресурс работы, является количество допустимых циклов стирание-запись. Например, для внутренней флэш-памяти ОВМ, где хранятся коды команд, эта цифра зависит от типа микросхемы и имеет величину 100..1000. Для энергонезависимой памяти данных *EEPROM* – 100000 и более циклов перезаписи.

Дело в том, что процессы стирания и последующей записи приводят к постепенной деградации диэлектрика, окружающего плавающий затвор, и последующему отказу элемента памяти. Поэтому память *EEPROM*, обладающая большим ресурсом работы, имеет более толстую изоляцию и большие габариты, чем флэш-память. Однако увеличение толщины диэлектрика возможно только в определенных пределах. Поэтому разработчики микросхем могут применять выборочное стирание ячеек, не затрагивающее тех ячеек, где информация совпадает с вновь записываемой. Другим способом увеличения ресурса работы является введение дополнительных (избыточных) ячеек и применение самокорректирующихся кодов.

Во флэш-памяти для обеспечения сохранности данных может использоваться специальное устройство, которое следит за деградацией ячеек и при необходимости переносит данные в другое место. В этом случае количество перезаписей увеличивается, но в конце рабочего срока может уменьшиться производительность микросхемы, поскольку на выполнение указанной операции требуется определенное время.

Но все эти меры практически не помогают при высокой температуре окружающей среды, когда некоторые электроны, получившие повышенную тепловую энергию, могут самопроизвольно покидать плавающий затвор, приводя к искажению записанной информации. Это обстоятельство необходимо иметь в виду, если разрабатываемое устройство работает в широком температурном диапазоне или подвергается термотренировке для выявления некоторых видов отказов.

Не имеют деградации микросхемы для долговременного хранения данных с другими физическими принципами действия [30]. Например, память типа FRAM (Ferroelectric RAM) выполнена на основе сегнетоэлектрического (ферроэлектрического) конденсатора и МОП транзистора. В магниторезистивной памяти (MRAM – Magnetoresistive RAM) данные хранятся в магнитном туннельном переходе, образованном двумя слоями сегнетоэлектрика, между которыми заключен слой диэлектрика. Чтобы подчеркнуть значительно больший ресурс работы, подобные микросхемы относят к классу энергонезависимых ОЗУ.

Важно также знать при использовании ППЗУ, что процессы стирания и записи данных, связанные с удалением и накоплением заряда на плавающем затворе, инерционные и занимают определенное время, от единиц до десятков миллисекунд, которое зависит от типа микросхемы.

14.1.6. Интерфейсы микросхем ППЗУ

УГО микросхемы ППЗУ с параллельным интерфейсом практически не отличается от приведенного на рис. 14.1 и даже имеет подобно ОЗУ вход *WE*, разрешающий запись. Чтение данных из ППЗУ происходит точно также, как и из обычного ПЗУ (см. п. 14.5.2). Однако в режим программирования, который занимает определенное время и имеет свои особенности по сравнению с занесением данных в ОЗУ, микросхемы ППЗУ переключаются путем записи в них специальных команд определенного формата, которые в данном случае не рассматриваются.

Отметим только, что микросхемы ППЗУ с параллельным интерфейсом являются чаще всего являются по своему принципу действия флэш-памятью, то есть стираются и программируются целыми блоками. Подобные микросхемы широко используются, например, в компьютерах для хранения кодов *BIOS*, а также служат основой твердотельных накопителей, все чаще заменяющих жесткие диски. Микросхемы *EEPROM* с параллельным интерфейсом производятся значительно меньшими тиражами.

Учитывая, что процесс записи принципиально медленный, микросхемы ППЗУ иногда комплектуют последовательным интерфейсом, небольшое быстродействие которого в этом случае не критично. При этом резко уменьшается количество внешних выводов, что особенно ценится в малогабаритных автономных устройствах. Например, многие последовательные ППЗУ (обычно типа *EEPROM*, но встречается и флэш-память) емкостью до единиц мегабайт выпускаются в миниатюрном корпусе с трехпроводным интерфейсом типа *SPI*, двухпроводным – *I2C* и даже с однопроводным – *MicroLAN*.

В качестве примера последовательного ППЗУ рассмотрим УГО 8-ножечной микросхемы AT25512 (серия 25xx, емкость 512 кбит или 64 Кбайт), которая имеет трехпроводный интерфейс типа *SPI* (рис. 14.3). Как уже отмечалось (см. п. 12.8.1) подобный интерфейс имеют и мультимедиа карточки типа *MMC* или *SD*.

Обозначения последовательного интерфейса микросхемы укорочены по сравнению со стандартными аббревиатурами, но мнемонический смысл их сохранен (см. п. 12.8.1). Так вход *SI* соответствует *MOSI*, выход *SO* – *MISO*, *Clk* – *MCLK*.

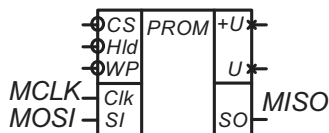


Рис. 14.3

Вход **CS** как всегда служит для выбора микросхемы, вход **WP** (*Write Protect*) обеспечивает, помимо кодовых команд, передаваемых по последовательному интерфейсу, внешнюю (аппаратную) защиту от случайной записи. Особую роль играет вход **Hold** (удержание), который предназначен для временной приостановки последовательного обмена. Активные уровни трех перечисленных входов нулевые.

Чтение или запись данных происходит под управлением специальных кодовых команд, которые здесь не рассматриваются.

14.1.7. Разновидности ячеек ОЗУ

Микросхемы ОЗУ традиционно делятся на статические **SRAM** (*Static RAM*) и динамические **DRAM** (*Dynamic RAM*).

В статических ОЗУ каждый элемент памяти представляет собой асинхронный **RS-триггер** (см. п. 12.1.2), хранящий один бит информации и реализованный на основе КМОП-технологии (см. п. 2.7.3). Подобные ОЗУ отличаются высоким быстродействием, но емкость микросхем из-за большого числа транзисторов в ячейке памяти обычно невелика.

Динамическая память более компактна, поскольку имеет простую запоминающую ячейку на основе «паразитной» емкости затвор-исток $C_{зи}$ одного полевого транзистора (рис. 14.4).

Эта емкость заряжается входным сигналом $D_{вх}$ через ключ записи **WR** до напряжения «0» или «1». После размыкания ключа заряд на конденсаторе некоторое время сохраняется, поддерживая транзистор запоминающей ячейки в проводящем или непроводящем состоянии и определяя тем самым значение $D_{вых}$.

Однако, емкость $C_{зи}$ очень маленькая (около 0.05 пФ) и может быстро (за несколько микросекунд) разрядиться токами утечек затвора и выбирающего ключа (см. рис. 14.2). Поэтому схема динамического ОЗУ требует очень частой (с периодом 1–2 мс) регенерации (восстановления) заряда.

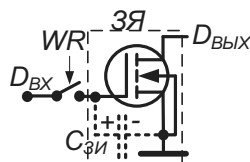


Рис. 14.4

Динамическая память значительно проще и дешевле по сравнению со статической и является в настоящее время наиболее массовой памятью компьютера. Однако ее быстродействие остается много лет примерно на одном уровне. Дело в том, что напряжение на емкости мгновенно измениться не может, поэтому быстродействие динамической памяти повышают в основном косвенными (структурными) методами, переходя, например, к страничному доступу или пакетной передаче данных [Угр].

§14.2. Некоторые применения ПЗУ

14.2.1. Реализация табличных функций

Как отмечалось ранее, заданные таблично логические функции можно реализовать, используя дешифратор (см. п. 11.2.3) или устройства на его основе, например, коммутатора (см. п. 11.8.7). Поэтому микросхему памяти, имеющую внутри полный дешифратор и неявный коммутатор (см. п. 14.1.1) можно применить для указанной цели, как это было рассмотрено на примере коммутатора (см. п. 11.8.7). Не останавливаясь на подробностях, которые при необходимости можно найти, например, в [16], перечислим только, что на основе ПЗУ можно реализовать сумматор, преобразователь 7-сегментного кода, неполный дешифратор для ВУ, знакогенератор и другие комбинационные устройства.

Следует отметить, что количество входных сигналов подобных устройств ограничено разрядностью адресных сигналов, а выходов – разрядностью данных.

14.2.2. Реализация командоаппарата

Идея использования ПЗУ в командоаппарате (см. п. 13.4.3) заключается в том, что состояние выходных сигналов в различные временные такты можно описать цифрами согласно порядку включения и выключения объектов (см. рис. 14.5), а эти получившиеся двоичные цифры можно занести в ПЗУ.

Для формирования временных тактов, как и прежде (см. п. 13.4.3), используется двоичный счетчик (рис. 14.6), перебирающий свои состояния под действием заданной тактовой частоты $F_{ВХ}$. Непоказанные на рисунке входы управления ПЗУ должны быть активными и заданы распайкой.

Таким образом, счетчик периодически адресует внутренние ячейки памяти, а при помощи внутреннего дешифратора из ячеек ПЗУ извлекаются

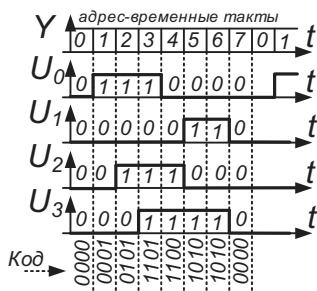


Рис. 14.5

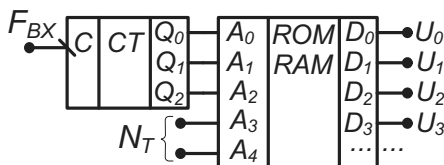


Рис. 14.6

записанные ранее двоичные коды. Эти коды легко программировать, то есть микросхема ПЗУ вместе со счетчиком аппаратно реализует заданный алгоритм и являются в определенной степени прообразом ПЛМ и ПЛИС (см. п. 14.3.2).

Отметим, что подобные командоаппараты реализуют только одну заранее заданную последовательность сигналов. Для оперативного изменения алгоритма работы в память можно занести несколько нужных кодовых таблиц, и при помощи старших адресов можно осуществлять их индексацию, управляя номером таблицы $N_{\text{--}}$.

Но более гибким способом изменения импульсной последовательности в зависимости от входных сигналов является микропрограммный автомат (см. далее), который как бы хранит разные таблицы и управляет их заменой на разных стадиях работы и в зависимости от входных переменных.

14.2.3. Микропрограммный автомат

включает в себя ПЗУ и регистр, тактируемый фронтом с частотой F_{BX} (рис. 14.7) [16]. Часть входных сигналов регистра (N_1) зависит от логических переменных X_{BX} , другая часть входов регистра (N_2) определяется кодами из ПЗУ. В свою очередь адресация ПЗУ осуществляется от части выходов регистра (N_3) и зависит, таким образом, как от входных сигналов, так и предыдущих кодов ПЗУ. Поэтому коды извлекаются из ПЗУ не последовательно, как в предыдущей схеме командоаппарата, а в «произвольном» порядке, который зависит, как от микропрограммы в ПЗУ, так и от условий, задаваемыми входными логическими переменными.

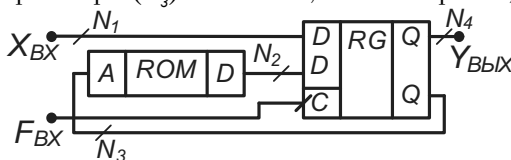


Рис. 14.7

Подобные микропрограммные автоматы могут использоваться для самых разнообразных целей, в том числе и внутри процессорных БИС для реализации машинных алгоритмов.

14.2.3. Цифро-аналоговый генератор

Сочетание двоичного счетчика и микросхемы памяти (рис. 14.8) может применяться не только для генерации импульсных последовательностей, но и для формирования периодических аналоговых

сигналов произвольной формы, так называемого цифрового синтеза сигналов [2]. Для

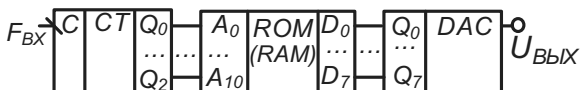


Рис. 14.8

этого цифровые значения, описывающие форму требуемого аналогового сигнала за период, заносятся в ПЗУ, а затем при помощи микросхемы ЦАП (см. п. 12.6.4) снова преобразуются в аналоговый вид.

В ходе работы счетчик перебирает свои состояния от нулевого до максимального, последовательно адресуя ячейки памяти, откуда извлекаются двоичные значения выходного сигнала в данный дискретный момент времени. Поэтому форма ступенчатого сигнала, появляющегося на выходе ЦАП, будет определяться таблицей в ПЗУ или ОЗУ, которая в свою очередь соответствует цифровым значениям аналоговой величины заданного сигнала на разных тактах работы.

Отметим, что в ряде случаев для хранения таблицы значений выходного сигнала можно использовать внутреннее ПЗУ ОВМ, а функции счетчика для извлечения кодов реализовать программно.

Для подавления высших гармоник («ступенек») выходного сигнала в простейшем случае может использоваться интегрирующая цепочка, как низкочастотный фильтр. Особенности разработки подобных генераторов можно найти в [16],

§14.3. Программируемые логические ИС

14.3.1. Общие сведения

Под ПЛИС понимают микросхемы универсального назначения, внутренняя структура которых позволяет пользователю при программировании устанавливать при помощи управляемых кодом переключателей необходимые связи между составными частями в соответствии с решаемой схемотехнической задачей. ПЛИС выпускаются достаточно большими тиражами и потому имеют меньшую себестоимость по сравнению с малотиражными узко специализированными БИС.

В настоящее время разработано множество разновидностей ПЛИС, описание которых выходит за рамки пособия. Поэтому ниже описаны лишь их общие особенности. Подробности при необходимости можно найти в соответствующей литературе, например, в [30] или [29].

На начальном этапе развития ПЛИС условно разделялись на наборы программируемых логических элементов и наборы программируемых логических устройств. Первые из них содержали базовые логические ячейки или вентили (*Gate*), которые могли соединяться при программировании в нужной комбинации. Вторые – готовые узлы (*Devices*) для реализации более сложных практических систем.

Наборы программируемых элементов в переводной литературе именуются сокращением PGA (Programmable Gate Array) или PLA (Programmable Logic Array) и производными от них – например, FPGA (Field PGA) или MPGA (Mask PGA). В отечественной литературе такие наборы называются базовыми матричными кристаллами (БМК) и программируемыми логическими матрицами (ПЛМ).

Наборы логических устройств обозначаются аббревиатурой PLD (Programmable Logic Devices).

Но с развитием элементной базы и усложнением технологии в ПЛИС стали включать, как наборы логических элементов, так и наборы узлов. Поэтому для классификации стали чаще использовать признак сложности, выделяя простые ПЛИС (*SPLD – Simple PLD*) и сложные (*CPLD – Complex PLD*).

Причем внутренняя структура подобных ПЛИС реализуется в большинстве случаев по матричному принципу из однородных укрупненных элементов (макроячеек) или логических узлов (см. п. 14.3.3). И потому термин ПЛМ, которым в свое время обозначались матрицы логических элементов «И-ИЛИ», зачастую используется в широком смысле, относя к ПЛМ матрицы макроячеек или крупных логических блоков.

14.3.2. Принцип действия программируемого комбинационного устройства

основан на том, что всякую логическую функцию можно представить в виде суммы сомножителей входных логических переменных и их инверсных значений [24].

Например, ранее для базовых комбинационных устройств (дешифратора, сумматора и др.) были определены выходные сигналы в виде суммы логических сомножителей (см. п.п. 11.1.3 и 11.9.3).

Поэтому для реализации программируемого комбинационного устройства нужны два набора логических элементов – для логическо-

го умножения и сложения произведений. Причем, число элементов должно быть выбрано с запасом.

Для получения необходимых сомножителей нужно учитывать все возможные комбинации входных переменных, то есть количество элементов умножения «И», как и в полном дешифраторе (см. п. 11.1.1), определяется числом 2^N , где N – количество входных сигналов. В результате образуется набор (матрица) логических элементов «И», каждый из которых имеет N входов.

С другой стороны количество элементов «ИЛИ» зависит от числа выходных сигналов и внешних ножек микросхемы.

Третьей составной частью ПЛИС должны быть аппаратные средства, позволяющие выбирать необходимые сомножители для суммирования. В первых ПЛИС это производилось изменением фотошаблона, потом прожиганием ненужных перемычек, а в настоящее время используются электрически программируемые ячейки на основе полевого транзистора с плавающим затвором [30].

Иллюстративная структура универсального программируемого устройства с двумя входами и двумя выходами приведена на рис. 14.9. Она имеет четыре 2-входовых элемента «И» для вычисления сомножителей входных сигналов и их инверсий. Для формирования выходных сигналов F_1 и F_2 на основе логических сумм используются два элемента «ИЛИ».

Отметим, что плавкие перемычки, показанные на рисунке для наглядности, уже давно не используются, и коммутация цепей происходит с помощью управляемых кодом пользователя логических ключей (см. п. 11.8.2), показанных на рисунке пунктиром.

При этом нетрудно заметить, что элементы «И» образуют, по сути, дешифратор, который управляет образовавшимся выходным коммутатором на основе элемента «ИЛИ» и логических

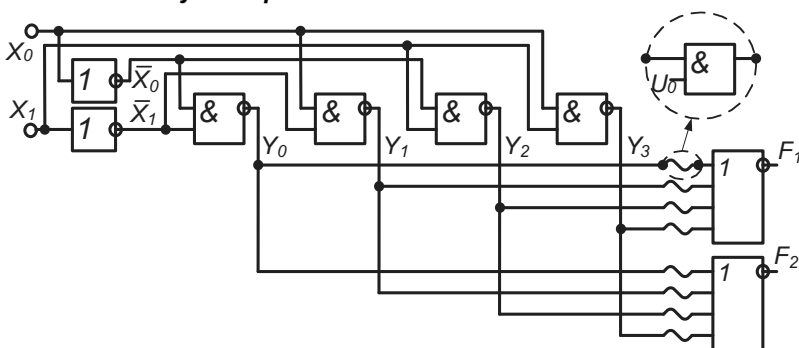


Рис. 14.9

ключей. А управляющие коды задают таблицу выходных значений функции (см. п. 11.8.7).

Управляющие коды $U_0 \dots U_7$ для логических ключей задаются пользователем при программировании и формируются в запоминающих ячейках ПЗУ. Если какой-то из сомножителей в устройстве не используется, то его управляющий сигнал должен быть нулевым.

Таким образом, в основе работы первых ПЛИС лежали принципы табличной реализации логических функций на основе полного дешифратора (см. п. 11.1.1), распознающего все возможные комбинации входных сигналов, и программируемого коммутатора (см. п. 11.8.3), выделяющего необходимый логический результат (см. п. 11.8.7). Комбинацию такого дешифратора, выполненного по матричной схеме (см. п. 11.1.7) и неявного коммутатора (на основе матрицы «ИЛИ») было принято называть ПЛМ.

Получившаяся структура ПЛМ оказывается похожей на структуру ПЗУ (см. п. 14.1.2), поскольку тоже имеет наборы (матрицы) логических элементов «И», «ИЛИ». Но в отличие от ПЗУ, в котором используются все комбинации полного дешифратора для доступа к данным, в ПЛМ многие комбинации дешифратора могут быть отключены при программировании.

Число используемых комбинаций ПЛМ, называемых «термами», является одним из важных параметров ПЛМ. Другим параметром служит число выходов матрицы «ИЛИ». Для выбора прямого или инверсного значения выходного сигнала в ПЛМ обычно используется управляемый элемент «исключающего ИЛИ» (см. п. 11.9.2).

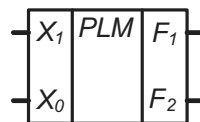


Рис. 14.10

УГО иллюстративной ПЛМ приведено на рис. 14.10.

14.3.3. Составные части ПЛИС

Основой ПЛИС (рис. 14.11) является коммутационная матрица, логические блоки (ЛБ), входной (ВХК) и выходной (ВЫХК) каскады.

Коммутационная матрица при програм-

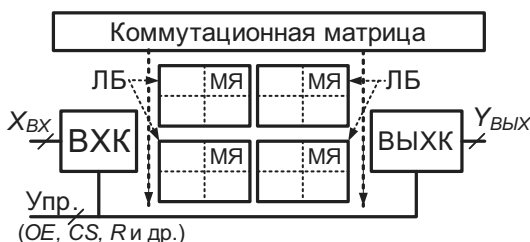


Рис. 14.11

мировании устанавливает внутренние связи между частями ПЛИС, определяя взаимодействие логических блоков между собой и внешним миром.

Логические блоки в свою очередь состоят из наборов макроячеек (МЯ). Каждая из макроячеек является миниатюрным программируемым устройством, выполненным на основе локальной ПЛМ (дешифратора и коммутатора) для реализации комбинационных функций и триггерно-регистровых устройств, зависящих от типа микросхемы. Поэтому на основе макроячеек можно создавать не только комбинационные устройства, но и устройства с памятью или микропрограммные автоматы (см. п. 14.2.3), вводя при помощи управляемого коммутатора ПЛМ необходимые обратные связи. Более сложные ПЛИС могут иметь в своем составе и другие цифровые узлы, такие как компараторы, массивы ОЗУ и ПЗУ.

Входной и выходной каскады формируют необходимые характеристики, определяют нагрузочную способность, тип выхода и другие заданные внешние параметры. Отметим, что эти каскады могут быть двунаправленными, то есть работать в качестве входного или выходного вывода в зависимости от управляющего сигнала.

14.3.4. Конфигурирование ПЛИС

Важной не упомянутой ранее частью ПЛИС является встроенное устройство для занесения конфигурационного кода, который генерируют на заключительном этапе системы проектирования [29]. Благодаря этому устройству большинство современных ПЛИС допускают внутрисистемное программирование *ISP* (см. п. 7.3.3) без внешних программаторов через последовательный интерфейс с малым числом проводов.

Следует только иметь в виду, что для хранения кода, определяющего конфигурацию коммутационной матрицы и локальных ПЛМ, могут использоваться ячейки *EEPROM* (см. п. 14.1.4) или более скоростные *SRAM* (см. п. 14.1.7).

Поэтому в первом случае ПЛИС программируется только однажды, поскольку при выключении питания ранее записанная конфигурация сохраняется в энергонезависимом *EEPROM*.

Во втором случае, когда код хранится в ОЗУ, необходимо каждый раз после включения питания заново загружать в микросхему программу конфигурации. Эта загрузка может происходить из внешнего ППЗУ с последовательным интерфейсом (см. п. 14.1.6) автомати-

чески или под управлением внешнего процессора. С одной стороны такая загрузка увеличивает время подготовки к работе, но с другой стороны обеспечивает возможность оперативного и многократного перепрограммирования ПЛИС.

14.3.5. Программируемые аналоговые ИС

Достоинства ПЛИС были оценены и производителями микросхем для аналоговой обработки данных, которые стали выпускать программируемые наборы типовых аналоговых устройств, таких как усилитель с программируемым коэффициентом (см. п. 6.2.3), сумматор (см. п. 6.1.1), интегратор (см. п. 6.1.2), аналоговый компаратор (см. п. 17.1.3), ЦАП (см. п. 16.1.1) и др. Подобные узлы, обычно сгруппированные в макроячейки, можно соединять при помощи управляемых кодом *аналоговых* переключателей для реализации своих устройств.

Главным критерием при выборе аналоговых программируемых ИС наряду с функциональной полнотой является обеспечение необходимой точности и температурной стабильности. Поэтому в некоторых микросхемах может иметься режим автокалибровки, минимизирующий погрешность системы.

Помимо логических или аналоговых программируемых структур в настоящее время разработаны комбинированные микросхемы, имеющие на одном кристалле наряду с цифровыми и аналоговыми цепями и процессорное ядро с ОЗУ и ПЗУ. Подобные микросхемы позволяют в ряде случаев реализовать на кристалле почти все части микропроцессорной системы.

14.3.6. Система на кристалле

Дополняя цифровые и аналоговые программируемые ИС процессорным ядром с памятью, можно обеспечить более высокую степень твердотельной интеграции и совместить два основных способа реализации универсального схемотехнического устройства, аппаратный и программный. Такие системы на кристалле (СнК), обозначаемые в переводной литературе сокращением *SoC (System on Chip)*, позволяют оптимизировать структуру системы и, используя достоинства каждого из указанных способов, достигать максимально возможных результатов [30].

Отметим, что в качестве процессорного ядра в СнК достаточно часто используется ядро семейства x51 (см. главу 7), поэтому

му опыт работы с такой ОВМ может пригодиться и в дальнейшем.

Для пользователя существует два варианта использования СнК. В первом подбирается уже готовая микросхема со структурой, наиболее близкой к требуемой, и в соответствии с рекомендациями производителя осуществляется этап проектирования программной части для процессорного ядра и конфигурации для имеющихся аналогового блока и ПЛИС.

Во втором варианте, когда большой тираж заказной микросхемы может окупить издержки, пользователь (или по его заказу разработчик микросхем) на этапе проектирования комбинирует узлы уже не внутри готового кристалла, а на модели, которая воплощается затем в твердом теле.

Дело в том, что опыт, накопленный разработчиками твердотельной электроники, позволил тиражировать наряду с микросхемами и топологию отдельных узлов в виде программных (*soft*) модулей или «виртуальных» узлов, называемых иногда *IP*-блоками (от *Intellectual Property* – интеллектуальной собственности). Причем благодаря стандартизации в проекте можно использовать «виртуальные» узлы из библиотек других разработчиков [30].

Наряду с изменяемыми по выбору пользователя блоками, в СнК имеется большая неизменяемая часть в виде процессорного ядра, массивов памяти и блоков встроенных ПЛИС. Чтобы подчеркнуть неизменяемость этих блоков, их иногда называют *hard*-модулями [30].

Микросхемы СнК являются весьма перспективными изделиями, поскольку позволяют не только оперативно (в ходе эксплуатации) улучшать свойства системы, но и существенно ускорять отладку и тестирование на этапе разработки.

§14.4. Обмен данными между процессором, ЗУ и ВУ

14.4.1. Средства для обмена

Процессор передает и принимает информацию от ЗУ и внешних устройств (ВУ) при помощи системной или локальной магистрали, которая представляет собой комбинацию трех шин, адреса, данных и управления.

Шина адреса (ША) служит для установки двоичного кода, посредством которого осуществляется выбор (адресация) одного из внешних источников информации. Эта процессорная шина обычно выполнена на однонаправленных элементах с третьим состоянием, позволяющим процессору при необходимости отключаться от ША. Дело в том, что в некоторых случаях адрес может устанавливаться контроллером прямого доступа к памяти (см. п. 11.2.5) или сопроцессором.

Шина данных (ШД) – двунаправленная, поскольку служит, как для приема, так и передачи данных.

Шина управления (ШУ) обеспечивает общую синхронизацию процессов, указывая момент и тип обмена, например, направление движения информации по ШД или селекцию отдельных видов ВУ.

В некоторых случаях шины адреса и данных могут быть совмещены за счет поочередного использования (мультиплексирования) общих выводов адреса/данных (см. п. 7.1.6). В этом случае адрес и данные сопровождаются своими синхросигналами, обозначающими тип кода.

Размерности шин адреса, данных и набор управляющих сигналов зависят от процессорного стандарта и назначения системной или локальной магистрали. В качестве примера в §14.6 рассмотрены шины для подключения внешней памяти к стандартной ОВМ семейства x51.

14.4.2. Основные механизмы обмена данными

Различают три основных типа обмена данными, программный обмен, обмен с прерыванием основной программы и обмен с прямым доступом ВУ к памяти.

Работа процессора происходит под управлением программы, поэтому системная магистраль работает большей частью в режиме *программного* (предсказуемого) обмена информацией под управлением процессора. Например, извлечение очередного кода программы, чтение операнда из памяти, запись результата, обслуживание внешних устройств и др.

Известны две разновидности программного обмена, асинхронный и синхронный. При асинхронном обмене (см. п. 14.4.4) темп работы зависит от реакции памяти или ВУ. При синхронном обмене (см. п. 14.4.3) темп работы жестко задает процессор, формируя в нужный момент синхросигнал.

Однако внешние события чаще всего бывают неожиданными для программы, и процессор может с опозданием реагировать на них, если он использует только программный обмен.

Для срочного привлечения внимания к произошедшему событию используется *механизм прерывания* работы процессора по программе и переход к специально оговоренной процедуре обслуживания ВУ (см. п. 8.4.1). После завершения обслуживания процессор возвращается к выполнению основной программы.

Отметим, что при выполнении процедуры прерывания по обслуживанию внешнего объекта используется в основном программный обмен.

При обслуживании некоторых устройств (накопителей, сетевых адаптеров и пр.) процессор должен, в цикле принимать и передавать большие массивы данных. Использование программного обмена для этой цели приводит к большим потерям времени, поскольку процессор должен вычислять адрес в памяти, проверять окончание цикла, выполнять опрос ВУ и обслуживание памяти.

Альтернативой программному обмену в подобных случаях является механизм *прямого доступа к памяти* (ПДП). Для этого используется специальное устройство – контроллер ПДП в виде отдельной БИС или части процессора. Этот контроллер выполняет пересылку массива из ВУ в память или обратно без участия процессора, который в режиме ПДП отключается от своих шин (см. п. 11.2.5). Контроллер, который должен быть заранее запрограммирован, захватив шины, самостоятельно вычисляет адрес, формирует *пару* управляющих сигналов (например, читать ВУ – писать в ОЗУ), проверяет предельные границы массива. Причем эти операции выполняются очень быстро (не программно) аппаратурой контроллера – адрес последовательно перебирается счетчиком, контроль массива также осуществляет вычитающий счетчик или цифровой компаратор.

Поэтому режим ПДП всегда используют высокоскоростные внешние устройства, сопроцессоры и мультипроцессорные вычислительные комплексы.

14.4.3. Синхронный программный обмен

Темп работы при синхронном обмене задается процессором при помощи управляющих сигналов.

Например, в микросхемах фирмы Моторола (см., например, описание цифровых часов в п. 13.3.6) один сигнал R/W (Read/Write) определяет направление обмена, а второй – DS (Data Strob) – момент обмена.

Но чаще используется стандарт фирмы *Интел*, в котором пара управляющих сигналов *RD* и *WR* определяют и направление, и момент

обмена. Причем для памяти (*Memory*) эти сигналы обозначаются как *MRD* и *MWR*, а для ВУ (*Input/Output*) – *IORD* и *IOWR*.

Отметим также, что все эти сигналы инверсные, то есть имеют нулевой активный уровень.

Для пояснения процесса записи данных из процессора в память или ВУ воспользуемся временными диаграммами, приведенными на рис. 14.12, где под сигналом *WR* будем понимать обобщенный сигнал записи (или *MWR*, или *IOWR*).

Вначале процессор устанавливает адресный код на ША, чтобы память или ВУ успели расшифровать его и подготовиться к приему данных. Затем процессор устанавливает информацию на ШД и спустя некоторое время формирует инверсный синхросигнал записи *WR*, как бы гарантируя на участке «А–Б» неизменность данных. Таким образом, при записи данных все сигналы на трех шинах формируются процессором.

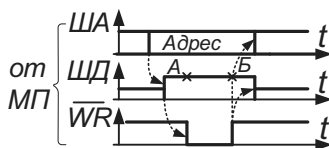


Рис. 14.12

Длительность синхроимпульса WR должна быть рассчитана на самое медленное устройство в системе. Поэтому при обслуживании быстрых устройств некоторое время процессор простаивает, то есть его возможности используются не полностью.

При чтении данных (рис. 14.13) процессор по-прежнему устанавливает адрес и спустя некоторое время синхросигнал чтения *RD*. А информацию на ШД устанавливает адресуемая память или ВУ, причем реагирование их привязано к началу синхроимпульса. Поэтому сигналы на ШД можно считать установившимися только к моменту окончания синхроимпульса (точка «А»), поскольку точные временные характеристики источников данных не известны.

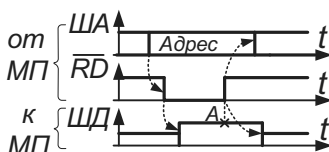


Рис. 14.13

Отсюда следует, что, как и при записи, длительность синхроимпульса должна выбираться на худший случай. В простых системах с этим мирятся, а в быстродействующих имеется возможность программной подстройки длительности синхроимпульсов. В некоторых случаях медленные устройства могут даже приостановить работу процессора, чтобы успеть произвести обмен, как это делается при асинхронном обмене.

14.4.4. Асинхронный программный обмен

В том случае, если микропроцессорная система обслуживает устройства с заведомо различным временным запаздыванием, синхронный обмен, ориентированный на самое медленное устройство, становится невыгодным. В этом случае применяют иной принцип, когда процессор инициирует начало обмена, а окончание последнего зависит от внешнего подтверждения о том, что процесс записи или чтения завершен. Момент поступления этого подтверждения *Ack* (от *Acknowledgment*) процессору неизвестен (нерегулярен) и поэтому подобный обмен называют асинхронным.

Временные диаграммы для асинхронного обмена приведены на рис. 14.14 и 14.15. Общей их особенностью является то, что процессор, как и при синхронном обмене, устанавливает код на ША (и данные на ШД, если нужно), а затем устанавливает синхросигнал *WR* или *RD*. Адресуемое внешнее устройство реагирует на это с определенной задержкой и, выполнив операцию обмена, сигнализирует об окончании установкой инверсного подтверждения *Ack*. Процессор в ответ убирает синхросигнал и переходит к следующему циклу обмена.

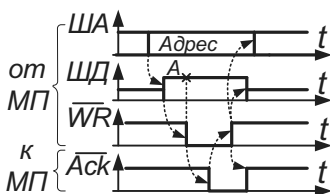


Рис. 14.14

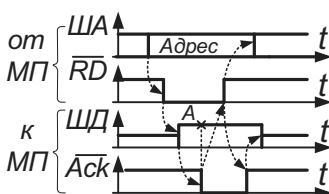


Рис. 14.15

Примечание. Сигнал *Ack* обычно формируется по схеме «мониторинга ИЛИ» от многих ВУ.

Основным недостатком асинхронного способа является то, что отсутствующее или неисправное устройство может оставить процессор без подтверждения и система может зависнуть. Для устранения этой нежелательной ситуации обычно ограничивают максимальное время ожидания ответа процессором, например, 10 микросекундами.

В заключение отметим, что большая роль при обмене, как следует из логики работы и временных диаграмм, отводится шине адреса. На запросы процессора откликаются только ячейки и устройства с указанным номером. Поэтому важной частью ВУ (см. п. 11.2.5) и микросхем памяти (см. п. 14.1.2) являются внутренние дешифраторы двоичного кода.

Работа и временные диаграммы контроллеров прерывания и ПДП выходят за рамки пособия и здесь не рассматриваются.

§14.5. Подключение регистров и памяти к процессору

14.5.1. Чтение данных из регистра

Простейшим ВУ (источником или приемником данных) является регистр памяти. Пусть, например, в регистр (рис. 14.16) заносятся под управлением сигнала записи C некие данные X_{BY} .

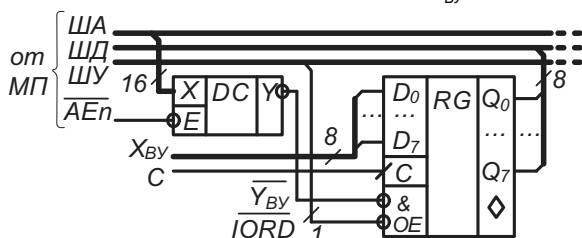


Рис. 14.16

При подключении выходов регистра к общей ШД может возникнуть конфликт данных (см. п. 2.3.5) из-за того, что источников информации много. Чтобы не было конфликтов, подключаться к шине данных можно при помощи неявного коммутатора (см. п. 11.8.6), только получив разрешение в виде уникального номера регистра на ША. Поэтому важной частью ВУ, обеспечивающей подключение выходов регистра к ШД является дешифратор адреса.

Отметим, что дешифратором (при необходимости) должен управлять сигнал AEN, который является индикатором того, что адрес установлен процессором, а не контроллером ПДП (см. п. 11.2.5).

Как видно из рис. 14.16 выход дешифратора Y_{BY} , реагирующий на присвоенный данному ВУ адрес, подается только на один из управляющих входов регистра для разрешения выдачи данных OE (см. п. 12.4.4). На второй разрешающий вход OE подается инверсный синхросигнал чтения $IORD$. Учитывая, что оба входа объединяются на внутренней схеме «И», выдача данных на ШД будет осуществляться при наличии одновременно двух активных (нулевых) разрешающих сигналов, Y_{BY} и $IORD$.

Если процессор будет опрашивать ВУ с другими адресами, формируя синхросигнал $IORD$, то дешифратор не будет реагировать на эти коды, то есть сигнал $Y_{в\gamma}$ будет пассивным, удерживая выходы регистра в третьем (неподключенном к ШД) состоянии.

Если процессор будет обращаться к памяти и случайно установит адрес, совпадающий с адресом ВУ, то дешифратор среагирует и сформирует активный сигнал $Y_{в\gamma}$. Но чтение памяти происходит под управлением своего управляющего сигнала MRD , поэтому в этом случае сигнал $IORD$ будет пассивным, запрещая подключение регистра к ШД.

Одновременно два разрешающих сигнала $Y_{в\gamma}$ и $IORD$ будут только в том случае, когда процессор, выполняя команду чтения из заданного ВУ, установит адрес ВУ и сформирует синхросигнал чтения (рис. 14.13). Тогда будет разрешена выдача данных из регистра на ШД по обоим входам OE и процессор прочитает информацию из регистра.

14.5.2. Чтение данных из ПЗУ

подобно чтению из регистра, только внутри ПЗУ находится множество ячеек (регистров), адресуемых своим встроенным полным дешифратором (см. п. 14.1.2). Поэтому внешний дешифратор (рис. 14.17) на основе старших битов адреса $A_{ст}$ реагирует только на зону, банк памяти (см. п. 11.2.4), формируя общее разрешение на входе выбора микросхемы CS .

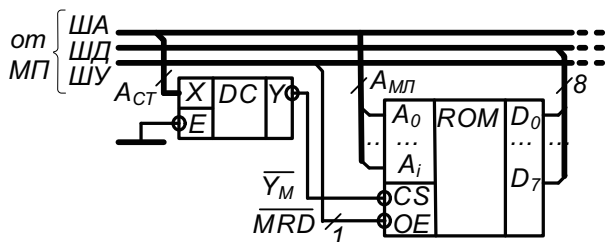


Рис. 14.17

Другим управляющим сигналом, как и ранее, является синхроимпульс чтения, но только из памяти MRD , который по входу OE разрешает выдачу данных из ПЗУ при наличии разрешения по входу CS от дешифратора. А это будет возможно, если процессор обратится к данной микросхеме, установив нужный адрес банка памяти и синхроимпульс чтения.

В остальных случаях, например, при чтении из другого банка, дешифратор не разрешит работу микросхемы. Если же процессор,

обслуживая ВУ, случайно установит адрес, совпадающий с адресом ПЗУ, то сработает дешифратор, но синхросигнал MRD, обслуживающего память не будет.

Нужно также отметить, что в дешифраторе банка памяти сигнал AEN не используется, поскольку память реагирует на адреса, устанавливаемые, как процессором, так и контроллером прерываний.

14.5.3. Запись данных во внешний регистр

В режиме записи данных из процессора конфликтов на ШД нет, так как все регистры подключены к шине данных своими не мешающими друг другу входами (рис. 14.18). Но сохранить данные может только один регистр, получивший «приказы» одновременно от дешифратора и от сигнала управления IOWR.

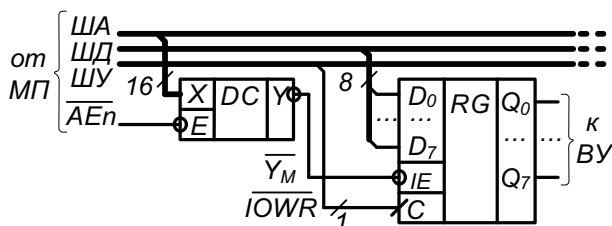


Рис. 14.18

Обычно сигнал от дешифратора $Y_{ВУ}$ поступает на вход разрешения приема данных IE (см. п. 12.4.3), а синхросигнал - на вход записи С. Как во всех ВУ, дешифратор при необходимости должен учитывать сигнал-индикатор AEN.

14.5.4. Подключение ОЗУ для чтения и записи данных

в принципе похоже на подключение ПЗУ (рис. 14.19). Дешифратор зоны памяти, анализирующий старшие разряды ША, управляет входом выбора микросхемы CS, а синхросигнал чтения MRD – входом выдачи данных OE. Отличие заключается в том, что используются дополнительный синхросигнал записи MWR, который меняет в выходном каскаде ОЗУ (см. п. 14.1.2) направление движения данных.

Когда процессор производит чтение из этой микросхемы, формируется нужный адресный сигнал, дешифратор устанавливает сигнал Y_m для выбора микросхемы и в момент действия синхросигнала MRD происходит съем кода с шины данных.

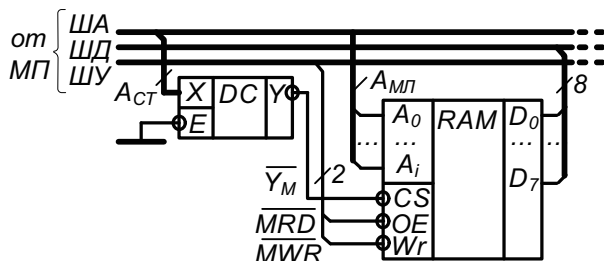


Рис. 14.19

При записи данных последовательность действий такая же, но формируется сигнал *MWR*, который сопровождает выводимые данные (см. рис. 14.12).

На «чужие» адреса не реагирует дешифратор. Если же случайно адрес ВУ совпадает с адресами ОЗУ, то при этом генерируются синхросигналы IORD и IOWR, которые игнорируются ОЗУ.

§14.6. Подключение внешней памяти к ОВМ

Часть семейств ОВМ, ориентированных на достаточно сложные практические задачи, имеют кроме внутренней памяти средства для подключения внешних ПЗУ, ОЗУ, интерфейсных модулей и других внешних устройств. Для этого внутренняя процессорная магистраль выводится наружу в виде трех шин – адреса, данных и управления. Особенности такой внешней магистрали рассмотрим на примере ОВМ х51.

14.6.1. Шины ОВМ для подключения внешней памяти

организованы на основе внешних выводов портов **P0** и **P2** (см. п. 7.1.6), которых все равно не хватало для организации 16-битной ША и 8-битной ШД. Поэтому разработчикам пришлось использовать мультиплексный принцип передачи адреса/данных. Для этого порт **P0** задействован для поочередной передачи младшего байта адреса и байта данных, а порт **P2** – для формирования старшего байта адреса.

Чтобы сохранить младший байт адреса, который на короткое время появляется на выводах порта **P0** и сопровождается адресным синхросигналом *ALE* (см. п. 7.1.6), необходимо использовать внешний регистр (например, КР1533ИР22), как показано на рис. 14.20.

После адресной передачи порт **P0** выполняет функции ШД.

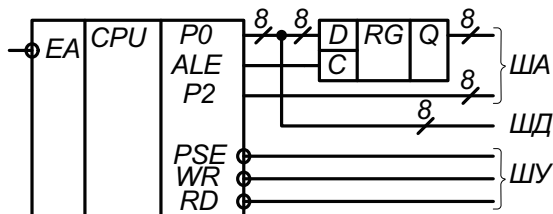


Рис. 14.20

К управляющим сигналам ШУ относятся *PSE* (чтение из ПЗУ), *RD* (чтение из ОЗУ), *WR* (запись в ОЗУ), которые тактируют внешнюю память (см. п. 7.1.7).

Следует иметь в виду, что внешние шины будут работать только в двух случаях. Во-первых, когда будет активен сигнал **EA** (см. п. 7.1.6), заставляющий процессор извлекать коды программы из внешнего ПЗУ или выполняется команда перехода на адрес, больший размера внутреннего ПЗУ. Во-вторых, когда будет выполняться одна из команд пересылки с участием внешнего ОЗУ (см. п. 7.8.3).

Отметим также, что использование внешней памяти существенно уменьшает количество внешних выводов для обслуживания иных внешних объектов, поскольку в распоряжении пользователя остаются только порт **P1** и часть выводов **P3**. Чтобы несколько уменьшить этот недостаток разработчики x51 ввели упрощенный режим 8-битной адресации (см. §14.7), когда старший байт адреса не используется, что позволило освободить порт **P2** для пользователей.

14.6.2. Подключение внешнего ПЗУ к ОВМ

Осуществляется по стандартной схеме (см. п. 14.5.2) с использованием шин адреса, данных и синхросигнала *PSE* (рис. 14.21), который тактирует чтение кода команды из внешнего ПЗУ.

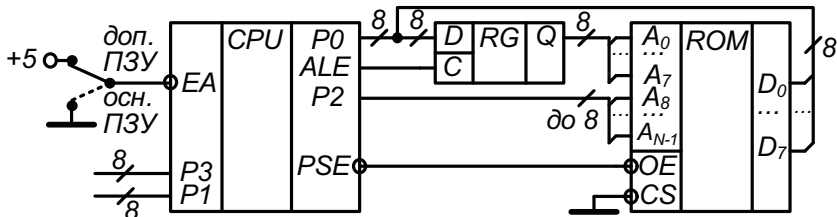


Рис. 14.21

Временные диаграммы чтения из внешнего ПЗУ (рис. 14.22) также в основном подобны ранее рассмотренным (см. п. 14.5.2), то есть сначала устанавливается адрес (только младший байт сопровождается синхроимпульсом *ALE* и сохраняется по отрицательному перепаду во внешнем регистре), затем формируется синхроимпульс чтения *PSE*.

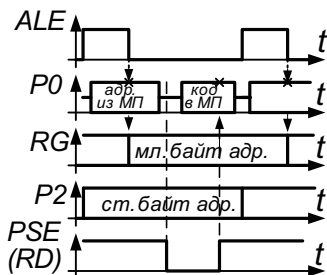


Рис. 14.22

В зависимости от состояния сигнала EA в работе внешних шин можно выделить два варианта. Если сигнал EA активен (равен нулю), то внешнее ПЗУ используется в качестве основного, то есть внешние шины работают все время, обеспечивая извлечение программных кодов команд.

Если сигнал EA пассивен, то ОВМ работает с внутренней памятью команд, внешние шины не будут активированы (за исключением сигнала ALE) и состояния портов P0 и P2 зависят от программы. Но, если память команд в процессе работы по программе будет адресована за пределами внутреннего ПЗУ, то порты P0 и P2 станут выполнять свои шинные функции. При этом внешнее ПЗУ будет дополнять внутреннее и находиться в верхней части адресного пространства.

Если ПЗУ выполнено из нескольких микросхем, то для их селекции можно использовать дешифратор банка памяти (см. п. 11.2.4). Но в этом случае, чтобы не увеличивать количества внешних схем, рекомендуется выбрать ПЗУ большей емкости.

14.6.3. Подключение внешнего ОЗУ к ОВМ

также осуществляется по общим правилам (см. рис. 14.23 и п. 14.5.4). Нужно только иметь в виду, что два тактирующих сигнала *WR* и *RD*

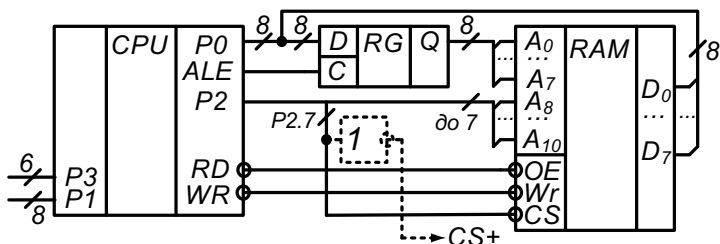


Рис. 14.23

используют выходы *P3.6* и *P3.7* порта *P3*, то есть в схеме и программе пользователя *указанные выходы использовать нельзя*.

*Количество микросхем ОЗУ можно увеличить, используя дополнительный внешний дешифратор банка памяти (см. п. 11.2.4). Но в простейшем случае, например, для двух микросхем, для селектирования можно использовать старший бит адреса (*P2.7*) и инвертор, который показан на рисунке пунктиром, для выбора дополнительного ОЗУ.*

Временные диаграммы при обмене с внешним ОЗУ почти отличаются от рассмотренных в п. 14.4.3, за исключением формирования младшего байта адреса. На рис. 14.24 показаны временные диаграммы при записи во внешнее ОЗУ, а процесс чтения из ОЗУ подобен чтению из ПЗУ (рис. 14.22).

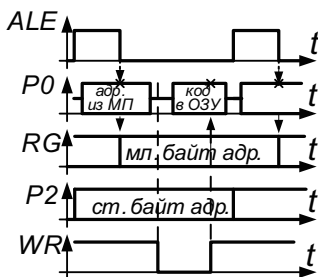


Рис. 14.24

Вместе с ОЗУ к шинам ОВМ можно подключить и внешнее ПЗУ, конкурировать они не будут, так как тактируются различными синхросигналами.

14.6.4. Программное использование внешнего ОЗУ

На языке Си обязательно нужно объявить массив или отдельные переменные во внешней памяти данных, например

xdata at 20 unsigned char mXdataVar;

Ключевое слово «at 20» позволяет указать место переменной в конкретной ячейке памяти.

Затем в программе можно читать

mY= mXdataVar;

или сохранять эту переменную

mXdataVar = mX;

В программах на ассемблере специального объявления не требуется, поскольку для внешнего ОЗУ используются иные, чем для внутренней памяти, команды пересылки. Например, для режима 16-битной адресации (см. п. 7.8.3)

MovX A, @DPTR
MovX @DPTR, A

В регистр-указатель данных **DPTR** предварительно нужно занести адрес нужной переменной. Режим 8-битной адресации рассмотрен ниже (см. п. 14.7.1).

14.6.5. Внешнее ОЗУ в качестве памяти команд

При частой модификации программ ОВМ происходит деградация ячеек внутреннего или внешнего ПЗУ (см. п. 14.1.5). Чтобы исключить возможность подобных отказов системы, в некоторых случаях в качестве памяти команд используют внешнее ОЗУ, которое не имеет ограничений по количеству циклов перезаписи.

В качестве примера рассмотрим учебный модуль для многократной загрузки программ из компьютера, упрощенная схема которого приведена на рис. 14.25.

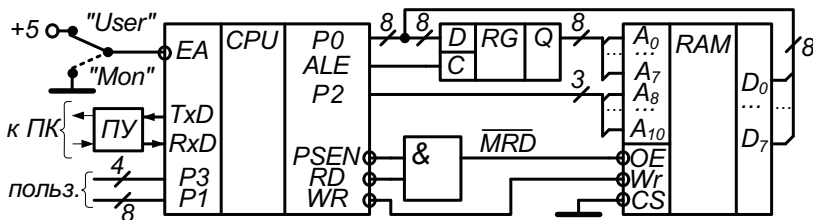


Рис. 14.25

Особенностью использования ОЗУ (например, типа K537PY10) в этой схеме является то, что благодаря объединению сигналов чтения данных (**RD**) и чтения команд (сигнал **PSE**) при помощи элемента «И», ОЗУ может работать как память команд [27].

*Для инверсных сигналов (**PSE** и **RD**) согласно принципу дуальности [23] элемент «И» выполняет функцию логического сложения, то есть инверсный выходной сигнал (**MRD**) будет активным, когда активен, хотя бы один из входных сигналов (**PSE** или **RD**). Таким образом, процессор может читать из ОЗУ коды команд (**PSE**=0) или коды данных (**RD**=0).*

С другой стороны микросхема ОЗУ доступна, и как память данных (синхросигналы **WR** и **RD**), что позволяет процессору записывать или читать коды из ОЗУ.

Учебный модуль имеет два режима работы в зависимости от положения переключателя, устанавливающего сигнал **EA**. В положении «**Mon**» (**EA**=1) процессор работает под управлением кодов из внутреннего ПЗУ, где размещается монитор-загрузчик, выполняющий функции «игрушечной» операционной системы. Эта программа монитора

при помощи интерфейсного преобразователя уровней ПУ получает от компьютера команды и выполняет их, например, загружая в ОЗУ, как в память данных, бинарные коды отлаживаемых процедур.

После загрузки программа компьютера может проверить содержимое ОЗУ, посылая монитору команды чтения из памяти данных.

При установке переключателя в положение «User» ($EA=0$) процессор модуля начинает извлекать коды команд из внешнего ОЗУ, выполняя загруженную программу пользователя.

Отметим, что биты P3.6 (WR) и P3.7 (RD) заняты тактированием внешнего ОЗУ и их нельзя использовать в программе. Последовательный порт находится в распоряжении пользователя и может работать с другой программой ПК.

Использование внешнего или внутреннего ОЗУ в качестве памяти команд имеет ряд преимуществ, кроме возможности многократной модификации программ. Например, в сигнальных процессорах применение быстродействующих ОЗУ вместо сравнительно медленных ПЗУ позволяет увеличить производительность. Программа в этом случае хранится в компьютере (например, драйвер для модема) или во внешнем последовательном ПЗУ и при включении загружается в ОЗУ для исполнения.

§14.7. Подключение к ОВМ в режиме 8-битной адресации

14.7.1. Два типа адресации внешней памяти

При общении с внешним ОЗУ в зависимости от типа используемых команд возможны два варианта адресации внешней памяти, 16-разрядный и 8-разрядный.

В первом случае используются команды пересылок с адресацией через 16-битный указатель данных **DPTR** (см. п.п. 7.8.3 и 14.6.4):

Во втором случае используются команды, применяющие для косвенной адресации байтовые регистры **R0** и **R1** (см. п. 7.8.3):

MovX A, @Rk	; A = @Rk
MovX @Rk, A	; @Rk = A

которые формируют 8-битный адрес при помощи только порта **P0**. Старшая часть адреса при этом не выводится, то есть порт **P2** можно использовать для своих целей, но адресуются только 256 ячеек внешней памяти данных.

Подключение ОЗУ с такой малой емкостью вряд ли целесообразно, так как лучше выбрать другое семейство ОВМ с большей внутренней памятью. Поэтому 8-разрядный тип адресации используется в основном для внешних устройств, которые не могут быть интегрированы в состав ОВМ, например, узкоспециализированных микросхем, индикаторов и пр. Количество в системе таких устройств с параллельным интерфейсом чаще всего невелико, поскольку в большинстве случаев разработчики отдают предпочтение устройствам с последовательным интерфейсом, для взаимодействия с которыми требуется минимум внешних выводов.

14.7.2. Подключение микросхемы часов

В качестве примера использования 8-битного режима адресации рассмотрим подключение к ОВМ микросхемы часов КР512ВИ1 с параллельным интерфейсом (см. п. 13.3.6).

Несмотря на то, что она считается устаревшей, но в некоторых случаях ее применение может оказаться оправданным. Дело в том, что наряду с часами реального времени (Real Time Clock – RTC), она имеет небольшое внутреннее ОЗУ (~50 байт), которое, по сути, является энергонезависимым, поскольку микросхема часов всегда имеет резервное питание. Вдобавок микросхема имеет мультиплексированные выводы адреса/данных, что согласуется с типом передачи адреса/данных порта P0 и позволяет исключить внешний регистр хранения адреса.

На рис. 14.26 приведена упрощенная схема подключения микросхемы часов к порту P0.

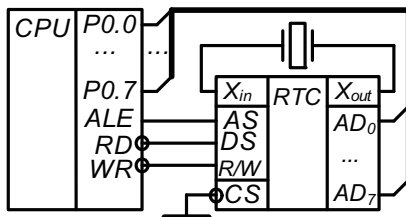


Рис. 14.26

Некоторые выводы и элементы оформления для простоты не показаны, чтобы выделить особенности подключения.

Сразу заметим, что порт P0 напрямую соединяется с выводами адреса/данных микросхемы, причем в качестве синхроимпульса адреса AS используется сигнал ALE от ОВМ, сопровождающий выдачу младшего байта адреса.

Что касается соединения сигналов, тактирующих запись/чтение данных, то, несмотря на внешнюю простоту соединения (RD с DS, WR с R/W), имеется неявная особенность.

Дело в том, что микросхема часов имеет «мотороловский» набор тактирующих сигналов, отличающийся от используемого в ОВМ (см. п. 14.4.3). Сигнал R/W (Read/Write) определяет направление обмена, а DS (Data Strob) – момент. Тогда как в ОВМ, для чтения используется сигнал RD, определяющий ввод данных и момент, а для записи – WR, также стробирующий данные. Однако микросхема часов сама определяет тип тактирующих сигналов, что позволяет исключить дополнительные внешние схемы.

Не останавливаясь на подробностях организации программы для обслуживания, отметим, что на ассемблере могут использоваться четыре команды пересылки (см. п. 7.8.3), две для чтения данных

MovX A, @R0
MovX A, @R1

две для записи

MovX @R0, A
MovX @R1, A

Причем перед выполнением этих команд необходимо задать адрес в соответствующем регистре **R0** или **R1** для косвенной адресации.

14.7.3. Подключение АЦП к порту P0

Если внешнее устройство имеет только один регистр (ячейку), то можно игнорировать внешний адрес и упростить схему за счет исключения регистра адреса. В качестве примера рассмотрим подключение к порту **P0** АЦП с параллельным интерфейсом КР572ПВ3 (см. рис. 14.27).

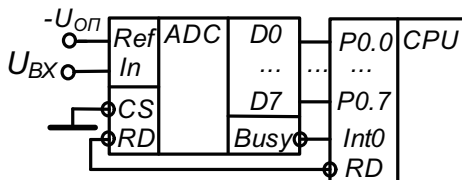


Рис. 14.27

Для анализа сигнала готовности Busy можно использовать механизм прерываний (см. п. 8.4.1) или читать данные спустя время, большее времени преобразования. Временные диаграммы для указанного АЦП здесь не рассматриваются, они приведены в п. 18.3.1, где описана программная реализация параллельного интерфейса.

Чтение данных, как и прежде (см. п. 14.7.1), можно производить двумя ассемблерными командами. Причем адрес в регистре **R0** или **R1** может быть любой, поскольку к порту **P0** подключено единственное устройство. Аналогично и на языке Си для внешнего АЦП также

объявляется единственная переменная, причем ее размещение может быть произвольным.

*Заметим, что при серии последовательных чтений АЦП компилятор в некоторых случаях может «оптимизировать» программу, прочитав АЦП один раз и сохранив результат во внутренней переменной. Чтобы заставить программу каждый раз читать АЦП, переменной mADC желательно присвоить тип **volatile** (см. п. 9.3.3).*

В рассмотренную схему чтения из АЦП можно добавить ЦАП с параллельным интерфейсом, подключенный своими цифровыми входами также к порту *P0*. Конкурировать АЦП и ЦАП не будут, так как они тактируются разными сигналами, сигналом *RD* будет обеспечиваться чтение из АЦП, а сигналом *WR* – запись в ЦАП.

*Если внешних устройств для параллельного ввода-вывода несколько, то их необходимо разделять. Для этой цели, чтобы не увеличивать внешней аппаратуры, управление сигналами выбора устройств (*CS*) можно производить с помощью свободных битов портов, один из которых нужно переводить в активное состояние перед операцией чтения. То есть алгоритм работы параллельного интерфейса порта *P0* становится аппаратно-программным.*

14.7.4. Подключение ЖКИ с параллельным интерфейсом

В ряде случаев параллельный интерфейс внешних устройств может несколько отличаться от интерфейса ОВМ последовательностью тактирования или набором сигналов. В этом случае для устранения различий приходится привлекать некоторые программно-аппаратные средства.

В качестве примера рассмотрим временные диаграммы для ЖКИ типа МТ12864, приведенные на рис. 14.28. Из диаграмм видно, что ЖКИ использует «мотороловский» набор (см. п. 14.4.3) с раздельными синхросигналами направления (*R/W*) и момента (*E*) обмена. Причем сигнал стробирования данных *E*, имеет единичный активный уровень.

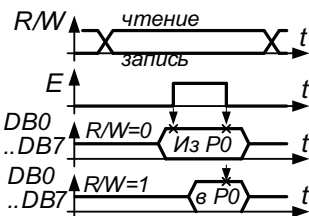


Рис. 14.28

Как правило, управление параллельным интерфейсом ЖКИ осуществляется программным способом, особенно для простых

ЖКИ, в которых шина данных может работать в 4-битном режиме.

Для иллюстрации сочетания аппаратно-программных средств при обслуживании ЖКИ рассмотрим его подключение к порту *P0* в режиме 8-битной адресации (рис. 14.29).

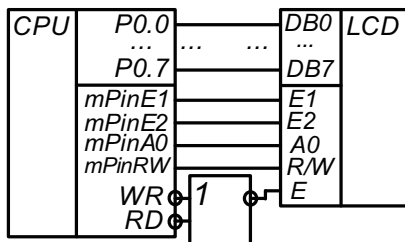


Рис. 14.29

В этой упрощенной схеме для формирования сигнала *E*, тактирующего и запись, и чтение данных, используются объединенные при помощи элемента «И-НЕ» инверсные сигналы *RD* и *WR*.

Согласно принципа «дуальности» [23] элемент «И» для инверсных входных сигналов выполняет логическое сложение. Причем результирующий выходной сигнал тоже будет инверсным, поэтому для получения прямого тактирующего сигнала E применен элемент «И-НЕ» с инверсией.

Остальные управляющие сигналы (*R/W* – направление обмена, *A0* – разделитель данных/команд, *E1* – выбор первого кристалла внутренней памяти, *E2* – выбор второго кристалла) формируются программно на основе битов *mPin*, например, порта *P1*.

Поэтому последовательность обслуживания ЖКИ будет усложнена предварительной установкой управляющих сигналов. Например:

```
mPinE1 = 1;           // Выбираем первый кристалл памяти ЖКИ.
mPinE2 = 0;
mPinA0 = 0;           // Будем передавать команду для ЖКИ.
mPinRW = 0;           // Режим записи в ЖКИ.
mXdataVar = mKomanda; // Запись данных для ЖКИ через порт P0.
```

Другие особенности управления ЖКИ выходят за рамки пособия и здесь не рассматриваются.

15 АНАЛОГОВЫЕ КЛЮЧИ И КОММУТАТОРЫ

Для подключения сигналов с аналоговых датчиков к входу АЦП, соединенного с ВМ, служат аналоговые коммутаторы (мультиплексоры). На управляющих входах аналогового коммутатора действуют цифровые сигналы, разрешающие или запрещающие прохождение аналогового сигнала с одного из входов на общий выход.

Основными элементами коммутатора являются аналоговые ключи, которые входят в состав и более сложных устройств, таких как ЦАП и АЦП, и оказывают существенное влияние на их характеристики.

§15.1. Аналоговый ключ

15.1.1. Общие сведения

Аналоговый ключ, в отличие от цифрового, работает с непрерывным множеством значений и должен максимально точно передавать сигнал от источника информации к приемнику. Поэтому аналоговый ключ по своим свойствам ближе к идеальному ключу, то есть имеет минимальное сопротивление в проводящем состоянии и максимальное – в непроводящем.

Информационный сигнал, обычно представляющий напряжение, отсчитывается относительно общей точки системы, условно называемой «землей». Поэтому аналоговый ключ, подсоединяемый между источником сигнала и приемником (нагрузкой), чаще всего включается непосредственно в информационный провод, как показано на функциональной схеме, приведенной на рис. 15.1.

В качестве ключей можно использовать и контактные ключи, напри-

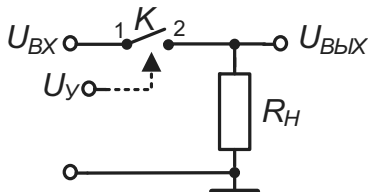


Рис. 15.1

мер, герконы, которые рассмотрены в п. 15.4.3. Но их быстродействие и ресурс работы не велики, и поэтому в микропроцессорной технике в основном применяются бесконтактные ключи на основе полевых транзисторов с изолированным затвором.

Например, на схеме рис. 15.2 в качестве ключа K использован полевой транзистор с изолированным затвором и каналом p -типа. В принципе тип транзистора зависит от используемой технологии и реально в одном ключе могут комбинироваться несколько транзисторов для улучшения его характеристик. Например, параллельное соединение двух разнотипных транзисторов уменьшает сопротивление ключа в проводящем состоянии и позволяет коммутировать двухполярное входное напряжение.

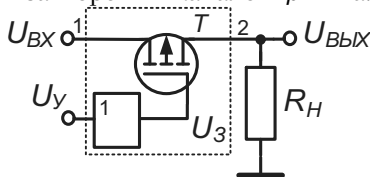


Рис. 15.2

Вывод 1 аналогового ключа называют входом, а вывод 2 – выходом. Отметим, что вход и выход можно поменять местами, то есть *аналоговый ключ* в отличие от цифрового (см. п. 11.8.2) является *обратимым элементом*.

Управление ключом осуществляется напряжением на затворе транзистора. Оно должно обеспечивать надежное переключение при заданных величинах и полярностях входного сигнала. Формирование напряжения U_3 обеспечивается специальными каскадами внутри микросхем и скрыто от пользователей. Наружное управление аналоговым ключом обычно осуществляется цифровым сигналом U_y , близким по уровням к ТТЛ сигналу.

Принято считать сигнал U_y активным, когда ключ проводит, и пассивным, когда не проводит. Если активный уровень нулевой, то вход называют инверсным и на УГО помечается кружком. Если активный уровень единичный, то управляющий вход называют прямым (см. п. 15.1.5).

15.1.2. Проводящее состояние ключа

Будем считать, что напряжение U_y имеет активный уровень. Тогда ключ находится в проводящем состоянии и его в простейшем случае можно заменить внутренним сопротивлением канала R_{ON} транзистора (рис. 15.3). Значение R_{ON} зависит от типа ключа, приводится

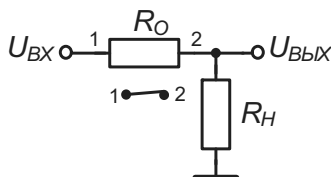


Рис. 15.3

в справочниках и имеет величину от единиц до сотен *Омов*.

$$U_{ВЫХ} = U_{ВХ} \cdot \frac{R_H}{R_H + R_{ОН}}. \quad (15.1)$$

Полученную схему замещения можно использовать для оценки влияния ключа на выходной сигнал:

Из формулы видно, что дробь меньше единицы и $U_{ВЫХ} < U_{ВХ}$, то есть ключ ослабляет сигнал и вносит определенную погрешность.

$$\delta_{ОН} = \frac{U_{ВХ} - U_{ВЫХ}}{U_{ВХ\max}}. \quad (15.2)$$

Относительную погрешность для проводящего состояния можно рассчитать, как отношение напряжения, падающего на ключе, к максимальному входному сигналу:

$$\delta_{ОН} \approx \frac{R_{ОН}}{R_H}. \quad (15.3)$$

Подставив сюда значение $U_{ВЫХ}$ при $U_{ВХ\max}$, получим (при $R_H \gg R_{ОН}$):

Пусть, например, $R_{ОН} = 50$ Ом, а $R_H = 10$ кОм, тогда $\delta_{ОН} = 0,5\%$. Здесь следует отметить, что не так важна величина этой погрешности (ведь ее можно учесть после оцифровки, введя поправочный коэффициент в программу), как ее неизменность. Дело в том, что $R_{ОН}$ для конкретного ключа в точности неизвестно, поскольку производитель дает чаще всего максимально возможные значения. Вдобавок нужно помнить, что $R_{ОН}$ нелинейно зависит от входного сигнала и, кроме того, существенно меняется при изменении температуры. Поэтому на практике стараются максимально увеличивать R_H , чтобы даже в худшем случае погрешность была не более заданной. Один из способов увеличения R_H путем подключения нагрузки к коммутатору через повторитель сигнала с большим входным сопротивлением иллюстрирован в п. 15.2.3.

15.1.3. Непроводящее состояние ключа

Характеризуется огромным сопротивлением (более *ГОм*), которое в большинстве случаев не учитывается в расчетах. Однако паразитный ток через ключ все-таки протекает — это достаточно малый ток утечки через различные внутренние цепи, попадающий на нагрузку. Поэтому эквивалентная схема замещения ключа в непроводящем состоянии (рис. 15.4) учитывает этот источник

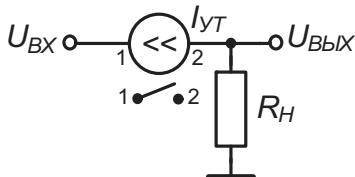


Рис. 15.4

тока I_{YT} . Отметим также, что на направление и величину этого тока дополнительно оказывает влияние полярность и величина входного сигнала

$$U_{BbIX} = I_{YT} R_H. \quad (15.4)$$

Нетрудно видеть, что на выходе будет отличное от нуля неинформационное напряжение:

$$U_{BbIX} = I_{YT} R_{ON}. \quad (15.5)$$

Однако в составе коммутатора ток утечки непроводящего ключа будет замыкаться через малое сопротивление проводящего ключа в соседнем канале, и тогда паразитная составляющая будет значительно меньше:

$$\delta_{OFF} = \frac{U_{BbIX}}{U_{BXmax}} = \frac{I_{YT} R_{ON}}{U_{BXmax}}. \quad (15.6)$$

Для оценки относительной погрешности нужно поделить эту неинформационную составляющую U_{BbIX} на максимальный входной сигнал:

Например, имеем справочные значения $I_{YT} = 1 \text{ мкА}$, $R_{ON} = 50 \text{ Ом}$, а напряжение $U_{BXmax} = 1 \text{ В}$ известно из технического задания. Тогда $\delta_{OFF} = 10^{-6} \cdot 50 / 1 = 0,005\%$.

И опять отметим, что поскольку точное значение тока I_{YT} для конкретного ключа неизвестно, то речь идет только об упрощенной оценке максимально возможного значения погрешности. На практике нужно также учитывать температурные изменения тока I_{YT} , которые могут быть весьма значительными.

15.1.4. Динамическая погрешность ключа

Кроме статических, медленно изменяющихся погрешностей, которые мы рассмотрели, в аналоговом ключе имеются и динамические составляющие, вызванные перезарядом внутренних паразитных емкостей при переключении управляющего сигнала. Токи заряда и разряда этих емкостей просачиваются в информационный канал и образуют на нагрузке так называемые *всплески* (см. рис. 15.5).

Эти всплески подмешиваются к информационному сигналу и вносят определенную погрешность, зависящую от способа оцифровки. Например, при

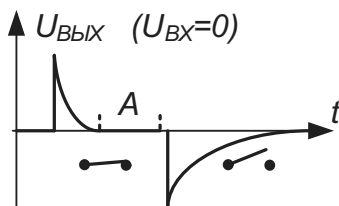


Рис. 15.5

использовании в цепи обработки сигнала осредняющих цепей или интегрирующего АЦП (см. п. 18.5.1) погрешность будет максимальна. Если же осреднения нет, то стараются проводить оцифровку спустя некоторое время после переключения (участок «А» на рис. 15.5).

Подобная динамическая погрешность не всегда указывается в документации, чаще приводится *время переключения* из одного состояния в другое (десятки и сотни наносекунд), которое характеризует потенциальное быстроедействие ключа.

15.1.5. Обозначение аналогового ключа

На принципиальных схемах внутреннюю структуру ключа не изображают, а пользуются стандартным обозначением в виде прямоугольника (рис. 15.6), внутри которого указывают символ функции, выполняемой ключом – SW (от *SWitch*). Аналоговый и цифровой управляющий входы ключа расположены, как обычно, с левой стороны, а выход – с правой. Аналоговые выводы чаще всего не маркируются отдельными буквами, а управляющий вход, для отличия, имеет буквенное обозначение, например, E (от *Enable* – разрешить).

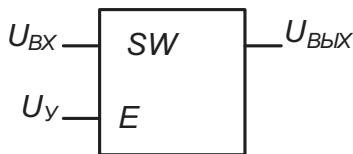


Рис. 15.6

15.1.6. Выбор микросхемы ключа

Микросхемы аналоговых ключей в настоящее время выпускают множество зарубежных фирм. В России номенклатура выпускаемых ключей значительно меньше и ограничивается в основном устаревшими микросхемами серии К590, имеющими буквенный индекс КН, например, КР590КН13.

При выборе типа ключа обычно обращают внимание на время переключения, значения R_{ON} и $I_{УТ}$, напряжение питания микросхемы, диапазон переключаемых напряжений и уровни управляющего сигнала.

Одиночные аналоговые ключи используются сравнительно редко. В основном они входят в состав более сложных устройств, таких как коммутаторы, схемы выборки-хранения, ЦАП, АЦП, синхронные детекторы, усилители с программируемым коэффициентом усиления, усилители постоянного тока с коррекцией нуля и др. Часть из этих ключевых устройств будет рассмотрена далее.

§15.2. Аналоговый коммутатор

15.2.1. Внутренняя структура и обозначение

Коммутатор по своей структуре (рис. 15.7) представляет собой набор аналоговых ключей, выходы которых объединены. Для уменьшения числа управляющих внешних выводов используется внутренний дешифратор *DC*. Сигналы на входе дешифратора задают в двоичном виде номер выбранного канала, то есть активным будет только один управляющий выход дешифратора. В том случае, если разрешающий сигнал дешифратора *E* пассивен, все ключи находятся в непроводящем состоянии. Этот режим позволяет подключать выходы нескольких микросхем коммутаторов на одну нагрузку, разрешая работу только одному из них по входу *E*.

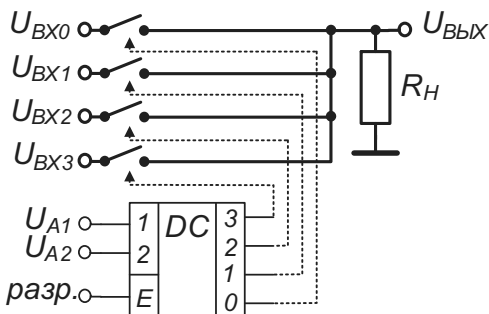


Рис. 15.7

Условное графическое обозначение аналогового коммутатора (рис. 15.8) похоже на обозначение цифрового (см. п. 11.8.3), имеет входы и выходы, адресные входы и вход разрешения. Отличие состоит только в том, что в буквенном обозначении выполняемой

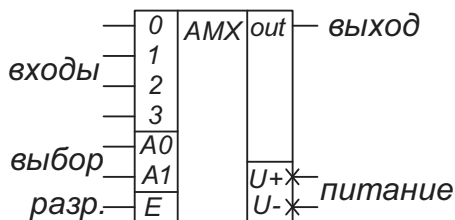


Рис. 15.8

функции коммутатора (мультиплексора) присутствует буква *A* и информационные цепи – аналоговые. Допускается вместо *AMX* использовать обозначение *SW*, как в электронных ключах.

Отметим, что обратимость аналоговых коммутаторов (выход может стать входом) позволяет использовать их в режиме демultipлексора (см. п. 11.8.1), когда один источник информации подключен к выходу, который становится входом, а адресуемые входы становятся выходами информации, к которым подключены приемники.

15.2.2. Параметры

Микросхемы коммутаторов характеризуются в основном такими же параметрами, как и аналоговые ключи – R_{ON} , I_{UT} , время переключения, диапазон входных напряжений, напряжение питания и уровни управляющих сигналов.

Например, 8-входовая микросхема КР590КН6 имеет следующие параметры: $R_{ON} \approx 300 \text{ Ом}$, $I_{UT} \approx 70 \text{ нА}$, время переключения 300 нс . [14]. Указанные параметры зависят от напряжения питания, уровня входного сигнала, температуры и поэтому справочные данные лишь приблизительно указывают границу параметров для определенного режима тестирования.

Следует отметить, что микросхемы серии К590 имеют двухполярное напряжение питания, тогда как большинство современных микропроцессорных систем имеет единственный источник +5 В или менее. Поэтому разработчики применяют или импортные коммутаторы, или в не очень ответственных случаях пользуются цифровыми микросхемами, позволяющими коммутировать и аналоговые сигналы. К таким микросхемам относятся КР561КП1 (два коммутатора с 4 входами) и КП2 (8 входов), имеющие при пятивольтовом питании $R_{ON} \approx 200...400 \text{ Ом}$, $I_{UT} \approx 20 \text{ нА}$, время переключения 30 нс . Уровни управления: $U^0 \approx 0$, $U^1 \approx 5 \text{ В}$.

15.2.3. Пример использования аналогового коммутатора

Пусть имеются восемь датчиков, которые необходимо по программе подключать к АЦП и оцифровывать соответствующий параметр. В структурной схеме, приведенной на рис. 15.9, для этой цели используются аналоговый коммутатор, управляемый от ОВМ x51, буферный повторитель и АЦП.

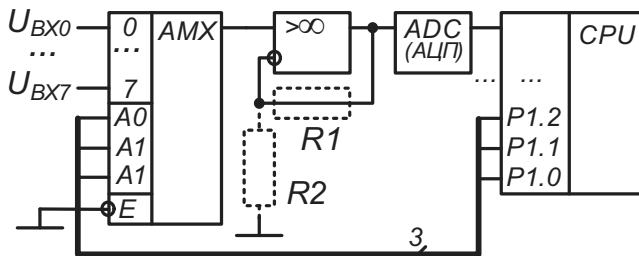


Рис. 15.9

Повторитель сигнала играет особую роль. Имея, как известно (см. п. 6.2.1), очень большое входное сопротивление, он практически не нагружает коммутатор, и вследствие этого снижается погрешность в проводящем состоянии ключей. Поэтому в формуле (15.3) для расчета погрешности ключа в проводящем состоянии вместо сопротивления R_H нужно использовать входное сопротивление повторителя

$$R_{вх} = R_y (1 + K). \quad (15.7)$$

Нужно, конечно, помнить, что вместе с ИОУ вносятся и его погрешности, в основном обусловленные входным током и напряжением смещения микросхемы усилителя (см. §5.7).

ОВМ по программе должна задать адрес выбранного канала, запустить АЦП после некоторой паузы, учитывающей время переключения коммутатора, и, дождавшись окончания преобразования, прочесть результат. Особенности обслуживания АЦП будут рассмотрены в главе 18, здесь мы остановимся на особенностях управления коммутатором.

Будем считать, что в программе определена байтовая переменная **myNomerK**, которая задает номер канала коммутатора, требующий обслуживания.

Обычная операция копирования (**Mov P1, myNomerK**) в данном случае непригодна, так как переменная **myNomerK** может принимать значения от 0 до 7, поэтому старшие пять битов порта **P1** будут обнулены, что нарушит работу системы, потому что эти биты могут использоваться для обслуживания других объектов.

Поэтому требуется неразрушающее копирование только трех битов. Для этого программа должна прочесть текущее значение порта **P1** и, обнулив при помощи операции **AND** три младших бита, поместить на их место значение **myNomerK** с помощью операции **OR**:

Mov A, P1	; Читать биты порта P1.
Anl A, #11111000b	; Обнулить три младших бита.
Orl A, myNomerK	; Поместить НомерКанала.
Mov P1, A	; Вернуть P1 с измененными битами

Подобные системы, реализованные в виде БИС и имеющие в своем составе коммутатор, АЦП и процессорное ядро, часто называют системами сбора аналоговых данных или даже (преувеличенно) аналоговыми процессорами.

15.2.4. Разновидности микросхем аналоговых коммутаторов

Большинство коммутаторов имеют параллельную и асинхронную форму задания номера канала. Некоторые из микросхем имеют дополнительно внутренний регистр для хранения двоичного адресного кода и параллельный интерфейс с сигналами синхронизации, позволяющий подключаться к информационным шинам микропроцессора.

При удаленном расположении обслуживаемых датчиков или большом числе коммутируемых каналов подобный параллельный интерфейс становится неэкономичным, и поэтому в некоторых микросхемах используют последовательные интерфейсы с меньшим числом проводов. Не останавливаясь на подробностях, которые изучаются в курсе «Сети ЭВМ», отметим, что такими интерфейсами, используемыми для управления коммутаторами, являются трехпроводный *SPI* (см. §12.8) и двухпроводный *I²C* [25].

Для сложных соединений аналоговых сигналов, когда необходимо соединить несколько источников сигнала с несколькими приемниками, разработаны специальные матричные коммутаторы [2].

§15.3. Схема выборки-хранения

15.3.1. Схема и принцип действия

Схема выборки-хранения, иногда называемая ячейкой аналоговой памяти, применяется для фиксации мгновенных значений быстроизменяющихся аналоговых сигналов на время, необходимое АЦП для преобразования напряжения в код. При использовании таких устройств *уменьшается* динамическая (апертурная) погрешность АЦП.

Простейшая схема имеет всего два функциональных элемента, ключ *K* и конденсатор *C* (рис. 15.10). Но на работу устройства оказывают влияние и внешние факторы, поэтому на схеме показаны сопротивление нагрузки *R_н*, сопротивление источника сигнала *R_и*. Явно не выделены остаточные параметры ключа, но и они вносят свою долю погрешности.

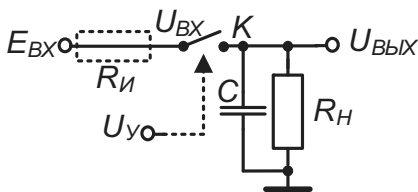


Рис. 15.10

Предположим, что в исходном состоянии конденсатор был разряжен. Поэтому в момент перехода ключа в проводящее состояние (этап выборки сигнала) начнется заряд конденсатора (рис. 15.11). И конденсатор должен запомнить U_{BX} , то есть его напряжение должно достигнуть уровня входного сигнала, прежде чем выборка закончится.

Затем ключ под действием управляющего сигнала перейдет в непроводящее состояние, а конденсатор должен сохранить запомненное U_{BX} до окончания интервала хранения – пока АЦП не закончит оцифровку.

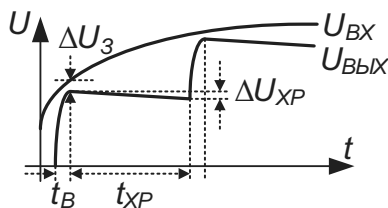


Рис. 15.11

Основные требования – малые погрешности при записи (выборке) и хранении. Однако процессы в этой схеме имеют экспоненциальный характер, поэтому возникают погрешности, как на этапе выборки, так и на этапе хранения.

15.3.2. Погрешность недозаряда

Считая на коротком этапе выборки напряжение U_{BX} неизменным и $(R_{\text{и}} + R_{\text{ОН}}) \ll R_{\text{н}}$, напряжение на конденсаторе можно записать в виде стандартного решения дифуравнения первого порядка:

$$U_c(t) = U_c(\infty) - [U_c(\infty) - U_c(0)] \cdot \exp[-t / (R_{\text{и}} + R_{\text{ОН}}) C]. \quad (15.8)$$

В нашем случае $U_c(t) = U_{\text{ВЫХ}}(t)$. Тогда, учитывая, что на интервале выборки $U_c(\infty) = U_{\text{ВХ}}$, а $U_c(0) = 0$, найдем

$$U_{\text{ВЫХ}}(t) = U_{\text{ВХ}} - U_{\text{ВХ}} \cdot \exp[-t / (R_{\text{и}} + R_{\text{ОН}}) C]. \quad (15.9)$$

Время выборки t_B не может быть большим, поэтому выходной экспоненциальный сигнал в момент перехода в режим хранения не будет в точности равен $U_{\text{ВХ}}$, и недозаряд емкости можно оценить, как разность входного и выходного напряжений:

$$\Delta U_3 = U_{\text{ВХ}} - U_{\text{ВЫХ}}(t_B) = U_{\text{ВХ}} \exp[-t_B / (R_{\text{и}} + R_{\text{ОН}}) C]. \quad (15.10)$$

Используя эту формулу, по заданным значениям ΔU_3 и t_B можно определить постоянную времени $(R_{\text{и}} + R_{\text{ОН}})C$ и найти затем максимальное значение емкости конденсатора C .

Для уменьшения влияния сопротивления источника сигнала $R_{\text{и}}$ на заряд обычно используют дополнительный повторитель сигнала (рис. 15.12). При этом напряжение $E_{\text{ВХ}}$ передается на ключ практически без ослабления. А выходное сопротивление повторителя,

благодаря глубокой ООС, оказывается значительно меньшим сопротивления проводящего ключа R_{ON} . Поэтому в формуле (15.10) для данного случая можно не учитывать $R_{и'}$.

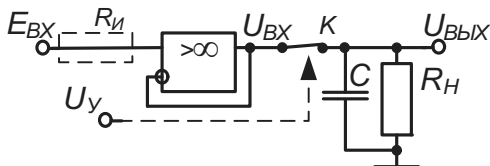


Рис. 15.12

15.3.3. Погрешность хранения

На этапе хранения ключ не проводит и выходное напряжение не остается неизменным, поскольку идет разряд конденсатора током нагрузки и токами утечки ключа. Причем характер изменения напряжения близок к линейному и можно воспользоваться упрощенной оценкой, используя формулу

$$C = \Delta Q / \Delta U_{xp}, \quad (15.11)$$

где

$$\Delta Q = \Delta t I = t_{xp} \cdot (I_{yT} + I_H), \quad (15.12)$$

где I_H – ток через нагрузку R_H .

Для уменьшения тока I_H на этапе хранения нагрузку обычно подключают через дополнительный повторитель (рис. 15.13). В этом случае входное сопротивление повторителя практически не будет нагружать емкость, но входной ток ИОУ $I_{вх}$ учитывать приходится. Поэтому изменение выходного напряжения на этапе хранения можно оценить по формуле:

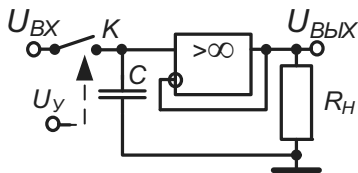


Рис. 15.13

$$\Delta U_{xp} = t_{xp} \cdot (I_{yT} + I_{вх}) / C. \quad (15.13)$$

Отметим, что погрешность схемы выборки-хранения имеет и динамическую составляющую (см. п. 15.1.4), например, из-за прохода U_y через паразитные емкости.

15.3.4. Микросхема выборки-хранения КР1100СК2

имеет кроме ключа и внешнего запоминающего конденсатора (рис. 15.14) два повторителя сигнала, на входе и выходе, которые уменьшают влияние источника сигнала и нагрузки на заряд и разряд конденсатора. Особенностью входного повторителя является обрат-

ная связь непосредственно с выхода устройства, что позволяет форсировать заряд конденсатора во время выборки.

Схема работоспособна [14] при $C = 10 \dots 1000$ пФ и $t_B = 5 \dots 10$ мкс.

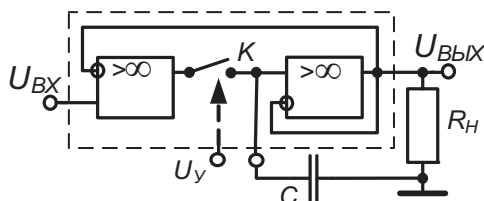


Рис. 15.14

В настоящее время схемы выборки-хранения выполняются обычно внутри микросхем АЦП и более сложных систем сбора аналоговых данных.

§15.4. Аналоговые ключи с изолированным управлением

В некоторых случаях при оцифровке сигналов, например, на фоне больших паразитных составляющих, приходится использовать аналоговые ключи с изолированным управлением. Передача сигнала управления на ключ при этом осуществляется магнитным полем (через импульсный трансформатор или электромагнитное реле) или оптическим излучением (через оптрон).

15.4.1. Ключ с трансформаторным управлением

При использовании импульсного трансформатора (см. п. 4.4.1) для передачи сигнала управления (рис. 15.15) необходимо помнить, что трансформатор имеет ограничения на длительность передаваемых импульсов (см. п. 4.5.3) и вдобавок низкочастотные трансформаторы имеют большие габариты. Поэтому подобные ключи применяют для сравнительно высоких частот коммутации.

Другим вариантом использования трансформатора является передача сигнала управления пачками импульсов, путем амплитудной

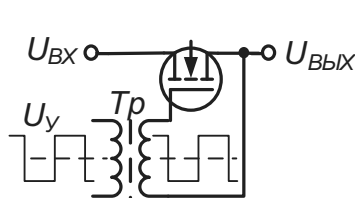


Рис. 15.15

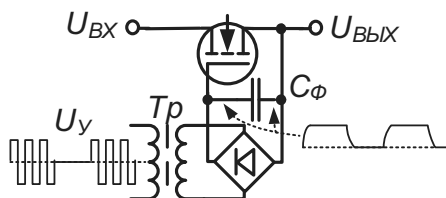


Рис. 15.16

модуляции высокочастотного сигнала (рис. 15.16). Но и здесь есть свои ограничения, связанные с затянутостью фронтов переданного и выпрямленного сигнала из-за наличия фильтра, подавляющего гармоники несущей частоты.

Трансформаторное управление аналоговыми ключами в настоящее время используется сравнительно редко, поскольку изготовление трансформатора достаточно трудоемко. Однако в высокоскоростных цифровых трактах передачи информации иногда используют миниатюрные трансформаторы, которые конкурирует с оптронами, обеспечивая большее быстродействие и хорошее согласование цепей.

15.4.2. Ключ с оптическим управлением

Оптроны более технологичны, чем трансформаторы, и поэтому широко используются для гальванического разделения цепей. Наиболее простыми оптически управляемыми ключами являются транзисторные оптроны, состоящие из светодиода (излучателя) и фототранзистора (приемника). Подобные оптически управляемые ключи, ориентированные на коммутацию больших уровней тока и напряжения, называют *оптореле*. Отметим, что оптореле в большинстве случаев непригодны для работы с аналоговыми датчиками.

Например, *биполярный фототранзистор* в оптроне (рис. 15.17) имеет существенное остаточное напряжение в доли вольта и поэтому такие оптроны используются только в цифровых или высокоуровневых ключевых схемах (см. п. 4.1.1).

Оптроны на основе *полевых фототранзисторов* (рис. 15.18) более пригодны для коммутации аналоговых сигналов от датчиков. Например, оптореле К249КП4АТ [14] с двумя отдельными ключами, предназначенное для коммутации сигналов до 60 В, имеет $R_{ON} < 15 \text{ Ом}$, $I_{UT} < 1 \text{ мА}$.

Общим недостатком оптически управляемых биполярных и полевых фототранзисторов является сравнительно большое время переключения (единицы и десятки мкс и более), особенно при переходе в непроводящее состояние.

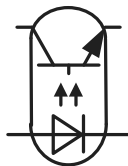


Рис. 15.17

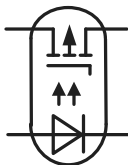


Рис. 15.18

15.4.3. Контактные ключи

В ряде случаев при коммутации сигналов милливольтового уровня приходится применять более близкие к идеальным контактные

ключи, которые имеют значительно меньшую погрешность. Известны две основных разновидности управляемых контактных ключей – электромеханическое реле и магнитоуправляемый контакт (геркон).

В *реле* используется подвижный якорь, притягиваемый магнитным полем к сердечнику катушки и воздействующий через механическую тягу на контакты, то есть между магнитным полем и контактами есть механический посредник и поэтому быстродействие реле невелико.

Другой тип контактного ключа – *геркон*, (сокращение термина «**ГЕР**метизированный **КОН**такт») имеет большее быстродействие, поскольку в нем магнитное поле непосредственно воздействует на ферромагнитный контакт.

Геркон представляет собой стеклянную капсулу (рис. 15.19), внутри которой вакуум или нейтральный газ. В капсулу помещены контактные пластинки «1» и «2» из ферромагнитного материала. При действии на капсулу внешнего магнитного поля, направленного вдоль пластинок, последние притягиваются друг к другу, обеспечивая электрический контакт.

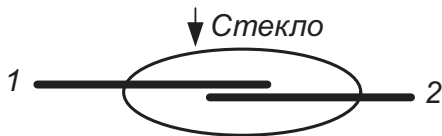


Рис. 15.19

Герконы без катушки часто используются в качестве пороговых датчиков внешнего магнитного поля в охранных системах и системах позиционирования. Например, геркон срабатывает, когда к нему приближается внешний магнит.

Внешнее поле может также создаваться катушкой, надеваемой на капсулу. Такая комбинация геркона и катушки называется *герконовым реле*. Очевидно, что быстродействие герконов больше, чем у электромеханических реле, поскольку отсутствует механическая тяга, которая вносит дополнительную инерционность. Например, время переключения герконового реле типа РЭС-44 составляет величину около 1 *мс*, против десятков *мс* у обычных реле.

Однако у герконов имеются и *два недостатка*, про которые нужно помнить.

Во-первых, это большее, чем у электромагнитных реле контактное сопротивление $R_{\text{он}}$, достигающее 0,1..0,2 *Ом*.

Во-вторых, *ограниченный ресурс* герконов, поскольку при большом числе срабатываний ферромагнитные пластинки устают и теряют свои пружинящие свойства. Например, при высокой частоте

коммутации ресурс в миллион срабатываний можно выработать за несколько дней.

Но альтернатив при коммутации милливольтовых сигналов практически нет, поскольку параметры герконов близки к идеальным – сопротивление $R_{он}$ значительно меньше, чем у транзисторных ключей, ток утечки практически отсутствует. Поэтому погрешность подобного коммутатора большей частью будет определяться внешними факторами, например, очень малыми термоЭДС в местах соединения подводящих проводов.

Управление обмотками герконовых реле осуществляется обычно через транзисторный ключ, причем нужно иметь в виду, что обмотка представляет собой индуктивную нагрузку, подобно обмотке импульсного трансформатора. Поэтому при отключении обмотки в ней возникает обратный выброс (см. п. 4.6.3) и в схеме (рис. 15.20) должна быть предусмотрена диодная цепочка в коллекторной цепи, которая при отключении обмотки создает путь для протекания индуктивного тока, уменьшает амплитуду этого выброса и предохраняет транзистор от перенапряжения. Для повышения быстродействия последовательно с диодом обычно включают дополнительный резистор, который уменьшает постоянную времени индуктивной цепи, но при этом напряжение выброса несколько увеличивается (см. п. 4.7.4).

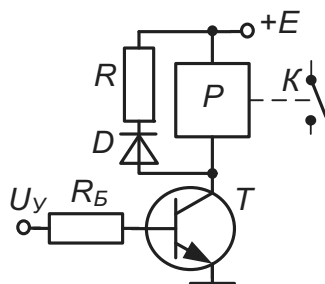


Рис. 15.20

15.4.4. «Летающий конденсатор»

В качестве примера использования ключей с изолированным управлением рассмотрим функциональную схему одного канала коммутатора *без гальванической связи* входа и выхода. В устройстве (рис. 15.21) использованы запоминающий конденсатор, как в схеме выборки-хранения (см. п. 15.3.1) и две пары аналоговых ключей K_1, K_2 и K_3, K_4 , работающих поочередно и обеспечивающих гальваническое разделение.

Когда проводит первая пара ключей K_1 и K_2 , подключая конденсатор для запоминания к входному напряжению, другая пара

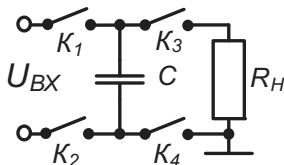


Рис. 15.21

не проводит, обеспечивая отделение входа от выхода. Затем начинают проводить ключи K_3 , K_4 , которые подключают конденсатор, заряженный до $U_{ВХ}$, к выходу схемы. В этот момент должна происходить оцифровка этого напряжения. Ключи K_1 , K_2 на этом этапе не проводят, обеспечивая в свою очередь гальваническое разделение входа и выхода.

На основе «летающего конденсатора» можно реализовать многоканальный коммутатор для сбора данных, не имеющих общей точки с АЦП. Кроме того схемы с переключаемыми конденсаторами иногда используются в ЦАП и АЦП [2]. Конденсаторы в сочетании с ключами применяются также для преобразования величины и полярности постоянного напряжения (см. п. 6.6.5).

16 ЦИФРО-АНАЛОГОВЫЕ ПРЕОБРАЗОВАТЕЛИ

В ряде случаев микропроцессорной системе для управления внешними объектами необходимо представить цифровую информацию в пропорциональном аналоговом виде. Делается это при помощи ЦАП, в котором сочетаются цифровые входные цепи и аналоговые выходные. Ниже рассмотрены структура и свойства подобных устройств.

§16.1. Структура ЦАП

16.1.1. Общие сведения

Передаточная характеристика ЦАП (зависимость $U_{\text{Вых}}$ от входного двоичного кода $X_{\text{ВХ}}$) из-за дискретности входного сигнала также является дискретной (рис. 16.1). Основными параметрами ЦАП являются число разрядов входного кода N и величина кванта (дискреты) выходного напряжения U_0 .

От разрядности N зависит число возможных уровней передаточной характеристики от 0 до X_M , где

$$X_M = 2^N - 1.$$

Например, при $N = 8$ значения чисел 0, 1, 2, .. 255 преобразуются в значения уровней напряжения 0, U_0 , $2U_0$, .. $255U_0$.

ЦАП условно можно рассматривать как потенциометр с цифровым управлением (рис. 16.2). Выходное напряжение при этом представляет собой часть некоторого стабильного напряжения, называ-

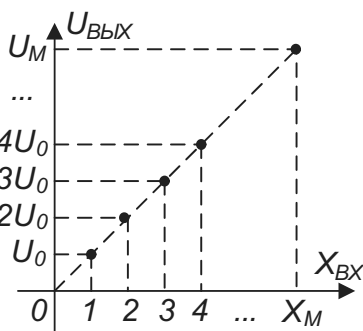


Рис. 16.1

(16.1)

емого обычно опорным. Движок этого иллюстративного ЦАП управляется входным двоичным кодом $X_{ВХ}$.

Если входной код $X = 0$, то движок находится в самом низу и напряжение $U_{ВЫХ} = 0$.

Если код $X = X_M$, то движок условного потенциометра должен находиться в верхнем положении. Тогда напряжение $U_{ВЫХ}$ будет равно $U_{ОП}$ и, как следует из передаточной характеристики, должно быть равно U_M , то есть

$$U_M = U_{ОП} = X_M \cdot U_0 \quad (16.2)$$

Отсюда можно получить формулу для определения дискреты ЦАП, учитывая, что напряжение $U_{ОП}$ обычно известно, а X_M зависит от разрядности ЦАП и рассчитывается по формуле (16.1):

$$U_0 = U_{ОП} / X_M = U_{ОП} / (2^N - 1) = U_M / (2^N - 1). \quad (16.3)$$

Значение U_0 обычно используется для оценки разрешающей способности, «цены» единицы младшего разряда (**LSB**) выводимого кода и основной погрешности ЦАП, определяемой по формуле:

$$\delta_{ОСН} = \pm 0,5 U_0 / U_M = \pm 0,5 / X_M = \pm 0,5 / (2^N - 1). \quad (16.4)$$

Например, для 8-битного формата $\delta_{ОСН} = \pm 0,5 / 255 \approx \pm 0,2\%$, что вполне допустимо для простейших устройств.

16.1.2. Основные параметры ЦАП

Характеристика реального ЦАП может отличаться от идеальной своим наклоном, ненулевым уровнем и нелинейностью. Поэтому кроме разрешающей способности, которая определяется разрядностью входного кода и характеризуется величиной $\delta_{ОСН}$, микросхемы ЦАП характеризуются соответственно погрешностями (см. п. 5.7.1) – *мультипликативной, аддитивной и нелинейности*. Эти погрешности приводятся в справочниках в единицах младшего разряда **LSB** или в долях % и зависят от температуры. В особую группу по рассмотренным параметрам выделяются прецизионные ЦАП, которые имеют повышенную разрядность ($N = 16...18$) и высокую точность.

Динамические свойства ЦАП характеризуют подобно ИОУ (см. п. 5.4.6) *временем установления* выходного напряжения в зону точности при скачкообразном изменении входного кода от 0 до X_M . По этому параметру различают высокоскоростные ЦАП, имеющие $t_{уст}$ в

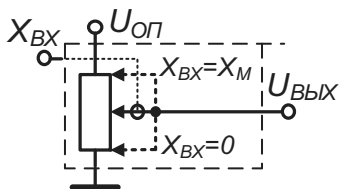


Рис. 16.2

доли *мкс* и микросхемы общего применения с $t_{уст}$ равным единицам и даже десяткам *мкс*. Кроме параметра $t_{уст}$ иногда используют скорость нарастания выходного напряжения, измеряемую в *В/мкс*.

При выборе ЦАП учитывают также напряжение питания, уровни входных управляющих сигналов, потребляемую мощность и тип интерфейса (см. §16.2) для входных данных.

16.1.3. Структура ЦАП

Учитывая, что рассмотренный ранее аналоговый ключ (см. п. 15.1.1) при $U_{ВХ}=U_0$ является, по сути, одноразрядным ЦАП, можно реализовать многоразрядный ЦАП на основе набора ключей, используя формулу для представления двоичного числа:

$$X_{ВХ} = 1Q_0 + 2Q_1 + 4Q_2 + 8Q_3 + \dots, \quad (16.5)$$

где Q – двоичные значения («0» или «1») соответствующих разрядов входного кода, а двоично-взвешенные коэффициенты при них называют весами или значимостью разрядов.

Из формулы видно, что для реализации ЦАП необходимо просуммировать коэффициенты веса отдельных разрядов с учетом значения Q . Делается это самыми различными схемами на основе резисторов, конденсаторов, транзисторных источников токов и пр. Ниже рассмотрены наиболее простые из них, резисторные.

16.1.4. ЦАП на основе двоично-взвешенных резисторов

Основой иллюстративного 4-разрядного ЦАП (рис. 16.3) является инвертирующий сумматор (см. п. 6.1.1) двоично-взвешенных токов на ИОУ с аналоговыми ключами для реализации двоичных значений

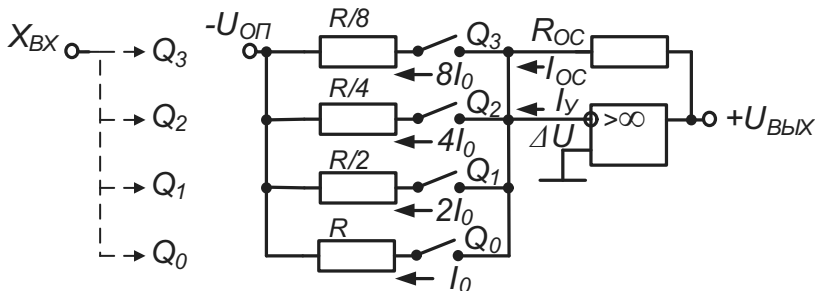


Рис. 16.3

Q. В этой схеме каждый последующий резистор в два раза меньше предыдущего, то есть используется, как говорят, матрица двоично-взвешенных резисторов, формирующая двоично-взвешенные токи.

Найдем выходное напряжение этой схемы, записав уравнение для контура « $U_{\text{Вых}} - R_{\text{OC}} -$ входы ИОУ»:

$$U_{\text{Вых}} = I_{\text{OC}} R_{\text{OC}} + \Delta U. \quad (16.6)$$

Для упрощения расчетов будем считать, что ИОУ имеет очень большой коэффициент усиления и следствием этого будет ничтожность двух параметров (см. п. 5.6.5), напряжения

$$\Delta U = U_{\text{Вых}} / K_{\text{ИОУ}} \approx 0 \quad (16.7)$$

и тока

$$I_y = \Delta U / R_{\text{ИОУ}} \approx 0. \quad (16.8)$$

В этом случае ток I_{OC} (без учета I_y) будет определяться только суммой двоично-взвешенных токов, которая в свою очередь определяется состояниями ключей Q. Поэтому можно представить $U_{\text{Вых}}$ в следующем виде:

$$U_{\text{Вых}} = I_{\text{OC}} R_{\text{OC}} = (I_0 Q_0 + 2I_0 Q_1 + 4I_0 Q_2 + 8I_0 Q_3) R_{\text{OC}} \quad (16.9)$$

Далее, учитывая формулу (16.5) можно получить зависимость выходного напряжения от входного кода

$$U_{\text{Вых}} = (1Q_0 + 2Q_1 + 4Q_2 + 8Q_3) I_0 R_{\text{OC}} = X_{\text{ВХ}} I_0 R_{\text{OC}}, \quad (16.10)$$

которую чаще используют в другом виде, записывая ток I_0 через закон Ома:

$$U_{\text{Вых}} = X_{\text{ВХ}} U_{\text{оп}} R_{\text{OC}} / R \quad (16.11)$$

Из полученной формулы видно, что рассмотренная схема выполняет функции ЦАП, причем выходное напряжение оказывается пропорциональным произведению входного кода $X_{\text{ВХ}}$ на опорное напряжение $U_{\text{оп}}$. Поэтому подобные ЦАП иногда называют *умножающими* и используют в комбинированных аналого-цифровых умножителях и вычислительных устройствах.

Например, если вместо опорного напряжения подать синусоиду, то можно будет управлять ее амплитудой на выходе, задавая нужный код. Подобные ЦАП могут работать в схемах регулировки громкости звука.

Следует обратить внимание на то, что *полярность* $U_{\text{Вых}}$ *обратна* полярности $U_{\text{оп}}$, так как используется инвертирующее включение ИОУ (см. п. 5.6.1).

Важно также отметить, что в микросхемах ЦАП полупроводниковые резисторы сильно изменяются с температурой, но на погрешности ЦАП это почти не сказывается. Дело в том, что выходной сигнал, как видно из формулы (16.10), пропорционален $R_{ос}$ и обратно пропорционален R . Поэтому температурный дрейф резисторов взаимно компенсируется.

Для увеличения разрядности необходимо добавить соответствующее число аналоговых ключей и резисторов, продолжающих матрицу. Однако при производстве многоразрядных ЦАП достаточно трудно изготовить на кристалле набор двоичновзвешенных резисторов с необходимой точностью, особенно очень больших и очень малых номиналов.

16.1.5. ЦАП на основе резисторной матрицы R - $2R$

Технология изготовления резисторов упрощается в случае применения матрицы типа R - $2R$ (рис. 16.4), в которой за счет *поразрядного* снижения опорного напряжения, можно использовать резисторы только двух значений R и $2R$.

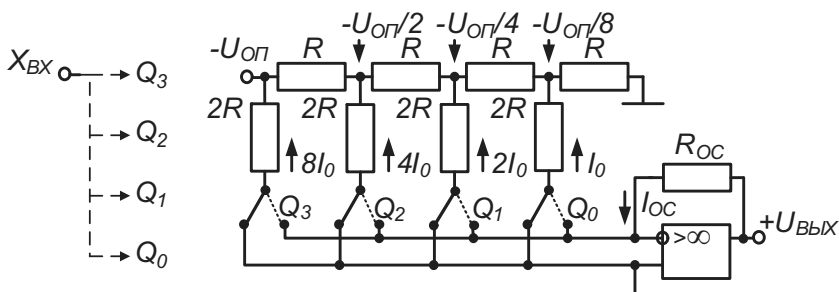


Рис. 16.4

В этой схеме, по своим свойствам полностью соответствующей вышерассмотренной, должны использоваться перекидные ключи, чтобы нагрузка в ветвях опорного напряжения не зависела от состояния ключей Q . При этом несколько усложняется схемотехника преобразователя, но для потребителей микросхем усложнение остается практически незаметным, а наоборот, появляются полезные качества. Например, при использовании перекидных ключей левые выводы (подсоединенные к общей точке) могут образовать дополнительный инверсный токовый выход.

§16.2. Микросхемы ЦАП и их интерфейсы

16.2.1. Обозначение ЦАП

рассмотрим на примере самой простой микросхемы K572ПА1 с выходом по току, схема включения которой показана на рис. 16.5. Выводы питания для простоты не показаны. Усилителя в составе данного ЦАП нет, поэтому

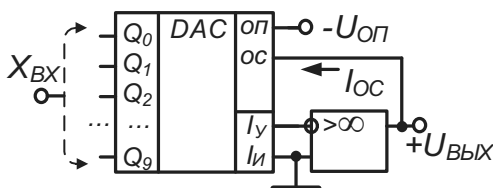


Рис. 16.5

нужно применять внешний ИОУ для преобразования выходного тока в напряжение. А в остальном эта микросхема по своей организации похожа на схему, показанную на рис. 16.4, только входной код 10-разрядный. Ключи выполнены на полевых транзисторах с изолированным затвором, матрица резисторов $R\text{-}2R$ и резистор R_{oc} – полупроводниковые.

Микросхема имеет два токовых выхода: прямой I_y и инверсный $I_{и}$. Входной двоичный сигнал отличается от ТТЛ уровня ($U^0 \leq 0,8 \text{ В}$, $U^1 \geq 3,6 \text{ В}$), поэтому лучше использовать управление от КМОПТЛ элементов или принять меры для увеличения ТТЛ сигнала.

Другие микросхемы ЦАП могут иметь внутри кристалла выходной усилитель и источник опорного напряжения. Иногда в одной микросхеме реализуется даже несколько отдельных ЦАП с общим питанием. Некоторые микросхемы ЦАП имеют регистр для хранения входного кода. Для записи кода в этот внутренний регистр могут использоваться параллельный или последовательный интерфейсы, особенности которых описаны далее.

16.2.2. Параллельный интерфейс ЦАП

Для записи входного кода во внутренний регистр используются дополнительные синхронизирующие выводы. К ним в первую очередь относятся E ($Enable$ – разрешать) и WR ($Write$ – записать). Такой набор сигналов позволяет в принципе использовать стандартное подключение ЦАП к общим микропроцессорным шинам (см. п. 14.5.3) – выводы для входного кода подключаются к шине данных, вывод WR к сигналу управления записью, а E – к шине адреса через дешифратор адреса для ЦАП.

Но для 8-разрядных процессоров такой способ не всегда применим, поскольку разрядность ЦАП в большинстве случаев больше. Поэтому микросхемы, ориентированные на байтовый формат, допускают поочередную загрузку младшего и старшего байта данных, для чего соответственно имеют два разрешающих сигнала *EL* (*Enable Low*) и *EH* (*Enable High*). Однако в этом случае аналоговый выходной сигнал некоторое время (в переходном режиме) не будет соответствовать входному коду. Для устранения этого недостатка в микросхему встраивают дополнительный полноразрядный регистр хранения данных, куда два ранее записанных байта копируются одновременно по дополнительному сигналу, обычно обозначаемому *LD* (от *Load* – загрузить).

Рассмотрим пример обслуживания такой микросхемы для случая программной реализации параллельного интерфейса со стороны ОВМ x51 (рис. 16.6). Порт *P1* слу-

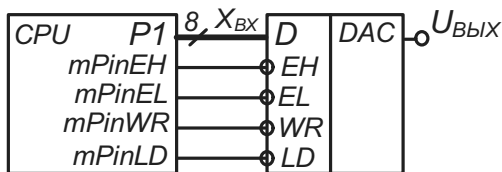


Рис. 16.6

жит для вывода байтов данных, а управляющие сигналы *WR*, *EL*, *EH* и *LD* обеспечиваются произвольными битами других портов.

Будем считать, что код данных $X_{вх}$ хранится в двух байтовых переменных внутреннего ОЗУ *mKodXH* и *mKodXL*. Программа должна последовательно записать эти байты (очередность не имеет значения), формируя инверсные управляющие сигналы *mPinEL*, *mPinWr*, *mPinEH* и снова *myPinWr*. Затем нужно копировать оба байта в основной регистр при помощи сигнала *mPinLD*.

```

Mov P1, mKodXL           ; Младший байт на линиях данных.
Clr mPinEL               ; Разрешение приема младшего байта.
Clr mPinWr               ; Копирование младшего байта.
SetB mPinWr              ; Возврат управляющих сигналов
SetB mPinEL              ; в исходное (пассивное) состояние.
Mov P1, mKodXH           ; Старший
Clr mPinEH               ; байт
Clr mPinWr               ; копируется
SetB mPinWr              ; аналогично
SetB mPinEH              ; в микросхему.
Clr mPinLD               ; Загрузка обоих байтов.
SetB mPinLD

```

Иногда для увеличения длительности синхроимпульсов (чтобы паразитные емкости интерфейса успели зарядиться или разря-

даться) между командами сброса/установки битов добавляют пустые команды для временной задержки.

Отметим также, что подобный ЦАП можно подключить к ОВМ, как две ячейки внешней памяти данных, например, в режиме 8-битной адресации (см. §14.7).

16.2.3. Последовательный интерфейс типа SPI

Учитывая, что ЦАП является сравнительно медленно действующим устройством, его микросхемы часто имеют последовательный интерфейс для занесения кода во внутренний регистр. Для преобразования последовательного входного кода в параллельную форму внутри ЦАП обычно используется регистр сдвига (см. §12.5) со своей схемой управления, зависящей от типа интерфейса, и регистр хранения для исключения изменения выходного сигнала на этапе интерфейсного обмена.

В простейшем варианте в микросхемах используется похожий на SPI (см. п. 12.8.1) синхронный интерфейс, в котором имеется две линии. Первая имеет обозначение *DI* (*Data Input*), является аналогом линии *MOSI* (*Master-Out Slave-In*) в интерфейсе SPI и служит для приема в ЦАП последовательных данных. Вторая линия служит для тактирования *CLK* и подобна *MCLK* (*Master CLock*). Кроме них имеются также обычные сигналы выбора-разрешения приема данных *EI* и копирования в основной регистр *LD*.

Схема подключения такого ЦАП приведена на рис. 16.7, а программа дана ниже. Последовательный интерфейс удобен тем, что схема подключения и алгоритм программы почти не зависят от разрядности используемого ЦАП:

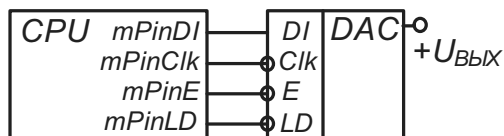


Рис. 16.7

```

Clr mPinE
Mov A, mKodXH ; Старший байт подготовлен для передачи.
Mov mTemp, # 8
mL1:
    RLC A ; Сдвиг бита данных в перенос.
    Mov mPinDI, C ; Передача одного бита.
    Clr mPinClk
    SetB mPinClk
    DJNZ mTemp, mL1 ; Все 8 битов переданы?
    Mov A, mKodXL ; Подготавливаем младший байт.
  
```

```

Mov myTemp, # 8
mL2:
    RLC A
    Mov mPinDI, C
    Clr mPinClk
    SetB mPinClk
    DJNZ mTemp, mL2
    Clr mPinLD
    SetB mPinLD
    SetB mPinE

```

; Загрузка обоих байтов

Время загрузки кода в ЦАП зависит в основном от времени выполнения цикла передачи данных, то есть, по сути, от частоты тактовых импульсов *CLK* и разрядности ЦАП. Это время должно быть меньше времени реакции реального объекта, на который работает ЦАП.

Таким образом, последовательный интерфейс экономично использует внешние выводы ОВМ, но имеет ограничения по быстродействию, так как программа не может выполняться мгновенно. Для повышения скорости передачи можно рекомендовать использование внутренних синхронных режимов и штатных интерфейсов типа SPI. Например, последовательный порт семейства x51 может работать в синхронном режиме (см. п. 8.8.3), а ОВМ типа AVR [31] имеют встроенный автомат для реализации аппаратного интерфейса SPI.

16.2.4. Последовательный интерфейс типа Up/Down

Быстродействие ЦАП с последовательного интерфейсом частично повышают, применяя счетный принцип передачи кода. Он заключается в том, что в качестве источника кода для ЦАП используется внутренний предварительно обнуленный реверсивный счетчик (см. п. 13.1.5), на который при инициализации поступает заданное число тактовых импульсов, равное входному коду. В дальнейшем изменение кода в обе стороны происходит достаточно быстро, без обнуления путем подачи корректирующих импульсов и изменения направления счета. Если нужно увеличить код, используют режим суммирования счетчика, если уменьшить, то режим вычитания.

Таким образом, посылая в счетчик определенное количество счетных импульсов, можно установить необходимое значение двоичного кода для преобразования. Заметим, что код в счетчике изменяется только на ± 1 , поэтому регистр хранения, присутствующий в других последовательных ЦАП, отсутствует, и нет необходимости в его загрузке.

Такой интерфейс называют интерфейсом *Up/Down* (вверх/вниз) или инкрементно-декрементным. Для его обслуживания (рис. 16.8) используются входы для счетных импульсов *CLK*, сброса *R* и направления счета *U/D* (*Up/Down*).

И, как и прежде, при увеличении разрядности ЦАП число обслуживающих линий не увеличивается, хотя нужно более внимательно контролировать достаточно большое время преобразования на начальном этапе установки кода.



Рис. 16.8

Программа, приведенная ниже, получает модуль приращения кода в переменной *mDeltaX*, знак приращения – в флажке *mFlagZnakDelta*. Команды обнуления внутреннего счетчика ЦАП для простоты не приведены.

```

    Clr mPinE
    Mov C, mFlagZnakDelta ; Устанавливаем направление счета.
    Mov mPinUpD, C
    mL3:
        Clr mPinClk ; Формируем импульсы по числу
        SetB mPinClk ; единиц в переменной mDeltaX.
        DJNZ mDeltaX, mL3
        SetB mPinE
  
```

16.2.5. Области использования ЦАП

Помимо того, что ЦАП работают в составе более сложных аналого-цифровых устройств, например АЦП, они имеют и самостоятельное использование. Они работают во всевозможных функциональных генераторах произвольной формы (см. п. 14.2.3), в модемах при формировании кодированной несущей, в цифровых телефонах, в синтезаторах звуковых сигналов (речь, музыка...) и др. Усилители с программируемым коэффициентом усиления (см. п. 6.2.3) также могут реализовываться на основе ЦАП.

Особой формой ЦАП являются не имеющие выходного усилителя *цифровые потенциометры*, которые реализованы на основе ступенчатого делителя из одинаковых резисторов общим числом $2^N - 1$, где N – разрядность схемы, и на единицу большего числа аналоговых ключей. Кроме того, функции входного регистра в цифровых потенциометрах выполняет энергонезависимая память, служащая для долговременного хранения установленного кода. Цифровые потенциометры мо-

гут иметь нелинейную характеристику и использоваться для замены механических потенциометров в цепях настройки аналоговых узлов, например, при программном управлении яркостью, контрастом, громкостью, полосой пропускания. Примером отечественного цифрового потенциометра на 256 уровней является микросхема 1272ПН1.

§16.3. Время-импульсный ЦАП

16.3.1. Общие сведения

В последнее время в микропроцессорных системах все чаще стал применяться время-импульсный ЦАП на основе широтно-импульсной модуляции (ШИМ), которую широко используют в мощных ключевых регуляторах или электроприводе. В режиме ШИМ (англ. сокращение *PWM* – от *Pulse Width Modulation*) процессор отсчитывает длительность (ширину) импульсов пропорционально некоторому коду при помощи внутреннего таймера или даже простейшей программы.

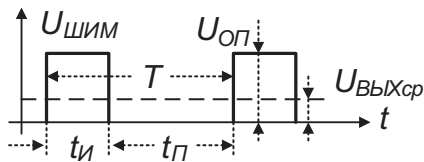


Рис. 16.9

Если такие импульсы (рис. 16.9) усреднить, то есть выделить постоянную составляющую импульсов $U_{ШИМ}$ и подавить высшие гармоники при помощи, например, простейшего *RC* фильтра (рис. 16.10), то выходное напряжение фильтра оказывается зависящим от амплитуды импульсов и временных соотношений:

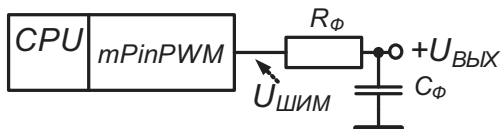


Рис. 16.10

$$U_{ВЫХср} = U_{ОП} \cdot t_{и} / T. \quad (16.12)$$

Если сделать длительность $t_{и}$ пропорциональной выводимому двоичному коду X , то выходной сигнал ($U_{ВЫХ} = U_{ВЫХср}$) будет определяться этим кодом, и тем самым будет осуществлено цифро-аналоговое преобразование.

Разрядность такого простейшего ЦАП можно программно изменять в широких пределах. И основную роль при этом имеет, как и прежде, максимальное значение преобразуемого кода $X_M = 2^N - 1$, зависящее от раз-

рядности ЦАП. Пусть изменения $t_{и}$ и T осуществляется дискретами t_0 , тогда в формуле (16.12) можно использовать $t_{и} = X_{вх} t_0$ и $T = X_M t_0$:

$$U_{вых} = U_{оп} X_{вх} t_0 / X_M t_0 = X_{вх} U_{оп} / X_M. \quad (16.13)$$

Отсюда следует замечательная особенность время-импульсного ЦАП – стабильность выходного сигнала зависит только от амплитуды $U_{оп}$. Если обеспечить стабилизацию амплитуды импульсов, то можно реализовать очень простой времяимпульсный ЦАП достаточно высокой точности и необходимой разрядности, не требующий набора двоично-взвешенных резисторов и их технологической подстройки.

Недостатком же подобного ЦАП является инерционность, вносимая сглаживающим фильтром. Поэтому времяимпульсный ЦАП используется в приложениях, не требующих высокого быстродействия.

Заметим, что в некоторых случаях вообще можно обойтись без сглаживающего фильтра. Например, для управления терморегулятором используют мощный ключ, подающий на нагреватель импульсы напряжения. Поскольку инерционность объекта управления значительно больше периода импульсов, то он не будет успевать реагировать на импульсы, выделяя среднее значение напряжения.

16.3.2. Формирование ШИМ сигнала

Поясним на примере использования внутреннего таймера ОВМ x51. Будем использовать механизм прерываний по переполнению таймера $T0$, причем программа прерывания будет вызываться дважды за период, при окончаниях импульса и паузы. В зависимости от состояния выходного бита ($mPinPWM$) в таймер будем загружать двухбайтовые коды формирования импульса $mKodIH$ и $mKodIL$ или паузы $mKodPL$ и $mKodPL$.

Поскольку состояние таймера инкрементируется, коды загрузки таймера на числовой оси определяются соответствующими кодами X и $X_{МАКС} - X$ со знаком минус (рис. 16.11). Вычисляют эти коды загрузки импульса и паузы обычно вне процедуры прерывания.

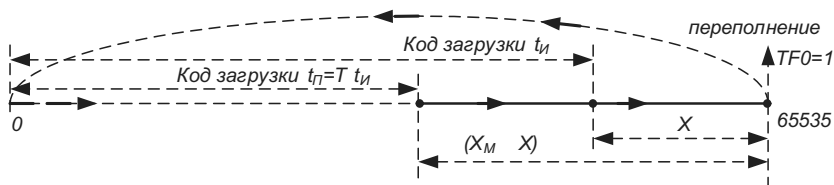


Рис. 16.11

Кроме того, в начале каждого периода будем производить в переменной *mCountT* счет периодов, как некоего «системного» времени. В остальном оформление процедуры прерывания по таймеру типовое (см. п. 8.7.5) – сохранение РОН, перезагрузка таймера и восстановление РОН.

Push PSW	; Сохранение признаков.
Clr TR0	; Останов таймера.
JNB mPinPWM, mTI	; Пауза или импульс?
Mov TH0, mKodPH	; Загрузка кода
Mov TL0, mKodPL	; на отсчет паузы.
Clr mPinPWM	; $U_{\text{ШИМ}} = 0$
Jmp mEx	
 <i>mTI:</i>	
Mov TH0, mKodIH	; Загрузка кода
Mov TL0, mKodIL	; на отсчет импульса.
SetB mPinPWM	; $U_{\text{ШИМ}} = 1$
Inc mCountT	; Счет периодов.
<i>mEx:</i>	
SetB TR0	; Пуск таймера.
Pop PSW	; Восстановление признаков.
Retl	

16.3.3. Аппаратное формирование ШИМ сигнала

Для упрощения формирования ШИМ сигнала таймеры некоторых ОВМ дополняют цифровым компаратором и несколькими вспомогательными регистрами, которые позволяют при отсчете длительностей импульса и паузы минимально отвлекать процессор от основной работы. Такие режимы таймерных устройств называют режимами компарирования (*Compare Modes*).

В подобных схемах (см. п. 13.2.5) таймер непрерывно работает в счетном режиме, перебирая свои состояния от 0 до X_m . Значение X_m определяет период ШИМ сигнала и обычно задается неявно при выборе разрядности ШИМ режима. Во вспомогательном регистре программа записывает код X_{BX} . В момент равенства кода таймера с кодом X_{BX} во вспомогательном регистре цифровой компаратор завершает импульс на внешнем выводе ОВМ (рис. 16.12). Нетрудно видеть, что чем больше код X_{BX} , тем больше длительность $t_{\text{и}}$.

Частота работы таймера, как правило, может программироваться путем изменения ко-

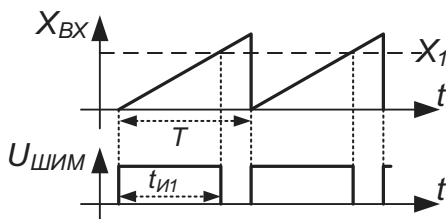


Рис. 16.12

эффициента деления предделителя (см. п. 13.2.1). Кроме того, цифровой компаратор в момент равенства может совершать не только сброс оговоренного выходного бита, но и его установку, инвертирование или, например, запуск внутреннего узла (АЦП или интерфейсного передатчика). Тип действия в «момент истины» программируется отдельными битами в спецрегистре режима таймера.

После инициализации ШИМ аппаратуры программа лишь по мере надобности заносит во вспомогательный регистр код импульса, то есть участие процессора в формировании сигнала минимально.

Наличие в ОВМ нескольких вспомогательных регистров позволяет при использовании одного таймера получить несколько ШИМ сигналов, имеющих общий период.

Кроме рассмотренного выше инкрементного способа генерации ШИМ, называемого несимметричным, в некоторых ОВМ используется изменение направления счета таймера при достижении максимального значения, что позволяет получить несколько центрированных или симметричных импульсов (рис. 16.13). Интересной особенностью отдельных ОВМ, ориентированных на управление мощными ключами, является генерация противофазных ШИМ сигналов с небольшой задержкой, чтобы устранить режим сквозных токов (см. п. 2.2.4).

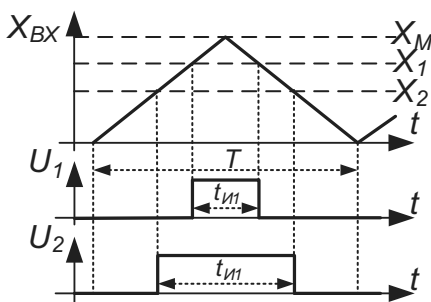


Рис. 16.13

16.3.4. Сглаживающий фильтр

ШИМ сигнал имеет, как следует из разложения в тригонометрический ряд Фурье, кроме информационной постоянной составляющей, дополнительные гармонические составляющие, которые проявляются в виде пульсаций выходного напряжения и которые нужно по возможности подавить. Учитывая, что высшие гармоники сглаживаются фильтром низкой частоты лучше, чем низшие, будем учитывать только первую гармонику, частота которой определяется периодом ШИМ сигнала, а амплитуда будет максимальна при $t_{и} = T/2$ [12]:

$$U_1 = 2 U_{оп} / \pi. \quad (16.14)$$

С другой стороны, напряжение пульсаций (первой гармоники) должно подавляться фильтром до величины, меньшей половины дискреты ЦАП:

$$U_{\text{Вых1}} = 0,5 U_0 = 0,5 U_{\text{оп}} / X_M. \quad (16.15)$$

Отсюда требуемый коэффициент сглаживания фильтра составит

$$K_{\text{сгл}} > U_1 / U_{\text{Вых1}} = 2 U_{\text{оп}} X_M / (\pi 0,5 U_{\text{оп}}) = 4 X_M / \pi. \quad (16.16)$$

Например, для 8-разрядного времяимпульсного ЦАП требуется $K_{\text{сгл}} > 325$.

В качестве примера будем использовать однозвенный RC фильтр (рис. 16.10), в котором для эффективного сглаживания должно выполняться неравенство $1/\omega_1 C_\phi \ll R_\phi$. А соотношение этих значений, по сути, и определяет коэффициент сглаживания однозвенного фильтра:

$$K_{\text{сгл}} = U_1 / U_{\text{Вых1}} = \omega_1 C_\phi R_\phi = 2\pi F_1 C_\phi R_\phi. \quad (16.17)$$

где $F_1 = 1/T$ – частота первой гармоники, определяемая периодом ШИМ.

Эта формула обычно используется при оценке величины емкости при заданных значениях F_1 и рассчитанных ранее $K_{\text{сгл}}$ и R_ϕ :

$$C_\phi = K_{\text{сгл}} / 2\pi F_1 R_\phi. \quad (16.18)$$

С другой стороны постоянная времени фильтра $C_\phi R_\phi$ определяет время установления, то есть быстродействие ЦАП. Для повышения быстродействия обычно увеличивают число звеньев RC фильтра или применяют активные фильтры на основе ИОУ [18].

Нужно отметить, что существуют и другие способы подавления пульсаций ШИМ сигнала. Например, во время-импульсных ЦАП, являющихся составной частью современных сигма-дельта АЦП (см. §18.6), дополнительно к интеграторам используют цифровые фильтры.

17 ПОРОГОВЫЕ УСТРОЙСТВА И ГЕНЕРАТОРЫ ИМПУЛЬСОВ

Для оценки внешней аналоговой информации процессор нуждается в дополнительных устройствах, преобразующих аналоговые сигналы с датчиков в цифру. К простейшим подобным устройствам с однобитовым цифровым выходным сигналом относятся пороговые устройства. Кроме того пороговые устройства являются основой для генераторов прямоугольных импульсов.

В зависимости от наличия гистерезиса в передаточной характеристике различают два вида пороговых устройств. Устройства без гистерезиса принято называть аналоговыми компараторами, а с гистерезисом – триггерами Шмитта.

§17.1. Аналоговые компараторы

17.1.1. Общие сведения

Компаратор служит для сравнения двух аналоговых сигналов, причем результат сравнения (больше или меньше) представляется в цифровом виде. Поэтому компаратор, занимает промежуточное положение между аналоговыми и цифровыми элементами. Некоторые ОВМ имеют внутренний компаратор (см. п. 7.2.3), что позволяет без внешних схем реализовать простейший однобитовый АЦП для грубой оценки аналогового сигнала (см. п. 18.4.2).

В качестве внешнего компаратора в принципе можно использовать ИОУ, приняв меры для приведения выходного сигнала к стандартному ТТЛ уровню (см. п. 2.5.5). Однако в этом случае трудно обеспечить высокое быстродействие ИОУ в режиме перегрузки. Дело в том, что компараторы принципиально работают с достаточно большими входными сигналами, находясь большую часть времени на нелинейных участках амплитудной характеристики и переключаясь из одного состояния в другое.

Поэтому в качестве компараторов применяют специально разработанные микросхемы, отличающиеся высоким быстродействием при перегрузке и стандартным уровнем выходного логического сигнала.

Входная часть компаратора подобно ИОУ выполнена на основе дифференциального каскада (см. п. 5.4.1) и имеет инвертирующий и не инвертирующий входы. Промежуточные каскады в компараторе обеспечивают в основном большой коэффициент усиления. Выходная цепь – цифровая, характеризуется стандартными логическими уровнями выходных сигналов и бывает однотранзисторной или двухтранзисторной.

Однотранзисторная выходная цепь компаратора с открытым коллектором (рис. 17.1), считается более универсальной по выходному напряжению, поскольку единичный уровень в ней можно задать напряжением, к которому должен подключаться внешний резистор. Следует отметить, что от величины этого резистора существенно зависит длительность положительного фронта, обусловленная зарядом паразитной емкости нагрузки (см. п. 1.2.1). Поэтому не рекомендуется выбирать величину сопротивления R больше 5..10 кОм.

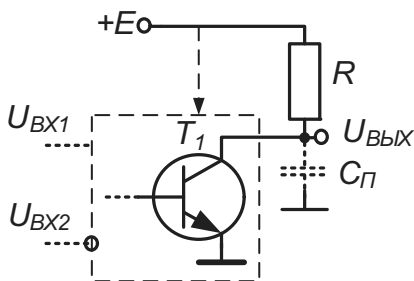


Рис. 17.1

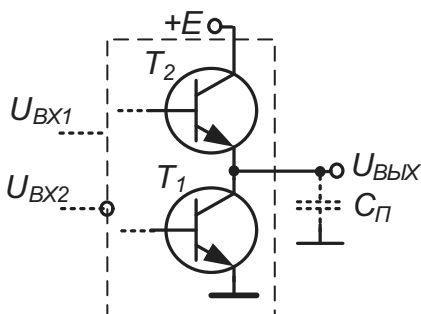


Рис. 17.2

Двухтранзисторная выходная цепь (рис. 17.2) является более быстродействующей (см. п. 2.2.2), потому что паразитная емкость нагрузки быстро заряжается и разряжается через малые сопротивления насыщенных транзисторов.

УГО компаратора похоже на обозначение ИОУ, инвертирующий вход так же помечен кружком, только в поле выполняемой

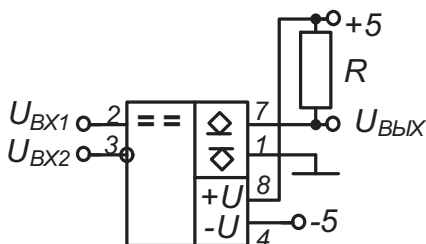


Рис. 17.3

функции указываются два знака равенства. Для примера на рис. 17.3 показана схема включения компаратора К554СА301, который имеет выходы с открытым коллектором (помечен ромбом с чертой внизу) и эмиттером (черта сверху). Если эмиттерный вывод «1» не используется (например, для стробирования), то его подключают к общей точке. Открытый коллектор, как обычно, нуждается во внешнем резисторе. Питание данного компаратора – двуполярное.

17.1.2. Основные параметры компаратора

Свойства компаратора определяются четырьмя основными типами параметров: входными, выходными, передаточными и временными.

Входные параметры такие же, как у интегрального усилителя (см. п. 5.4.3) – напряжение смещения $U_{см}$, входной ток $I_{вх}$, разностный ток I_p и др.

Выходная цепь имеет параметры подобно логическим элементам (см. п. 2.4.6) – U^0 , U^1 , $I_{вых}$ и др.

Передаточная характеристика компаратора, представляющая зависимость выходного сигнала от входного (рис. 17.4), использует в качестве входного разность напряжений между входами. На этой резко нелинейной характеристике можно выделить три участка, два пологих и крутой. На пологих участках $U_{вых}$ не изменяется и равно U^1 или U^0 . Поэтому основным параметром передаточной характеристики является коэффициент усиления, характеризующий наклон крутого участка, который определяется отношением приращений, выходного сигнала и входного:

$$K = \Delta U_{вых} / \Delta(U_{вх1} - U_{вх2}). \quad (17.1)$$

Отметим, что примерное значение коэффициента усиления в зависимости от типа микросхемы составляет величину 1000...300000.

Для точных применений важно также учитывать смещение крутого участка относительно нуля, характеризуемое входным параметром $U_{см}$, знак и точное значение которого для конкретной микросхемы неизвестно (см. п. 5.4.3). В справочниках, как правило, приводят только максимально возможные значения этого и других параметров.

Быстродействие компаратора характеризуют временем переключения, которое для большинства микросхем составляет величину 10..100 нс. У быстродействующих схем это время меньше указанного, но потребляемая мощность больше. У микромощных – наоборот.

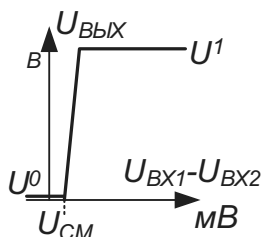


Рис. 17.4

При выборе типа компаратора обычно обращают внимание и на напряжение питания. Дело в том, что для питания аналоговой части компаратора необходимо двуполярное напряжение, которое не всегда есть в цифровых системах. Для образования отрицательного напряжения можно использовать инвертор полярности (см. п. 6.6.5) или применять однополярное включение компаратора, смирившись с ухудшением работы входного каскада (см. п. 6.6.4). Поэтому более перспективны микросхемы компараторов, допускающих работу от одного низковольтного источника питания.

Отметим также, что в одной микросхеме может быть размещено несколько компараторов.

17.1.3 Схемы сравнения

На практике используют две схемы сравнения входных сигналов, последовательную (сравнение напряжений) и параллельную (сравнение токов).

В *последовательной схеме* (рис. 17.5) сравниваются **однополярные** напряжения U_{BX1} и U_{BX2} , включенные *последовательно* друг с другом и разностным сигналом ΔU . Если напряжение U_{BX1} , поступающее на неинвертирующий вход, будет больше U_{BX2} , то, как видно из передаточной характеристики рис. 17.4, выходное напряжение будет единичным. Когда $U_{BX1} < U_{BX2}$, напряжение $U_{ВЫХ}$ будет равно нулю.

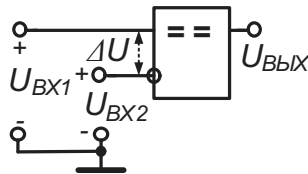


Рис. 17.5

Передаточную характеристику схемы сравнения обычно рисуют для случая, когда один из сравниваемых сигналов, например, U_{BX2} , считается неизменным, опорным. Поэтому передаточная характеристика для рассмотренной схемы имеет вид, приведенный на рис. 17.6. и ее называют прямой. Поменяв местами входы компаратора (изменив фазировку входов), можно получить инверсные значения выходного сигнала по отношению ко входному (рис. 17.7). Поэтому такую характеристику называют инверсной.

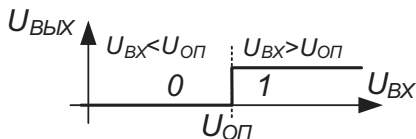


Рис. 17.6

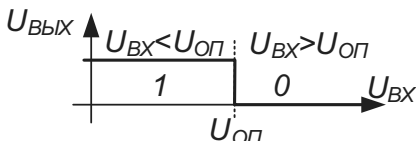


Рис. 17.7

Наклон крутого участка на таких иллюстративных характеристиках обычно не виден из-за большого масштаба по оси напряжения.

Параллельная схема (рис. 17.8) позволяет сравнивать напряжения по модулю (**разной полярности** относительно общей точки), причем условия сравнения записываются через значения токов. При $I_1 > I_2$ ($\Delta U > 0$) имеем $U_{Ввх} = 1$, если $I_1 < I_2$ ($\Delta U < 0$), то $U_{Ввх} = 0$. Учитывая, что в момент переключения $\Delta U = 0$, можно эти токовые условия при необходимости переписать для напряжений

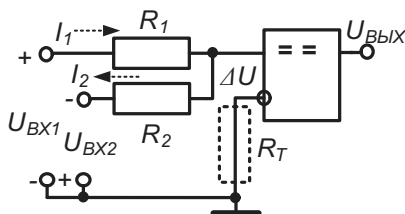


Рис. 17.8

$$U_{ВХ1} / R_1 = U_{ВХ2} / R_2, \quad (17.2)$$

то есть на условие сравнения влияют и величины резисторов. Передаточная характеристика параллельной схемы сравнения для абсолютных значений $U_{ВХ1}$ и $U_{ВХ2}$ также может быть прямой или инверсной в зависимости от фазировки входов компаратора.

Погрешности сравнения для обеих схем определяются в основном параметрами входной цепи $U_{см}$ и $I_{вх}$ (см. п. 5.7.4). Для параллельной схемы рекомендуется включать резистор $R_T = R_1 \parallel R_2$, уравнивающий пути протекания токов компаратора и уменьшающий их влияние на результат.

17.1.4 Двухпороговый компаратор

Это устройство (рис. 17.9) позволяет контролировать нахождение сигнала в зоне, заданной двумя пороговыми напряжениями, нижним $U_{п1}$ и верхним $U_{п2}$.

В схеме используются два компаратора, каждый из которых контролирует положение входного сигнала относительно одного из порогов, и выходная комбинационная схема, образующая логический результат сравнения. Заметим, что нижняя схема сравнения имеет прямую характеристику, а верхняя – инверсную (рис. 17.10).

Тогда для реализации общей характеристики в выходной цепи используется элемент логического умножения. Впрочем, типы характеристик могут быть и обратными.

Двухпороговый компаратор кроме самостоятельного применения (например, в виде микросхемы К521СА1) может использоваться и в составе более сложных схем, например, в прецизионном триггере Шмитта (см. п. 17.2.4).

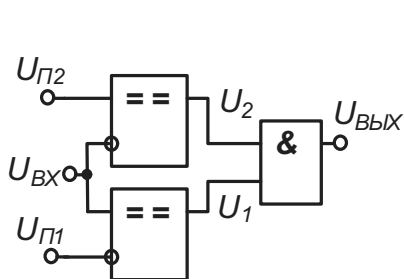


Рис. 17.9

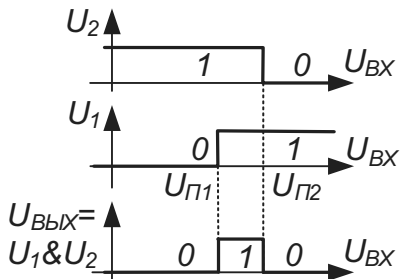


Рис. 17.10

17.1.5 Формирование прямоугольного сигнала

Во многих случаях информационные сигналы при прохождении через каналы передачи данных искажаются, затягиваются их фронты, и уменьшается амплитуда. Такой искаженный сигнал можно превратить в цифровой при помощи компаратора, например, с прямой характеристикой, что иллюстрируется на временных диаграммах рис. 17.11.

Обратим внимание на то, что крутизна выходного сигнала зависит от коэффициента усиления компаратора и иногда может быть недостаточной. Поэтому в таком случае необходимо выбирать микросхему с большим коэффициентом усиления. Но нужно помнить, что излишнее значение последнего может быть даже вредно, поскольку вместе с информацией усиливаются помехи и шумы, имеющиеся во входном сигнале. И тогда в выходном сигнале могут появиться ложные перепады (рис. 17.12), которые приведут к неправильной работе системы. Этого недостатка лишены пороговые устройства с гистерезисом, которые рассматриваются далее.

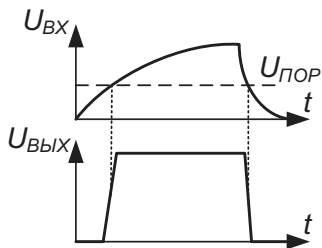


Рис. 17.11

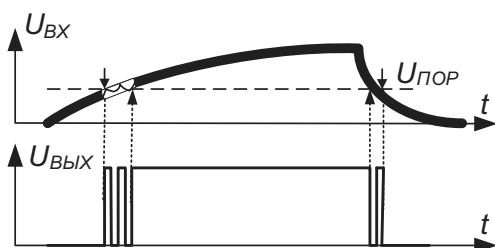


Рис. 17.12

§17.2. Пороговое устройство с гистерезисом (триггер Шмитта)

17.2.1. Общие сведения

У некоторых пороговых устройств передаточная выполняется характеристика раздвоенной. При уменьшении входного сигнала имеется один порог срабатывания схемы, а при увеличении – другой (рис. 17.13 и 17.14), то есть имеется два крутых участка и два порога при переключении. Если входной сигнал находится между порогами, то выходной сигнал сохраняет предыдущее значение. Такое устройство принято называть триггером Шмитта, по фамилии его изобретателя, а характеристику – гистерезисной.

Триггер Шмитта является аналого-цифровым устройством, имеющим свойства и компаратора, и триггера одновременно. Как цифровой триггер он имеет два устойчивых состояния с одним выходом и одним информационным входом. Но переключается триггер Шмитта не под действием стандартных логических уровней «0» или «1», как триггер, а реагирует, как компаратор, на величину входного напряжения. Причем, как уже отмечено, уровни включения и выключения триггера Шмитта различаются.

Характеристика триггера Шмитта, как и у компаратора, может быть *прямой* (рис. 17.13) или *инверсной* (рис. 17.14).

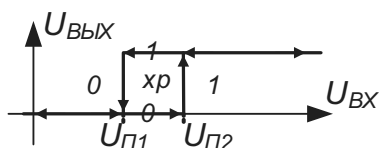


Рис. 17.13

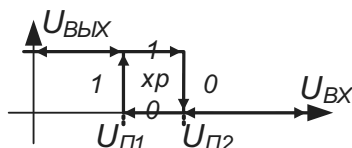


Рис. 17.14

Реализуется триггер Шмитта на основе усилительных схем и положительной обратной связи (ПОС), благодаря которой он, как и обычный триггер, поддерживает себя в одном из устойчивых состояний. Отметим также, что ПОС обеспечивает весьма большой коэффициент усиления на крутом участке передаточной характеристики. Последняя особенность оказывается полезной, если фронт входного сигнала затянута.

17.2.2 Формирование прямоугольного импульса

При обработке искаженного входного сигнала на выходе формируется прямоугольный импульс (см. рис. 17.15).

Переход схемы из одного состояния в другое происходит лавинообразно благодаря тому, что коэффициент усиления при ПОС стремится к бесконечности (см. п. 5.1.3). Поэтому независимо от длительности фронта входного сигнала фронт выходного сигнала будет очень крутым и будет определяться только частотными свойствами схемы.

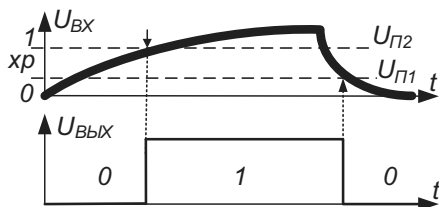


Рис. 17.15

Эта особенность триггера Шмитта весьма полезна при обработке сигналов, искаженных длинными линиями связи. Не менее важно и то, что благодаря наличию гистерезиса входные помехи не пройдут на выход, если величина их меньше разности порогов (величины гистерезиса).

Нетрудно видеть, что величина гистерезиса определяет помехоустойчивость, чем больше гистерезис, тем больше допустимая помеха.

Однако наличие гистерезиса приводит к некоторому сдвигу выходного сигнала по оси времени относительно входного, то есть к фазовым искажениям. Поэтому на практике стараются найти компромисс между помехоустойчивостью и вносимыми искажениями. Для этого некоторые входные устройства допускают регулировку (ручную или программную) порогов и величины гистерезиса.

На практике используют уже готовые элементы с гистерезисом. Например, все цифровые серии имеют в своем составе, логические элементы, предназначенные для работы с искаженным входным сигналом в условиях помех. Входные цепи некоторых ОВМ также имеют небольшой гистерезис.

17.2.3 Логический элемент с гистерезисом

имеет буквенный шифр ТЛ после номера серии, например, К155ТЛ1. Эта микросхема, УГО которой приведено на рис. 17.16, представляет собой две отдельные схемы логического умножения с инверсией и 4 входами. Заметим, что знак гистерезиса в обозначении дополнительной функции элемента иногда заменяется на буквы *ST* или *TH*.

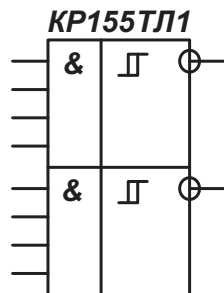


Рис. 17.16

В отличие от обычного логического элемента подобные схемы имеют внутреннюю ПОС, что приводит к возникновению гистерезиса при включении и выключении. Разница пороговых напряжений при переключении составляет величину примерно $0,8 \text{ В}$ ($U_{п1} \approx 0,8 \text{ В}$, $U_{п2} \approx 1,6 \text{ В}$).

Следует отметить, что у большинства цифровых триггеров Шмитта пороги точно не определены, и сильно зависят от температуры.

17.2.4 Прецизионный триггер Шмитта

Это устройство (рис. 17.17) имеет значения порогов, задаваемые внешними стабильными напряжениями, и реализуется на основе двухпорогового компаратора (см. п. 17.1.4).

Функции хранения и сравнения в прецизионном триггере Шмитта разделены. Хранение осуществляет цифровой RS -триггер, а сравнение производится с помощью двух компараторов. В подобной схеме можно реализовать различные передаточные характеристики в том числе и гистерезисные, как прямые, так и инверсные.

В качестве примера реализации *инверсной* гистерезисной характеристики на рис. 17.18 показаны передаточные характеристики отдельных компараторов (выходные напряжения которых являются входными для триггера с инверсными входами) и устройства в целом. Как и в двухпороговой схеме, отдельные компараторы реализуют прямую и инверсную характеристики. Поэтому активные уровни для входных сигналов триггера не пересекаются и есть зона, где оба компаратора обеспечивают пассивные сигналы. В этой гистерезисной зоне триггер хранит свое предыдущее состояние.

Изменяя порядок подключения входов триггера, можно получить прямую гистерезисную характеристику.

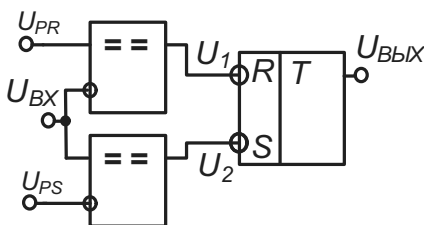


Рис. 17.17

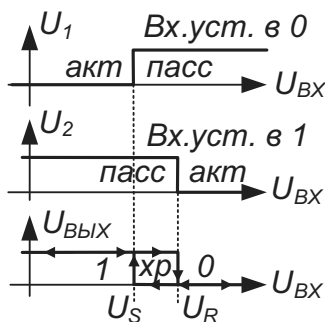


Рис. 17.18

Пороги прецизионного триггера Шмитта для определенности обозначают соответственно входам RS -триггера – U_{PS} и U_{PR} , то есть установки и сброса. И соответственно характеристике компаратора они могут быть прямыми или инверсными. Вход называют **прямым**, если он активен при U_{BX} большем порога. Если вход активен при напряжении U_{BX} меньшем порога, его называют **инверсным**.

§17.3. Микросхема таймера КР1006ВИ1

Микросхема таймера КР1006ВИ1 (ее зарубежный аналог – микросхема NE555 или просто 555) предназначена для генерации низкочастотных прямоугольных импульсов необходимой длительности.

17.3.1. Общие сведения

Основой микросхемы таймера является прецизионный триггер Шмитта, с инверсной передаточной характеристикой (см. п. 17.2.4), который для простоты пояснений можно представить в виде асинхронного D -триггера (см. рис. 17.19а) с инверсным аналоговым D_A входом (см. п. 12.2.1). То есть при нулевом уровне сигнала D_A триггер устанавливается в «1», а при единичном – в «0».

Но всегда нужно помнить, что у цифрового триггера порог различения пассивного уровня и активного один (например, у ТТЛ схем $\approx 1,5$ В, см. п. 2.4.5), а у триггера Шмитта – два.

Триггер D -типа, как известно (см. п. 12.2.1), может быть реализован на основе RS -триггера, поэтому для большей универсальности в микросхеме таймера аналоговый вход D_A разделен на два – вход для установки единицы S_A и вход для установки нуля R_A (см. рис. 17.19б). Причем отметим, что, как и в прецизионном триггере Шмитта, эти входы являются аналоговыми и имеют разные пороги U_{PS} и U_{PR} . Причем вход R_A является прямым, а S_A – инверсным. Эти входы при необходимости можно объединить и получить триггер Шмитта с одним входом D_A .

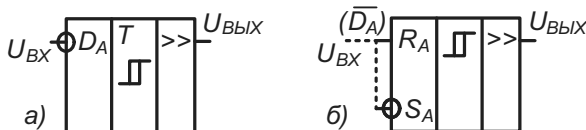


Рис. 17.19

На рис. 17.20 показаны передаточные характеристики для трех случаев, отдельно по входам S_A и R_A , а также для объединенных входов ($U_{BX} = U_{SA} = U_{RA}$). Важно выделить и помнить про два состояния для входных сигналов, активное и пассивное. При активном происходит установка или сброс триггера, а при пассивном – хранение, обеспечивающее гистерезисную характеристику.

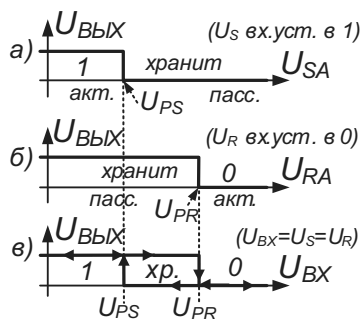


Рис. 17.20

17.3.2. Структура микросхемы таймера

приведена на рис. 17.21 и помимо прецизионного триггера Шмитта (в виде двух аналоговых компараторов и RS -триггера с дополнительным цифровым входом сброса R_Y) имеет резистивный делитель и два выходных инвертора, которые имеют разные типы выходных каскадов.

Основной инвертор выполнен по двухтранзисторной схеме (см. п. 2.2.2) и служит для образования выходного сигнала U_{BbIX1} .

Дополнительный инвертор с напряжением U_{BbIX2} представляет транзистор с открытым коллектором, который обычно используется для разряда внешней времязадающей емкости и поэтому этот выход часто называют «разрядным».

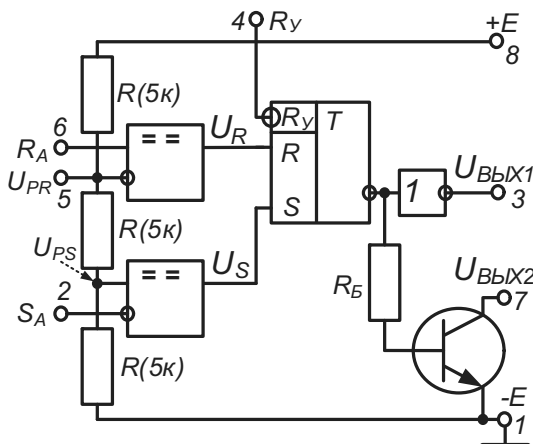


Рис. 17.21

Делитель из трех одинаковых резисторов служит для получения пороговых напряжений U_{PS} и U_{PR} , величина которых определяется напряжением питания

$$U_{PR} = I(R+R) = E \cdot 2R/(R+R+R) = E/3 \quad (17.3)$$

$$U_{PS} = I R = E R/(R+R+R) = E/3 \quad (17.4)$$

В некоторых случаях (см. п. 17.3.4) удобно представлять порог установки единицы U_{PS} как половину U_{PR} :

$$U_{PS} = U_{PR} / 2 \quad (17.5)$$

Следует отметить, что резисторы делителя твердотельные (полупроводниковые), поэтому они имеют существенный технологический разброс и большой температурный дрейф. Однако все три резистора имеют на кристалле одинаковые геометрические размеры, что позволяет считать их сопротивления (и температурный дрейф) практически одинаковыми. Поэтому, как следует из выражений (17.3) и (17.4), сопротивления R сокращаются и не входят в конечное выражение. Отсюда следует вывод, что пороговые напряжения не зависят от температуры и их стабильность определяются только напряжением питания, которое при необходимости можно заставить стабилизировать, но в стандартной схеме включения таймера этого не требуется, поскольку временные параметры не зависят от напряжения питания (см. п.п. 17.4.2 и 17.5.2).

Пороговое напряжение U_{PR} имеет внешний вывод, что позволяет при необходимости подключить к нему внешний фильтрующий конденсатор, чтобы уменьшить пульсации. Кроме того, используя этот вывод можно изменять величину пороговых напряжений (см. п. 17.3.5).

17.3.3. Обозначение таймера

Внутреннее устройство таймера на принципиальных схемах не раскрывают, а применяют УГО, приведенное на рис. 17.22.

Сразу следует отметить, что УГО, используемое в большинстве литературных источников, не соответствует отечественному стандарту [4] и, главное, имеет мнемонически не оправданные наименования внешних выводов. Поэтому зачастую на практике предпочитают вместо имен использовать номера выводов, даже в справочной [14] и учебной литературе [5], что существенно затрудняет понимание логики работы таймерных устройств.

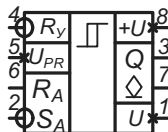


Рис. 17.22

Стандартное обозначение таймера на основе RS -триггера лишено указанного недостатка. Например, аналоговый вход R_A , как следует из обозначения, является прямым. Поэтому легко догадаться, что высокий уровень сигнала на этом входе (большой U_{PR}) сбросит таймер в ноль. Аналоговый вход S_A является инверсным, то есть при низком его уровне (меньше U_{PS}) таймер установится в единичное состояние. Цифровой установочный вход R_y также имеет нулевой активный уровень.

Второй выход с открытым коллектором обозначен стандартным знаком $\diamond$ с чертой внизу (см. п. 2.3.6), основной выход Q иногда оставляют без обозначения.

Выводы не несущие информацию (питания, порог U_{PR}) обычно помечают крестом.

17.3.4. Основные параметры таймера КР1006ВИ1

В связи с тем, что таймер является аналого-цифровой микросхемой, напряжение питания может изменяться в достаточно широких пределах от 5 до 16,5 В [14], что увеличивает универсальность устройства.

Соответственно изменение входных сигналов допускается от 0 до напряжения питания, причем величина входных токов зависит от типа входа. Так аналоговые входы S_A и R_A являются входами аналоговых компараторов на биполярных транзисторах и имеют подобно биполярным ИОУ (см. п. 5.4.3) значение I_{BX} менее 0,5 мкА.

Пороговые напряжения аналоговых входов U_{PS} и U_{PR} , как уже неоднократно отмечалось, различные и в соответствии с уравнениями (17.3) и (17.4) равны трети и двум третям от напряжения питания.

Цифровой вход R_y подобно ТТЛ элементу имеет несимметричную входную характеристику (см. п. 2.4.2) и соответственно сравнительно большой входной ток ($\approx 1,5$ мА) при активном нулевом сигнале и малый – при единичном.

Выходная цепь является цифровой и характеризуется логическими уровнями нуля и единицы, которые существенно зависят от тока нагрузки и температуры. Например, при токе 10 мА напряжение $U^0 \approx 0,1$ В, а единичный уровень $U^1 \approx E - 1,5$ В.

Полезным параметром при некоторых оценках может оказаться выходное сопротивление основного выхода таймера $R_{вых} \approx 10$ Ом.

Максимальные токи для выходов таймера зависят от типа микросхемы, например, для основного выхода этот ток может достигать нескольких десятков мА, а для выхода с открытым коллектором – 200 мА.

Таймер является сравнительно низкочастотным устройством, поскольку временные параметры имеют величину около долей *мкс*, поэтому частотный диапазон генерируемых импульсов для стандартных вариантов микросхем не превышает нескольких *МГц*.

17.3.5. Применение таймера в качестве порогового устройства

Прецизионный триггер Шмитта, лежащий в основе таймера, является простым и удобным пороговым устройством для преобразования уровня сигнала, отягощенного помехой, чему способствует универсальное питание таймера и возможность некоторого изменения величины порогов.

Для примера рассмотрим схему, приведенную на рис. 17.23 и предназначенную для преобразования искаженного информационного сигнала в прямоугольный ТТЛ сигнал. Стандартные пороги в этой схеме определяются напряжением питания (две трети и одна треть) и при необходимости могут быть уменьшены при помощи внешнего резистора R^* , который увеличивает коэффициент деления внутреннего делителя (см. рис. 17.21).

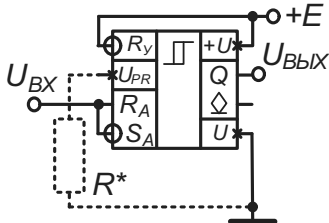


Рис. 17.23

Температурная стабильность порогов при этом ухудшается из-за влияния внутренних полупроводниковых резисторов, нестабильности которых ранее взаимно компенсировались. Для стабилизации порогов можно использовать стабилитрон или микросхему стабилизатора с напряжением $U_{CT} < 2E/3$ вместо внешнего резистора или заменить таймер прецизионным триггером Шмитта в первоначальном виде (см. п. 17.2.4).

Но чаще всего микросхема таймера используется для реализации разнообразных генераторов прямоугольных импульсов.

§17.4. Генератор одиночного импульса на таймере

Такие генераторы, которые в ответ на короткий запускающий импульс вырабатывают одиночный импульс заданной длительности, иногда называют одновибраторами или ждущими генераторами.

17.4.1. Схема и временные диаграммы

Помимо таймера в схеме генератора (рис. 17.24) имеется внешняя времязадающая цепочка в виде резистора R и конденсатора C . Таймер в режиме одновибратора использует отдельные входы для установки и сброса. Так вход S_A служит для запуска, то есть для переброса таймера из исходного нулевого состояния в единичное. А вход R_A , контролирующий напряжение на времязадающем конденсаторе, служит для возврата таймера через некоторое время в исходное нулевое состояние.

Заметим также, что выход с открытым коллектором подключен к конденсатору и служит для его разряда.

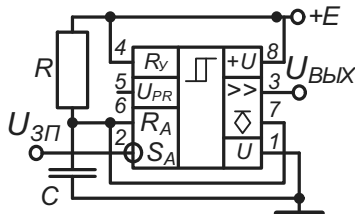


Рис. 17.24

В исходном нулевом состоянии оба входных сигнала таймера пассивны, так как напряжение $U_{3П}$ больше значения U_{PS} . А внутренний транзистор вспомогательного выхода в нулевом состоянии проводит и соединяет вход R_A и конденсатор с общим проводом. Другими словами сигнал на прямом входе установки в ноль также пассивен. В исходном состоянии схема может ожидать внешнего сигнала сколь угодно долго.

При поступлении внешнего нулевого импульса $U_{3П}$ (рис. 17.25) таймер устанавливается по входу S_A в единицу. Внутренний транзистор при этом переходит в непроводящее состояние (снимает «закоротку» с конденсатора) и начинается заряд конденсатора через резистор R .

Активный уровень внешнего сигнала запуска действует короткое время, после чего напряжение на входе S_A становится снова пассивным. Таймер будет хранить ранее установленное единичное состояние, поскольку напряжение на емкости мгновенно измениться не может и сигнал на входе R_A также будет пассивным.

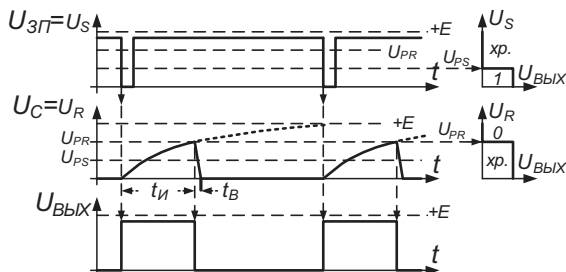


Рис. 17.25

Напряжение U_c начинает возрастать по экспоненте с постоянной времени RC , стремясь достигнуть напряжения питания E . Пока напряжение на конденсаторе остается меньше порога U_{PR} , сигнал на входе R_A остается пассивным. И только тогда, когда напряжение U_c сравняется с указанным порогом, таймер по входу R_A будет сброшен в исходное нулевое состояние.

Таким образом, в ответ на короткий запускающий импульс схема выработает одиночный импульс, длительность $t_{и}$ которого будет зависеть от напряжения на времязадающем конденсаторе.

Однако новый запускающий импульс сразу подавать нельзя, так как нужно дождаться полного разряда конденсатора через внутренний транзистор вспомогательного выхода таймера. Это время разряда обычно называют временем восстановления исходного состояния t_B . Вместе со значением $t_{и}$ оно определяет минимально возможный период и соответственно максимальную частоту запуска:

$$F_{3П\max} = 1/T_{3П\min} = 1/(t_{и} + t_B) \quad (17.6)$$

Поскольку разряд осуществляется через малое внутреннее сопротивление насыщенного транзистора обычно считают $t_B \ll t_{и}$. Но для определенности можно принять, например, $t_B \approx 0,1t_{и}$.

17.4.2. Определение длительности импульса

Поскольку длительность импульса определяется процессом заряда конденсатора, для расчетов необходим закон изменения напряжения на емкости в виде решения дифференциального уравнения по аналогии с уравнением (1.44):

$$U_c(t_{и}) = U_c(\infty) - [U_c(\infty) - U_c(0)] \exp(-t_{и} / RC) \quad (17.7)$$

Подставив в это уравнение граничные условия, которые нетрудно определить из временных диаграмм:

$$U_c(0) = 0, \quad U_c(\infty) = E, \quad U_c(t_{и}) = 2E / 3 \quad (17.8)$$

можно после простейших преобразований записать

$$\exp(-t_{и} / RC) = 1/3 \quad (17.9)$$

Логарифмируя это выражение, можно получить расчетное соотношение для длительности импульса, *не зависящее от напряжения питания*

$$t_{и} = RC \ln 3 \approx 1,1RC \quad (17.10)$$

17.4.3. Порядок расчета одновибратора

Длительность импульса $t_{\text{и}}$ бывает задана и требуется найти значения R и C . Однако для двух неизвестных одного уравнения (17.10) недостаточно и нужно привлекать еще одно условие работоспособности. Например, для того, чтобы таймер возможно меньше влиял на процесс заряда необходимо, чтобы ток через резистор R был много больше входного тока $I_{\text{вх}}$ и остаточного тока непроводящего внутреннего транзистора $I_{\text{ко}}$:

$$I_R = (E - U_{\text{с}}) / R \gg I_{\text{вх}} + I_{\text{ко}} \quad (17.11)$$

Это соотношение не имеет четких границ, потому что с одной стороны ток I_R меняется со временем, а с другой токи $I_{\text{вх}}$ и $I_{\text{ко}}$ для конкретной микросхемы в точности неизвестны. Поэтому на практике выражение (17.11) можно преобразовать в приблизительное, но более конкретное неравенство

$$R < E / 20 I_{\text{вх}} \quad (17.12)$$

Например, считая, что $E = 5 \text{ В}$, а $I_{\text{вх}} = 0,5 \text{ мкА}$, можно оценить максимальное сопротивление времязадающего резистора $R = 5 / 20 \cdot 0,5 \cdot 10^{-6} = 500 \text{ кОм}$. Для большей стабильности выбираем $R = 200 \text{ кОм}$ и далее из (17.10) находим, например, при $t_{\text{и}} = 1 \text{ мс}$, значение емкости $C = 10^{-3} / 1,1 \cdot 200 \cdot 10^3 \approx 4500 \text{ пФ}$ и выбираем ближайшее номинальное значение ее из стандартного ряда [5].

Ждущие генераторы используются в составе более сложных устройств, например, для формирования импульсов в частотных АЦП (см. п. 18.6.2) или для расширения коротких импульсов от датчиков. К примеру, сигнал от датчика радиоактивности имеет малую длительность, и процессор может не среагировать на него.

§17.5. Автоколебательные генераторы импульсов на таймере

17.5.1. Генератор несимметричных импульсов

Схема генератора в своей основе (рис. 17.26) похожа на схему одновибратора. Вход R_A по-прежнему контролирует напряжение на времязадающем конденсаторе, разряд конденсатора обеспечивается внутренним транзистором вспомогательного выхода с открытым коллектором. Только вход S_A , на который раньше подавался запускающий импульс, подключен теперь к конденсатору.

В случае, если напряжение питания имеет заметные пульсации, то к выводу U_{PR} обычно подключают фильтрующий конденсатор.

При таком включении таймер работает в режиме *D*-триггера (см. п. 17.3.1) и состояние обоих входов зависит от напряжения на конденсаторе. Поэтому таймер может устанавливаться и сбрасываться без внешнего запуска, то есть работать в автоколебательном режиме.

Предположим, что на схему подали питание. Напряжение на разряженном конденсаторе мгновенно измениться не может и поэтому некоторое время остается нулевым (рис. 17.27). Это означает, сигнал на входе S_A будет активным, а на входе R_A – пассивным, поэтому таймер сразу установится в единичное состояние.

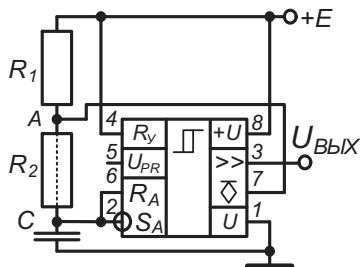


Рис. 17.26

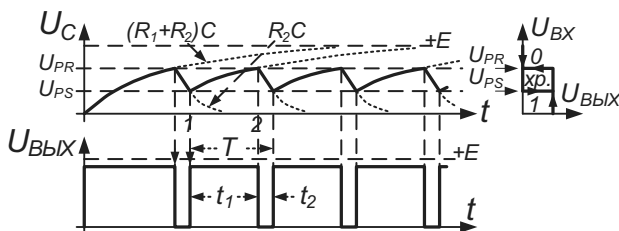


Рис. 17.27

Для наглядности на рис. 17.27 справа от диаграммы U_C показана в ортогональном виде передаточная характеристика триггера Шмитта (см. п. 17.3.1), по которой легко определить состояние выходного напряжения таймера в зависимости от значения $U_{BbIX} = U_C$.

При $U_{BbIX} = 1$ внутренний транзистор вспомогательного выхода не будет проводить и мешать заряду конденсатора. Ток заряда будет протекать через оба внешних резистора, поэтому напряжение на конденсаторе U_C будет увеличиваться с постоянной времени $C(R_1+R_2)$, стремясь к напряжению питания E . Но как только напряжение U_C достигнет порога активности прямого входа R_A таймер сбросится в нулевое состояние, то есть внутренний транзистор перейдет в проводящее состояние и подключит точку «А» к общему проводу.

При этом начнется разряд конденсатора через сопротивление R_2 , и напряжение U_C будет уменьшаться с постоянной времени CR_2 ,

стремясь к нулю. Как только это напряжение достигнет порога срабатывания по инверсному входу S_A , таймер снова установится в единичное состояние. Внутренний транзистор вспомогательного выхода перейдет в непроводящее состояние и начнется новый цикл заряда конденсатора.

Таким образом, таймер будет все время переключаться из одного состояния в другое, автоматически генерируя импульсы, длительность и период которых определяются процессами заряда и разряда времязадающего конденсатора.

17.5.2. Определение частоты несимметричных импульсов

Из временных диаграмм нетрудно видеть, что период генерируемых импульсов состоит из двух временных интервалов. Один из них t_1 определяется процессом заряда конденсатора, а другой t_2 – разряда.

Для нахождения величины t_1 воспользуемся уравнением (17.7), учитывая, что для данного случая (начало координат в точке «1») постоянная времени заряда равна $C(R_1 + R_2)$, а граничные условия нетрудно найти из рис. 17.27:

$$U_C(0) = E/3, \quad U_C(\infty) = E, \quad U_C(t_1) = 2E/3.$$

Решая получившееся уравнение, можно найти

$$t_1 = (R_1 + R_2) \cdot C \cdot \ln 2 \approx 0,7C \cdot (R_1 + R_2). \quad (17.13)$$

Аналогично, помещая начало координат в точку «2» и учитывая для процесса разряда постоянную времени CR_2 и граничные условия

$$U_C(0) = 2E/3, \quad U_C(\infty) = 0, \quad U_C(t_1) = E/3,$$

можно найти

$$t_2 = R_2 \cdot C \cdot \ln 2 \approx 0,7CR_2. \quad (17.14)$$

В итоге получаются следующие расчетные соотношения для генератора несимметричных импульсов

$$T = t_1 + t_2 = 0,7C \cdot (R_1 + 2R_2) \quad (17.15)$$

$$F = 1/T = 1/0,7C \cdot (R_1 + 2R_2) \quad (17.16)$$

Очень важной особенностью полученных формул является независимость длительностей и частоты от величины напряжения питания, что весьма ценится на практике.

17.5.3. Порядок расчета генератора несимметричных импульсов

Основным заданным параметром генератора является частота или период импульсов, поэтому при расчете используют одно из уравнений, (17.15) или (17.16), имеющие три неизвестных величины.

Как и при расчете одновибратора, сопротивление R_1 ограничено сверху (см. п. 17.4.3) и его значение обычно выбирают не более 200..500 кОм. Сопротивление R_2 рассчитывают, исходя из заданной длительности t_2 , решая совместно уравнения (17.13) и (17.14) при известном R_1 .

В некоторых случаях, когда $t_2 \ll t_1$ и соответственно $R_2 \ll R_1$, появляется возможность упрощения схемы, за счет выбора $R_2 = 0$. Но при этом частота генерации станет меньшей расчетной на 5..10%. Дело в том, что при отсутствии резистора R_2 внутренний транзистор будет очень быстро разряжать конденсатор, а аналоговый компаратор не будет успевать переключать таймер в единичное состояние. Поэтому напряжение на конденсаторе при разряде «проваливается» ниже уровня U_{PS} , что приводит к уменьшению начального значения $U_C(0)$ и к увеличению периода по сравнению с расчетным значением. Чтобы таймер успевал переключаться, надо замедлить разряд конденсатора, выбирая $R_2 \approx 50..100$ Ом.

17.5.4. Генератор симметричных импульсов

использует прежний принцип генерации, когда в процессе заряда и разряда напряжение на конденсаторе, определяющее входной сигнал таймера, заставляет его переключаться.

Только в отличие от несимметричной схемы управляет зарядом/разрядом не вспомогательный, а основной выход таймера (рис. 17.28). Это приводит к тому, что постоянные времени заряда и разряда становятся одинаковыми и равными RC , а также изменяются уровни, к которым стремится напряжение на конденсаторе.

Рассмотрим временную диаграмму для U_C , показанную на рис. 17.29, справа от которой для наглядности приведена передаточная характеристика таймера. Нетрудно видеть, что при включении питания напряжение на разряженном конденсаторе установит таймер в еди-

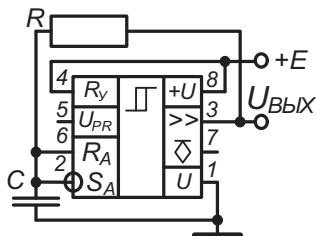


Рис. 17.28

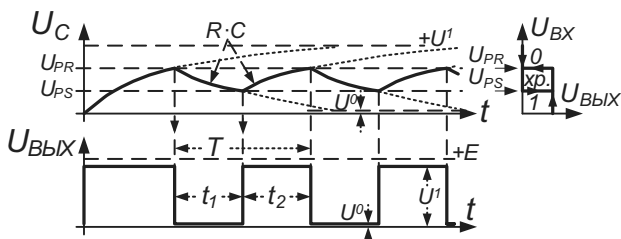


Рис. 17.29

ничное состояние. То есть на правом выводе внешнего резистора будет действовать напряжение U' , меньшее напряжения питания E (см. п. 17.3.4), и начнется заряд конденсатора. Когда напряжение U_C достигнет порога срабатывания U_{PR} для входа R_A , таймер сбросится в нулевое состояние и напряжение на его выходе станет равным нулевому уровню U^0 . Поэтому начнется разряд конденсатора, который закончится, когда напряжение U_C станет равным U_{PS} и вернет таймер по входу S_A в единичное состояние.

То есть таймер будет циклически переключаться, генерируя импульсы, длительность которых (благодаря одинаковости постоянных времени заряда/разряда) примерно равна половине периода. Такие импульсы иногда называют симметричными.

17.5.5. Определение частоты симметричных импульсов

Расчетную формулу для частоты колебаний симметричного генератора, можно получить, используя, как и ранее (см. п. 17.4.2), решение уравнения (17.7). Но граничные значения $U_C(\infty)$ и $U_C(0)$ для заряда/разряда в этом случае будут равны U' и U^0 . Поэтому в результирующих соотношениях не произойдет взаимного сокращения напряжения питания E , как в предыдущих случаях.

Это приводит к тому, что частота генератора становится зависящей от нестабильных выходных параметров таймера, которые имеют технологический разброс и существенный температурный дрейф. Кроме того при изменении напряжения питания частота также может изменяться. Поэтому в отличие от генератора несимметричных импульсов данную схему применяют там, где не требуется большая точность.

Поскольку выходные параметры таймера в точности не известны, для упрощения расчетов полагают, что $U' \approx E$, а $U^0 \approx 0$. В этом случае

нетрудно получить приближенные выражения, аналогичные уравнениям (17.15) и (17.16)

$$t_1 = t_2 \approx RC \cdot \ln 2 \approx 0,7RC \quad (17.17)$$

$$F = 1 / (t_1 + t_2) = 1 / 1,4RC \quad (17.18)$$

Последовательность расчета генератора симметричных импульсов остается типовой (см. п. 17.5.3) – выбирают R менее 500 кОм и, исходя из заданной частоты или периода, вычисляют необходимую емкость.

17.5.6. Генератор на основе логического инвертора с гистерезисом

В тех случаях, когда требуется сравнительно высокочастотный генератор импульсов, а скоростной таймер подобрать не удастся, используют логический инвертор с гистерезисом, который имеет лучшие, чем таймер, частотные свойства.

Схема, работа и временные диаграммы такого генератора (рис. 17.30) подобны рассмотренным в симметричном генераторе на таймере (см. п. 17.5.4). Отличие заключается в других порогах срабатывания (см. п. 17.2.3) и других величинах U^1 или U^0 .

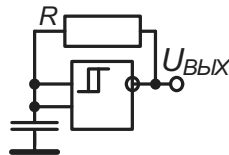


Рис. 17.30

Частоту генерации обычно определяют по приблизительной формуле (17.18), потому что пороги и уровни напряжений все равно в точности неизвестны и вдобавок нестабильны. Вследствие этого на практике приходится мириться не только с необходимостью подстройки частоты, но и ее сильной зависимостью от температуры. Поэтому лучшей альтернативой подобному генератору является высокочастотный кварцевый генератор (например, см. п. 6.8.2).

Другим недостатком генератора на ТТЛ инверторе является очень малая допустимая величина внешнего резистора. Дело в том, что входная характеристика ТТЛ элемента резко несимметричная и характеризуется большим током при нулевом напряжении (см. п. 2.4.2). Для обеспечения этого тока и уверенного переключения между порогами необходимо выбирать сопротивление R не более 400 Ом. Заметим, что для триггеров Шмитта на полевых транзисторах КМОП внешнее сопротивление можно выбрать на два-три порядка больше. Величину емкости при выбранном сопротивлении рассчитывают по формуле (17.18).

17.5.7. Частотные преобразователи для датчиков

Цифровая природа ОВМ хорошо согласуется с времяимпульсным и частотным представлением информации, поскольку длительность импульса или частоту легко измерить без внешних схем (см. п.п. 8.5.3, 10.4.5 и §10.7). Поэтому помимо основной функции (тактирования, задания временных интервалов) таймер часто применяют в качестве частотного преобразователя для простейших датчиков. Это позволяет в ряде случаев не использовать достаточно сложные и дорогие измерительные преобразователи и АЦП.

Например, включая вместо внешнего резистора датчик сопротивления R_x , обратносмещенный фотодиод в качестве датчика освещенности или датчик угла в виде переменного резистора, можно получить простой преобразователь параметрического сигнала в частоту, период или длительность импульса. Аналогично можно использовать вместо внешнего конденсатора датчик с изменяющейся емкостью C_x .

18 АНАЛОГО-ЦИФРОВЫЕ ПРЕОБРАЗОВАТЕЛИ

Для точной оцифровки аналоговой информации используют АЦП, которые условно делят на несколько групп, каждая из которых имеет свои достоинства и недостатки. Параллельные АЦП – самые быстрые, на основе ЦАП и компаратора – самые массовые, двойного интегрирования – достаточно точные и медленные, частотные и дельта АЦП – наиболее точные и современные, встроенные в ОВМ – не требующие внешних микросхем. Особенности построения указанных АЦП рассмотрены в следующих параграфах.

§18.1. Общие сведения и параллельный АЦП

18.1.1. Основные параметры АЦП

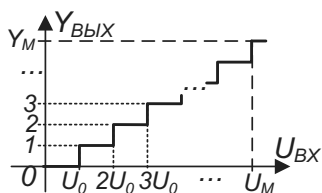


Рис. 18.1

Передаточная характеристика АЦП в соответствии с дискретной природой выходного сигнала имеет ступенчатый вид (рис. 18.1). Как и для ЦАП, пределы его характеристики ограничиваются

$$U_M = U_{оп} \quad (18.1)$$

$$Y_M = 2^N - 1, \quad (18.2)$$

где N – разрядность АЦП. Поэтому разрешающая способность АЦП также определяется величиной дискреты, зависящей от разрядности и опорного напряжения:

$$U_0 = U_{оп} / (2^N - 1). \quad (18.3)$$

Тогда основную погрешность АЦП можно вычислить по формуле

$$\delta_{\text{осн}} = \pm U_0 / U_M = \pm 1 / Y_M = 1 / (2^N - 1). \quad (18.4)$$

У реальных АЦП ступенчатая характеристика имеет дополнительные погрешности – *сдвига* (см. п. 5.7.1), *нелинейности*, *немонотонности* и др.

Результирующий код на выходе АЦП получается не мгновенно, а спустя некоторое время, которое называют *временем преобразования*. Следует отметить, что во время оцифровки входной сигнал должен быть неизменным, иначе возникает так называемая *апертурная погрешность*, при наличии которой выходной код не будет соответствовать входному сигналу в момент начала преобразования. Чтобы исключить апертурную погрешность, на вход АЦП включают схему выборки-хранения (см. §15.3).

18.1.2. Параллельный АЦП

Для реализации ступенчатой передаточной характеристики можно использовать, например, набор аналоговых компараторов, которые являются, по сути, простейшими однобитовыми АЦП. Для примера рассмотрим работу трех компараторов, которые будут контролировать входной сигнал (рис. 18.2). Пороговые уровни компараторов, последовательно кратные дискрете U_0 , установлены с помощью резистивного делителя с одинаковыми сопротивлениями. Подобная иллюстративная схема будет реагировать на четыре информационных уровня входного сигнала ($0, U_0, 2U_0, 3U_0$), что соответствует характеристике двухразрядного АЦП (рис. 18.3).

Количество единиц на выходах компараторов при плавном увеличении входного напряжения будет расти аналогично столбику термометра, поэтому такой код иногда называют *термометрическим*. Для свертки неэкономичного термометрического кода после компарато-

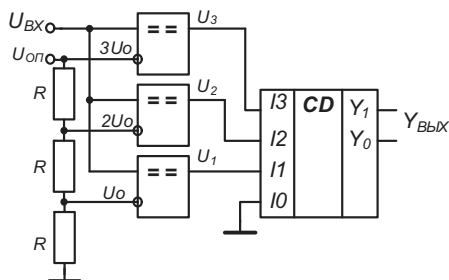


Рис. 18.2

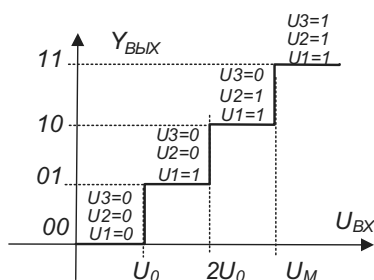


Рис. 18.3

ров включают приоритетный шифратор CD (см. п. 11.7.1), который превращает номер старшего сработавшего компаратора в нормальный двоичный код.

Разряды на выходах компараторов и шифратора образуются очень быстро и почти одновременно (параллельно), поэтому подобную структуру с набором компараторов называют параллельным АЦП.

Время преобразования параллельного АЦП определяется суммой задержек шифратора и самого медленного из компараторов. Для разных микросхем оно составляет величину от долей до десятков наносекунд.

В настоящее время параллельные АЦП являются самыми быстрыми, их используют на частотах выше десятков и сотен МГц, например, для оцифровки видеосигналов.

18.1.3. Последовательно-параллельный АЦП

Чем больше разрядность АЦП, тем больше требуется компараторов внутри микросхемы, и это является основным недостатком параллельной структуры. Поэтому при повышенной разрядности изготовители микросхем идут на определенную

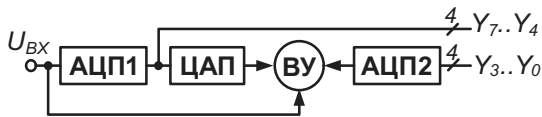


Рис. 18.4

потерю быстродействия и чисто параллельную структуру заменяют на более простую, *последовательно-параллельную* (рис. 18.4).

В такой схеме преобразование осуществляется двумя малоразрядными параллельными АЦП, например, 4-разрядными. Первый оцифровывает входной сигнал и получает старшие разряды $Y_4..Y_7$ кода. Эти разряды при помощи ЦАП превращаются снова в аналоговый вид и используются вычитающим устройством ВУ при вычислении «недооцифрованной» разности сигнала, которая при помощи второго АЦП преобразуется в младшие биты $Y_0..Y_3$ результата.

Последовательно-параллельные преобразователи характеризуются меньшим быстродействием по сравнению с параллельными за счет задержки ЦАП и последующей оцифровки разности, но имеют меньшее число компараторов. Так, например, для 8-разрядного параллельного АЦП необходимо 255 компараторов, а для последовательно-параллельного АЦП – всего 30. Для хранения мгновенных значений напряжения в каждом каскаде используются устройства выборки-хранения информации (см. п. 15.3.1).

Количество каскадов в таких АЦП может быть увеличено, поэтому они иногда называются многокаскадными или конвейерными. Примером отечественного конвейерного АЦП на 8 МГц может служить микросхема 1446ПВ2У.

При практическом использовании параллельных и конвейерных АЦП нужно иметь в виду, что, благодаря высокому быстродействию, они могут получать миллионы кодов за секунду. Поэтому от процессорных систем требуется адекватное реагирование – сохранение данных АЦП в режиме прямого доступа к памяти или использование вспомогательных буферных ОЗУ с последующей пересылкой в основную память. Некоторые микросхемы АЦП могут иметь собственную буферную память.

§18.2. АЦП на основе ЦАП и компаратора

18.2.1. Общие сведения

Процесс оцифровки иногда сравнивают с процессом взвешивания. Особенно это сходство проявляется в АЦП на основе компаратора, выступающего как бы в качестве двоичных весов, и ЦАП, выполняющего функции набора эталонов веса – гирь.

Структурная схема такого АЦП приведена на рис. 18.5. Схема управления СУ изменяет код в регистре РГ до тех пор, пока $U_{\text{ЦАП}}$ не приблизится к $U_{\text{ВХ}}$, что фиксируется по изменению напряжения компаратора U_K .

Процесс «уравновешивания» входного сигнала может происходить по разным алгоритмам,

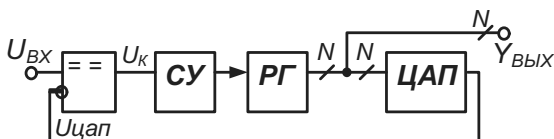


Рис. 18.5

среди которых выделяют развертывающий, следящий и поразрядно-уравновешивания.

18.2.2. Развертывающий алгоритм

Назван так потому, что напряжение $U_{\text{ЦАП}}$ (а значит и код $Y_{\text{ВЫХ}}$) ступенчато нарастает (развертывается) по ходу преобразования (рис. 18.6). Схема управления анализирует состояние компаратора и инкрементирует код до тех

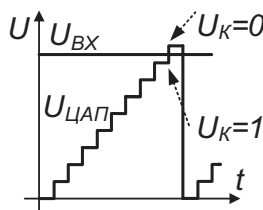


Рис. 18.6

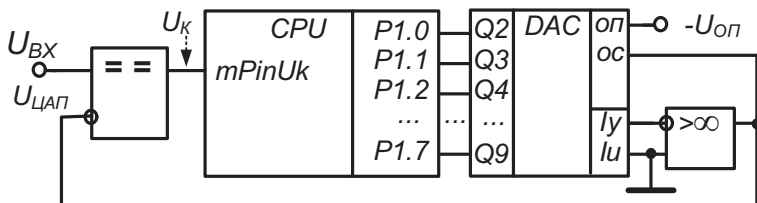


Рис. 18.7

пор, пока не изменится напряжение U_K , сигнализирующее о том, что $U_{\text{ЦАП}}$ стало больше U_{BX} .

Предположим, что в качестве схемы управления используется ОВМ x51 (рис. 18.7), управляющая внешним ЦАП и анализирующая сигнал от компаратора U_K .

Приведем программу оцифровки входного сигнала с использованием развертывающего алгоритма для байтового формата результата:

```

Mov mKodY, #0
mL6:
    Mov P1, mKodY      ; Вывод кода на ЦАП.
    Nop                ; Пауза, учитывающая задержку ЦАП
    Nop                ; и компаратора.
    JB mPinUk, mL7      ; Анализ состояния компаратора.
    Inc mKodY           ; Увеличение кода.
    Jmp mL6
mL7:

```

Конечно, реальные программы сложнее, например, они имеют защиту от переполнения и другие особенности.

После получения результата код обнуляется, и процесс преобразования начинается с начала. Поэтому время преобразования получается достаточно большим и зависящим от величины сигнала. Например, 8-битный АЦП получает максимальный результат через 255 тактов программы. Поэтому развертывающий алгоритм применяют редко.

18.2.3. Следящий алгоритм

На начальной стадии подобен развертывающему (рис. 18.8), но после получения первого результата код не обнуляется, а схема управления продолжает анализировать состояние компаратора, отслеживая входной

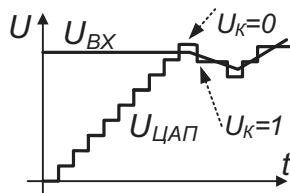


Рис. 18.8

сигнал. При увеличении $U_{\text{вх}}$ цифровой код, обычно образующийся в *реверсивном* счетчике (см. п. 13.1.5), инкрементируется, а при уменьшении – декрементируется. Поэтому время получения следующих цифровых отсчетов значительно сокращается.

Следящий АЦП теряет свои преимущества, если на входе применяется аналоговый коммутатор для оцифровки нескольких сигналов.

Следящий АЦП имеет возможность выдачи кода в *последовательном формате*. Для этого в качестве последовательного выхода может использоваться выход компаратора. Например, если на данном такте $U_k=1$, это означает, что входной сигнал увеличивается и надо увеличить выходной код; если $U_k=0$, то входной сигнал и код уменьшаются. Неизменный уровень приводит к чередованию U_k .

Следящий АЦП, использующий в своей работе предысторию процесса, является одним из прародителей современных сигма-дельта АЦП (см. §18.6).

18.2.4. Алгоритм поразрядного уравнивания

Такой алгоритм издавна применяют при взвешивании. Вначале ставят самую большую гирю (устанавливают старший разряд). Если весы (компаратор) показывают «много», то снимают гирю (сбрасывают разряд). Если «мало», то оставляют гирю на весах (разряд в результирующем коде). Процесс повторяют для всех имеющихся гирь (разрядов) до самой легкой, последовательно приближаясь с каждым шагом все ближе к результату (рис. 18.9). Поэтому второе название алгоритма – *алгоритм последовательного приближения*. Результат в подобных схемах получается достаточно быстро, через N тактов (где N – разрядность АЦП).

Для пояснения алгоритма ниже приведена программа для схемы рис. 18.7.

Mov mKodY, #0	; Начальная инициализация кода.
Mov mCountBit, #8	; Разрядность АЦП.
Mov mMaskaBitPtr, #0	; Маска для бита-указателя.
SetB C	; Установка бита-указателя.
mL8:	
Mov A, mMaskaBitPtr	; Сдвиг маски
RRC A	; для установки

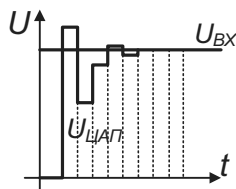


Рис. 18.9


```

Mov mMaskaBitPtr, A      ; очередной разряда.
OrL A, mKodY; Объединение маски с промежуточным
Mov P1, A                ; результатом и вывод на ЦАП.
JB mPinUk, mL9            ; При Uk=1 бит не учитывается
                          ; Uk=0, результат сохраняется.
                          Mov mKodY, A
mL9:
  DJNZ mCountBit, mL8    ; Переход к следующему разряду.

```

Рассмотренный алгоритм аппаратно реализуется во многих микросхемах АЦП массового применения, включая встроенные АЦП в ОВМ.

§18.3. Особенности обслуживания микросхем АЦП

18.3.1. Обслуживание АЦП K572ПВЗ

Эта микросхема последовательного приближения имеет 8 разрядов [8] и параллельный интерфейс, что хорошо согласуется с байтовым форматом ОВМ типа x51. Линии данных АЦП имеют третье состояние и в принципе могут бесконфликтно подключаться к общим шинам данных процессора (см. п. 14.5.1). Но некоторые ОВМ не имеют внешних шин, поэтому в данном случае организуем взаимодействие по параллельному интерфейсу программно (рис. 18.10).

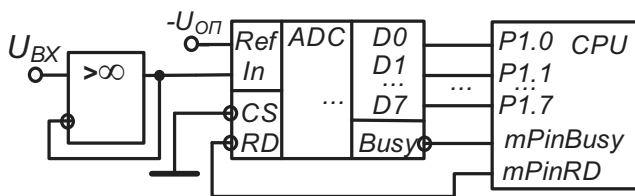


Рис. 18.10

Цепи питания и тактирования АЦП и ОВМ на схеме для простоты не показаны. Повторитель на входе нужен для увеличения входного сопротивления, поскольку микросхема K572ПВЗ имеет низкоомный вход.

Управление интерфейсом АЦП осуществляется при помощи входных сигналов *CS* (*Chip Select* – выбор микросхемы) и *RD* (*Read* – читать), которые, кроме основной функции выбора микросхемы и управления третьим состоянием данных, инициируют начало преобра-

зования. Для простоты сигнал **CS** делаем постоянно активным, что допустимо для несложных конфигураций.

Временные интерфейсные диаграммы для АЦП имеют вид, представленный на рис. 18.11. В момент, когда сигнал **RD** будет активным (нулевым), на линиях данных появляется результат

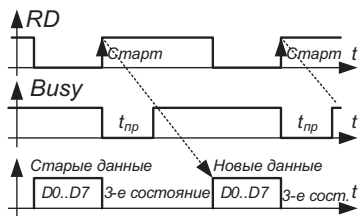


Рис. 18.11

предыдущего преобразования, который можно прочесть программой. При переходе сигнала **RD** в пассивное единичное состояние происходит запуск АЦП и бит-индикатор занятости – **BUSY**, принимает активное нулевое значение.

Для чтения с максимально возможной скоростью нужно дождаться пассивного состояния бита **BUSY** и после этого прочесть данные. Операция чтения выходного кода АЦП осуществляется в три этапа – сброс **RD**, собственно чтение (копирование) и установка **RD**:

mProcGetADC:

```
JNB mPinBUSY, mProcGetADC ; Ждем готовности АЦП
Clr mPinRD                  ; Формируем сигнал чтения
Mov mKodY, P1              ; Читаем данные.
SetB mPinRD                ; Запускаем АЦП на новый отсчет.
Ret
```

Использование этой процедуры приема кода от АЦП было описано ранее в п. 9.9.2.

Если высокая скорость оцифровки не нужна, то операторы ожидания исключают, а чтение производят обычно по таймеру, через время, большее $t_{пр}$.

Более сложные микросхемы АЦП имеют на входе встроенный аналоговый коммутатор и буферное ОЗУ и требуют перед началом работы соответствующей инициализации. Кроме того, отметим, что параллельный интерфейс занимает большое количество внешних выводов ОВМ, особенно при повышенной разрядности выходного кода.

18.3.2. АЦП с последовательным интерфейсом

Такие преобразователи имеют значительно меньшее число внешних выводов, что упрощает подключение к ОВМ. Однако программа обслуживания становится более сложной, поскольку требуется преобразование последовательного формата принимаемых данных в парал-

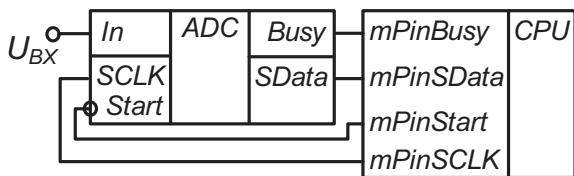


Рис. 18.12

лельный. Последовательный интерфейс большинства АЦП является синхронным и двухпроводным, подобным **SPI** (см. п. 12.8.1). Для рассмотрения особенностей интерфейса и его обслуживания рассмотрим схему, в которой используется АЦП типа AD7896 (рис. 18.12).

Эта 12-разрядная микросхема последовательного приближения со временем преобразования около 8 мкс не требует внешних элементов. Источник опорного напряжения, устройство выборки-хранения, тактовый генератор и другие узлы АЦП выполнены внутри микросхемы. Допустимый уровень входного сигнала от 0 до напряжения питания.

Кроме двух интерфейсных сигналов – информационного **SData** (**Serial Data**) и тактового **SCLK** (**Serial CLock**), для взаимодействия с процессором в АЦП используют вход для запуска процесса оцифровки **Start** и выход – индикатор занятости **Busy**.

Запуск АЦП происходит в момент отрицательного перепада на входе **Start** (рис. 18.13). После этого внутреннее устройство выборки-хранения переходит в режим хранения и устанавливается выходной сигнал **Busy** на время $t_{np} \approx 8$ мкс.

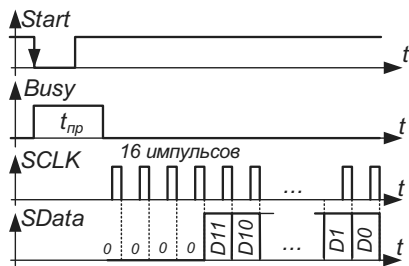


Рис. 18.13

Если стартовый сигнал остается нулевым после окончания времени преобразования, то АЦП переходит в экономичный режим, «засыпает» до следующего отрицательного перепада.

Из АЦП всегда выводятся 16 разрядов последовательных данных, начиная со старших битов – четыре неиспользуемых нуля и 12 битов результата преобразования. В качестве опорного используется напряжение питания, поэтому вес младшего разряда (см. п. 18.1.1) будет равен $U_0 = U_{оп} / 2^{12} = 5 \text{ В} / 4096 \approx 1,2 \text{ мВ}$.

18.3.3. Программа обслуживания АЦП с последовательным интерфейсом

Напомним, что сигналы *Start* и *SCLK* являются выходными для процессора, а *SData* и *Busy* – входными.

Программа после формирования импульса запуска *Start*, как и прежде (см. п. 18.3.2), должна дождаться пассивного состояния *Busy* или сделать паузу, большую времени преобразования. Затем необходимо 16 раз (дважды по 8) установить тактовый сигнал и принять со сдвигом информационный бит и сбросить тактовый сигнал. Отметим, что первыми в последовательной цепочке данных идут старшие значащие биты.

```

    Clr A
    Clr mPinStart           ; Запускаем АЦП
    Mov mCountBit, #8       ; Инициализируем счетчик битов.
    SetB mPinStart          ; Убираем стартовый сигнал.
mL11:
    JB mPinBusy, mL11       ; Ждем готовности АЦП.
mL12:
    SetB mPinSCLK           ; Формируем тактовый сигнал.
    Mov C, mPinSData        ; Читаем информационный бит
    Clr mPinSCLK           ; Убираем тактовый сигнал.
    RLC A                   ; Подготавливаем место биту.
    DJNZ mCountBit, mL12    ; Закончили прием 8 битов?
    Mov mKodYH, A           ; Да, сохраняем старший байт.
    Clr A                   ; Готовим цикл
    Mov mCountBit, #8       ; для приема младшего байта.
mL13:
    SetB mPinSCLK
    Mov C, mPinSData
    RLC A
    Clr mPinSCLK
    DJNZ mCountBit, mL13
    Mov mKodYL, A           ; Сохраняем младший байт.

```

18.3.4. Особенности встроенных АЦП

Большинство новых семейств ОВМ имеют внутренние 8..12-разрядные АЦП последовательного приближения, выполненные вместе со схемой выборки-хранения и аналоговым коммутатором, например, на 8 входов.

Встроенный АЦП представляется программисту набором спец-регистров – регистра «результата» (например, с именем *ADCDData*),

регистра «режима» (*ADCMode*), регистра «управления» (*ADCControl*) и регистра «состояния» (*ADCStatus*). Назначение этих регистров соответствуют их названию. Например, в регистре «режима» обычно размещены биты номера выбранного канала коммутатора, бит автоматического или одиночного преобразования и другие биты режима. В регистре «управления» основным является бит запуска АЦП, в регистре «статуса» – бит занятости или завершения преобразования. Для компактности в некоторых ОВМ биты режима, управления и статуса могут размещаться в одном или двух комбинированных регистрах «управления-состояния».

Обслуживание встроенного АЦП в простейшем случае заключается в задании режима путем записи кода в регистр режима, установке бита запуска и ожидании пассивного значения бита занятости АЦП. После этого читается результат.

При использовании встроенных АЦП нужно иметь в виду, что близко расположенное процессорное ядро наводит на аналоговые цепи сильные электромагнитные помехи, которые искажают результат и снижают на 1–2 разряда эквивалентную разрядность кода, которой можно верить. Для повышения точности встроенных АЦП нужно строго выполнять рекомендации производителей микросхем. Например, правильно разводить печатную плату с аналоговыми и цифровыми цепями и при возможности программно приостанавливать работу процессора на время преобразования входной величины для уменьшения влияния импульсных помех от процессорного ядра.

§18.4. Время-импульсный АЦП развертывающего типа

18.4.1. Основные сведения

Во время-импульсных АЦП входное напряжение предварительно преобразуется в импульсы, длительность которых пропорциональна входному сигналу. Временные параметры этих импульсов достаточно легко измерять при помощи двоичного счетчика (см. п. 13.4.1) или программным путем (см. §10.7).

Простейшие из подобных преобразователей аппаратно реализуют развертывающий алгоритм на основе компаратора и генератора линейно-изменяющегося («пилообразного») напряжения ГЛИН. Рассмотрим, структурную схему, представленную на рис. 18.14. В целом

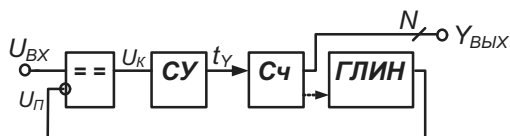


Рис. 18.14

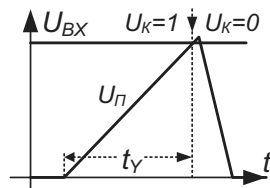


Рис. 18.15

она схожа с *развертывающим* АЦП на основе компаратора и ЦАП (см. п. 18.2.1), только роль ЦАП выполняет генератор «пилы» (см. временную диаграмму на рис. 18.15). Схема управления СУ в данном случае запускает ГЛИН и измеряет интервал t_{γ} и поэтому может представлять простейший счетчик импульсов от генератора образцовых заполняющих импульсов (см. п. 13.4.1).

Благодаря своей простоте подобные АЦП в свое время широко применялись в первых цифровых вольтметрах. Но сравнительно большое время преобразования и чувствительность к импульсным помехам во входном сигнале ограничили область использования времяимпульсных АЦП развертывающего типа. В настоящее время они применяются только в самых простых микропроцессорных системах для достаточно грубой оцифровки.

18.4.2. Схема простейшего время-импульсного АЦП

Учитывая, что большой точности от таких АЦП не требуется, применим в качестве ГЛИН время-задающую RC -цепочку с разрядным ключом (рис. 18.16). При заряде изменение экспоненциального напряжения на емкости (рис. 18.17) на начальном участке будет происходить по примерно линейному закону.

Функции схемы управления будем выполнять программно. Отметим также, что для дальнейшего упрощения схемы используем встроенный компаратор, имеющийся в некоторых ОВМ семейства x51 (см. п. 7.2.3), и внутренний ключ для разряда конденсатора. Поэтому функциональная схема для ввода аналоговой информации в ОВМ, представленная на рис. 18.16, будет иметь всего два внешних элемента – резистор и конденсатор.

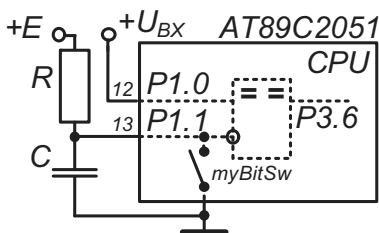


Рис. 18.16

В исходном состоянии внутренний ключ, управляющий битом порта, проводит и поэтому напряжение на конденсаторе равно нулю. Программа в начале цикла измерения переводит ключ в непроводящее состояние и начинает при помощи таймера отсчет времени t_{γ} .

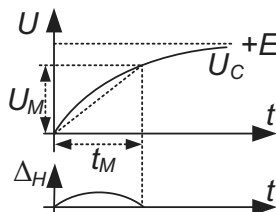


Рис. 18.17

Напряжение на конденсаторе начинает возрастать в результате заряда через резистор R , стремясь достигнуть напряжения питания. В момент сравнения входного сигнала с «пилой» срабатывает компаратор. Программа, контролирующая его состояние, заканчивает измерение t_{γ} и переводит ключ в исходное проводящее состояние, подготавливая схему к следующему измерению путем разряда конденсатора.

При рассмотрении программы полагаем, что таймер $T0$ ранее уже был установлен в режим 16-битного счета (см. п. 8.6.2):

Mov TH0, #0	; Обнуление таймера,
Mov TL0, #0	; то есть $Y_{\text{вых}} = 0$.
SetB mPinSw	; Ключ – в непроводящее состояние.
SetB TR0	; Пуск таймера на измерение t_{γ} .

mL14:

JB P3.6, mL14	; Ожидание переключения компаратора.
Clr TR0	; Останов таймера.
Mov mKodYH, TH0	; Читаем старший
Mov mKodYL, TL0	; и младший байты результата.

18.4.3. Расчетные соотношения

Пренебрегая остаточными параметрами ключа и компаратора, можно записать уравнение для напряжения на конденсаторе, подобное (17.7):

$$U_c(t) = U_c(\infty) - [U_c(\infty) - U_c(0)] \exp(-t/RC). \quad (18.5)$$

Далее, учитывая, что $U_c(\infty) \approx E$, а $U_c(0) \approx 0$, можно получить

$$U_c(t) = E - E e^{(-t/RC)}. \quad (18.6)$$

В момент t_{γ} входное напряжение сравнивается с напряжением на конденсаторе, то есть:

$$U_{\text{BX}} = E - E \exp(-t_{\gamma}/RC), \quad (18.7)$$

что позволяет, решив последнее уравнение, найти зависимость длительности импульса от входного сигнала:

$$t_Y = RC \cdot \ln(E / (E - U_{BX})). \quad (18.8)$$

Для простейшей аппроксимации полученной нелинейной функции воспользуемся прямой, проходящей на рис. 18.17 через ноль и точку на с максимальными координатами t_M и U_M :

$$t_Y \approx U_{BX} t_M / U_M. \quad (18.9)$$

При этом расчетная величина постоянной времени RC будет определяться по формуле:

$$RC \approx t_M / \ln(E / (E - U_M)). \quad (18.10)$$

Пусть частота кварца равна 12 МГц, тогда внутренний таймер Т0 инкрементируется импульсами с периодом $T_T = 1$ мкс. Зададимся некоторыми необходимыми параметрами. Пусть разрядность рассмотренного АЦП будет равна 11, тогда $Y_M = 2^N - 1 = 2047$, то есть $t_M = Y_M \cdot T_T = 2047$ мкс.

Считая напряжение питания равным 5 В, ограничимся входным сигналом в пределах примерно линейного участка экспоненты. Например, пусть $U_M = 2,047$ В. Тогда $RC = 2047 \cdot 10^{-6} / \ln(5 / (5 - 2,047)) \approx 4096$ мкс. Выбирая величину $R = 100$ кОм, получим $C = 0,05$ мкФ.

При таком расчете одна единица младшего разряда полученного кода, как следует из уравнения (18.9), будет соответствовать напряжению примерно 1 мВ.

18.4.4. Уменьшение погрешности

Экспоненциальная форма напряжения на конденсаторе определяет погрешность схемы от нелинейности Δ_H (рис. 18.17). Уменьшить ее можно двумя способами, аппаратным или программным.

При аппаратном способе времязадающий резистор заменяют токостабилизирующим элементом, например, транзистором [5]. Тогда конденсатор будет заряжаться постоянным током, а напряжение на нем будет линейно изменяться со временем, поэтому погрешность от нелинейности будет существенно уменьшена.

Программное уменьшение погрешности подразумевает использование при вычислении величины U_{BX} вместо (18.9) более точного нелинейного соотношения (18.8), что требует определенных вычислительных ресурсов. В большинстве случаев ограничиваются вычислением поправки Δ_H , когда формулу для погрешности табулируют и используют полученные цифры в виде поправочной таблицы к результату.

§18.5. АЦП двойного интегрирования

18.5.1. Общие сведения

Все предыдущие АЦП на основе компаратора, анализирующего входной сигнал, очень чувствительны к помехам. Например, импульсные «иголки» на входе вызывают преждевременное срабатывание компаратора и соответствующее искажение результата. От этих недостатков свободен АЦП двойного интегрирования, в котором преобразование осуществляется в два этапа, что позволяет снизить влияние помех.

Такой АЦП (рис. 18.18) помимо компаратора и схемы управления имеет перекидной ключ и интегратор (см. п. 6.1.2). Вначале перекидной ключ подсоединяется к $U_{ВХ}$ и в течение эталонного промежутка времени $T_э$ происходит интегрирование входного сигнала. Напряжение $U_{и}$ при этом с учетом инверсии интегратора уменьшается (рис. 18.19).

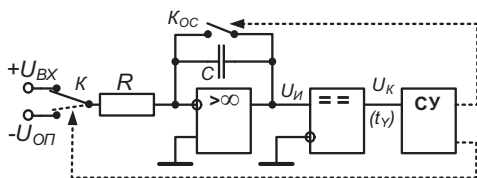


Рис. 18.18

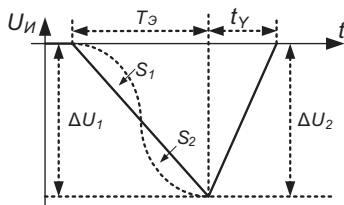


Рис. 18.19

Затем перекидной ключ переводится в нижнее положение и на интегратор подается опорное напряжение обратной по сравнению с $U_{ВХ}$ полярности. Напряжение $U_{и}$ начнет изменяться в другую сторону, а блок управления будет производить отсчет интервала времени $t_γ$, который закончится при $U_{и} = 0$ и будет пропорциональным $U_{ВХ}$.

Оценим изменения выходного напряжения интегратора за интервалы времени $T_э$ и $t_γ$ при нулевых начальных условиях и предположении постоянства $U_{ВХ}$ и $U_{ОН}$:

$$\Delta U_1 = \frac{1}{RC} \int_0^{T_э} U_{ВХ} dt = \frac{U_{ВХ} T_э}{RC}. \quad (18.11)$$

$$\Delta U_2 = \frac{1}{RC} \int_0^{t_γ} U_{ОН} dt = \frac{U_{ОН} t_γ}{RC}. \quad (18.12)$$

Приравняв правые части уравнений, найдем длительность

$$t_γ = U_{ВХ} T_э / U_{ОН} \quad (18.13)$$

Отсюда следует вывод, что результат измерения t_{γ} не зависит от постоянной времени RC интегратора и определяется только интервалом T_{γ} и напряжением $U_{оп}$, которые нетрудно сделать стабильными.

Важной особенностью АЦП двойного интегрирования является и то, что благодаря интегрированию входного сигнала на первом этапе усредняются и тем самым подавляются высокочастотные помехи.

Кроме того, у таких АЦП появляется возможность подавления сетевой помехи промышленной частоты 50 Гц. Если выбрать интервал $T_{\gamma} = T_c = 1/50 \text{ Гц} = 20 \text{ мс}$, то помеха с сетевой частотой не будет влиять на результат измерения, хотя, как видно из рис. 18.19, форма выходного сигнала интегратора в этом случае искажается (пунктирная синусоида). Но значение $U_{и}$ в момент окончания T_{γ} остается неизменным, поскольку интегралы полупериодов помехи равны и противоположны по знаку (вольт-секундные площади полуволн S_1 и S_2 на рисунке одинаковы), что обеспечивает их взаимную компенсацию. Эти качества очень важны в промышленных применениях, где уровень сетевых помех достаточно велик.

Недостатком АЦП двойного интегрирования является большое время преобразования (более 20 мс), что является своеобразной расплатой за высокие помехозащищенность и точность. Однако в тех случаях, когда быстродействие не требуется, такие АЦП широко используются, например, в цифровых вольтметрах и тестерах. Выпускаемые микросхемы отличаются в основном интерфейсом, разрядностью и возможностью самокалибровки, для чего вводится третий этап интегрирования.

18.5.2. Реализация АЦП двойного интегрирования

Реализация на ОВМ функций схемы управления АЦП двойного интегрирования (рис. 18.20) при минимуме внешних элементов описана в [27].

Ключ в цепи обратной связи интегратора в данном случае не используется, поскольку нулевое напряжение на интеграторе задается при помощи специального алгоритма. Для этого интегрируют опорный сигнал, добиваясь переключения компаратора (рис. 18.21). За-

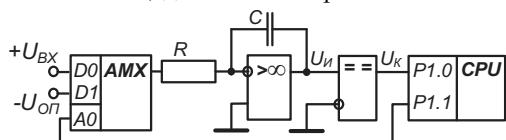


Рис. 18.20

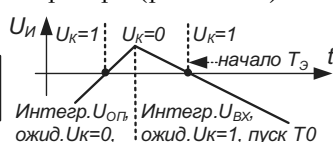


Рис. 18.21

тем подают входной сигнал, и в момент следующего переключения компаратора начинают отсчет T_3 и рабочее интегрирование входного сигнала.

В остальном программа [27] представляет последовательность следующих фрагментов – вначале загружается в таймер код T_3 (таймер не считает), затем выполняется алгоритм обнуления интегратора, после чего начинается интегрирование $U_{вх}$ и пускается таймер для отсчета T_3 . После окончания T_3 , что фиксируется по флажку переполнения таймера, на интегратор подается $U_{оп}$, но таймер не останавливается, начиная счет t_r . Данный алгоритм поясняется на рис. 18.22 на основе числовой оси таймера.

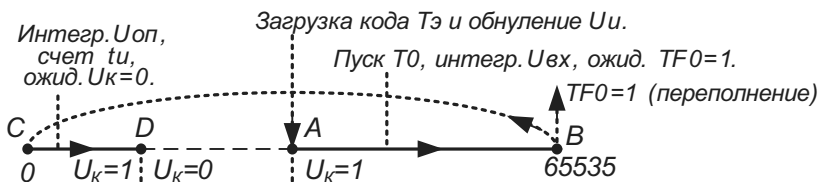


Рис. 18.22

```

Mov TH0,#mKodTH      ; Загрузка
Mov TL0,#mKodTL      ; таймера.
SetB P1.0             ; Подача Uоп для обнуления.
mL15:
JB P1.1 mL15          ; Ожидание перехода через ноль.
Clr P1.0              ; Подача Uвх на интегратор.
mL16:
JB P1.1, mL16         ; Ожидание нуля интегратора.
SetB TR0              ; Пуск таймера на отсчет T3 (точка «А»).
my17:
JNB TF0, my17         ; Ожидание окончания T3.
SetB P1.0             ; Начало интегрирования Uоп (точки «В» и «С»).
mL18:
JB P1.1, mL18         ; Ожидание окончания tr.
Clr TR0              ; Останов таймера (точка «D»).
Clr TF0              ; Очистка флажка таймера.
Mov mKodYH, TH0      ; Читаем байты
Mov mKodYL, TL0      ; результата.

```

Программное управление позволяет в принципе реализовать оперативное изменение параметров АЦП, например, уменьшая T_3 , получить быстрее приближенный результат, а при необходимости точной оцифровки вернуться к расчетным временным соотношениям.

Заметим, что схему можно упростить, если использовать внутренний компаратор ОВМ.

§18.6. Частотные и сигма-дельта АЦП

18.6.1. Частотные АЦП

Другим типом интегрирующих АЦП являются устройства с предварительным преобразованием входного напряжения в пропорциональную частоту следования импульсов $F_Y = kU_{ВХ}$. Измерение частоты сводится, как известно (см. п. 13.4.1), к подсчету числа импульсов на достаточно большом интервале времени, что эквивалентно цифровому интегрированию преобразованной информации. Поэтому частотные АЦП, как и АЦП двойного интегрирования, способны при правильном выборе интервала осреднения подавлять помехи, включая помеху от промышленной сети.

18.6.2. Преобразователь напряжения в частоту (ПНЧ)

В схеме частотного преобразователя (рис. 18.23) также используется принцип «весов», роль которых, как и прежде, выполняет компаратор. Он запускает ждущий генератор (см. п. 17.4.1) с образцовой длительностью импульса t_0 , поэтому «гири» в цепи обратной связи, компенсирующие входной сигнал, в данном случае используют времяимпульсное представление информации (см. временные диаграммы на рис. 18.24). Для сравнения разнополярных $U_{ВХ}$ и $U_{ОС}$ используется параллельная схема на основе двух резисторов, то есть, по сути, сравниваются токи (см. п. 17.1.3).

Разностный сигнал ΔU из-за импульсного характера $U_{ОС}$ нуждается в осреднении, для чего перед компаратором используется интегратор (см. п. 6.1.2). При этом интегратор работает непрерывно, а не тактами,

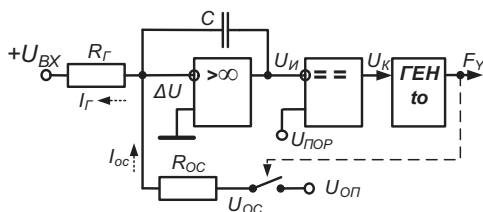


Рис. 18.23

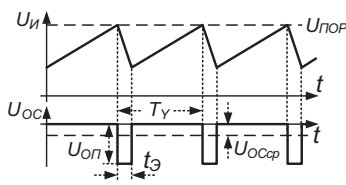


Рис. 18.24

как в АЦП двойного интегрирования (см. п. 18.5.1). Это позволяет увеличить быстродействие.

Для формирования импульсного сигнала ОС в схеме используется ключ, подсоединенный к источнику опорного напряжения. Поэтому вольт-секундная площадь импульса обратной связи с образцовой длительностью всегда постоянна. Иногда говорят, что в схеме, таким образом, происходит уравнивание заряда емкости интегратора.

Заряд конденсатора интегратора осуществляется входным напряжением U_{BX} , а разряд производится обратным по полярности импульсом с постоянной вольт-секундной площадью.

Напряжение на выходе интегратора, несущее информацию о разностном сигнале, линейно растет, пока не сравняется с пороговым напряжением $U_{ПОР}$, что сигнализирует о том, что среднего значения $U_{ОС}$ не хватает для компенсации U_{BX} . То есть компаратор должен обеспечивать равенство средних значений напряжения U_{BX} , которое действует весь период, и напряжения $U_{ОС}$, которое противодействует входному на время t_0 :

$$\frac{1}{RC} \int_0^{T_Y} U_{BX}(t) dt = \frac{1}{RC} \int_0^{t_0} U_{ОН} dt. \quad (18.14)$$

Проводя интегрирование, можно найти:

$$F_Y = \frac{1}{T_Y} = \frac{U_{BXcp}}{U_{ОС} \cdot t_0}. \quad (18.15)$$

Поэтому, измеряя частоту, как видно из уравнения, можно получить код, пропорциональный напряжению U_{BX} .

Сравнивая частотный АЦП и АЦП двойного интегрирования можно заметить, что они имеют примерно одинаковые потенциальную помехозащищенность и стабильность. Но результат в частотных АЦП получается быстрее, особенно если использовать цифровые методы обработки сигнала после компаратора, несущие информацию о разности U_{BX} и $U_{ОС}$. Поэтому дальнейшее развитие принципа частотного преобразования напряжения с времяимпульсной обратной связью привело к созданию сигма-дельта АЦП, которые в настоящее время широко используются для точной оцифровки сигналов звуковой частоты, например, аудиосигналов.

18.6.3. Структура сигма-дельта АЦП

Сигма-дельта АЦП имеет практически такую же структуру, как и ПНЧ (рис. 18.25).

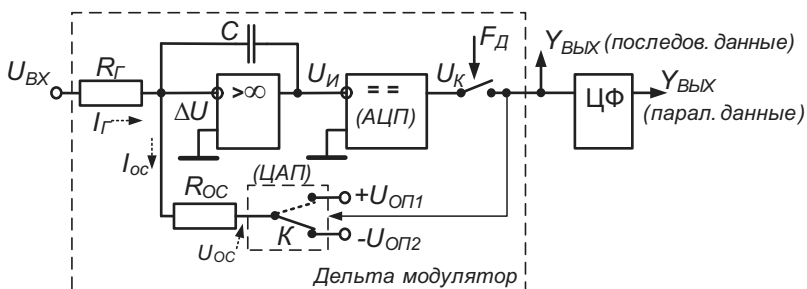


Рис. 18.25

Первое отличие – на компаратор вместо $U_{пор}$ подан ноль. Поэтому компаратор, выступающий в роли однобитового АЦП, только определяет знак осредненной интегратором *разности* ΔU (больше или меньше). Отсюда возник термин «дельта».

Второе отличие – компаратор сам не устанавливает $U_{ос}$, это делает цифровая схема управления с постоянной частотой дискретизации $F_{д}$.

Получившаяся схема, называемая *дельта-модулятором*, похожа по смыслу на следящий АЦП (см. п. 18.2.3), только вместо обычного ЦАП здесь используется двуполярный импульсный ЦАП на основе ключа K . В последовательном потоке данных от дельта-модулятора содержатся цифровые значения входного напряжения. В следящем АЦП импульсы дискретизации подсчитываются (усредняются) реверсивным счетчиком (см. п. 13.1.5). Если последовательный код равен «1», то результирующий код инкрементируется, если – «0», то происходит декремент (см. п. 18.2.3). В сигма-дельта АЦП эти функции выполняет цифровой фильтр ЦФ. Наличие подобного цифрового суммирования (предыстории) в АЦП отражено термином «сигма».

Иллюстративные временные диаграммы для случая $U_{оп1} = U_{оп2} = U_{оп}$ приведены на рис. 18.26.

Не вдаваясь в математические подробности, которых больше, чем схемотехники, рассмотрим основные идеи, лежащие в основе сигма-дельта АЦП.

18.6.4. Передискретизация

Основной проблемой при сравнении напряжения $U_{вх}$ с импульсным напряжением $U_{ос}$ является подавление пульсаций разностного сигнала ΔU . Интегратор, постоянную времени которого не желательно увеличивать, не справляется с этим, поскольку высокочастотный

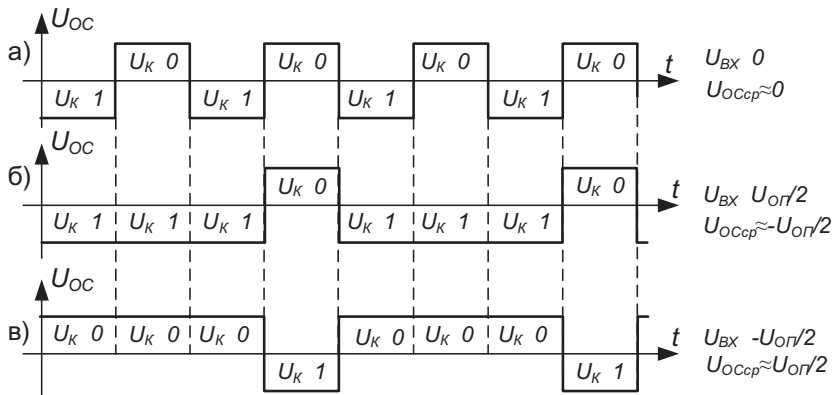


Рис. 18.26

шум, содержащийся во входном сигнале, и пульсации от наличия боковых частот при дискретизации дополняются шумами квантования. Например, даже при постоянном входном сигнале точное равенство компаратором не фиксируется, и он выдает чередующиеся нули и единицы (рис. 18.26а), а результат изменяется на ± 1 . Для подавления указанных шумов требуется осреднение на достаточно большом интервале времени, что реализуется цифровым фильтром.

Для увеличения быстродействия фильтра и его эффективности в сигма-дельта АЦП резко увеличивают частоту дискретизации и называют этот прием *передискретизацией*. С помощью цифрового фильтра, благодаря повышенной частоте и большему объему входных данных, получают результат быстрее и точнее.

А в заключение (для снижения темпа оцифровки до расчетного) результат прореживают, отбрасывая избыточные высокочастотные данные. Этот процесс называют *децимацией*.

18.6.4. Замена одноканальных устройств

В реальных сигма-дельта АЦП используют и ряд других приемов для увеличения точности и быстродействия. Например, вместо одноканального компаратора после интегратора ставят 4-разрядный АЦП, что увеличивает информативность анализа разностного сигнала, повышает быстродействие и достоверность работы цифрового фильтра и соответственно конечного результата.

Ряд усовершенствований касается и цепи обратной связи. Например, двухуровневый ключ заменяют многоуровневым, что позволяет

более точно приближаться к входному сигналу, аппроксимировать его. Поэтому реализуемую ЦАП функцию часто называют аппроксимирующей. Она может быть ступенчатой, как в нашем случае, линейной, квадратичной и т. д. С возрастанием степени функции возрастает точность, но увеличивается сложность изготовления.

Вместо интегратора иногда используют полосовые фильтры, выделяющие основную гармонику разностного периодического сигнала. В последнее время в состав прецизионных АЦП стали включать сигнальный процессор, который не только работает как программируемый многофункциональный цифровой фильтр, но и позволяет обеспечить ряд дополнительных возможностей, улучшая пользовательские характеристики.

В заключение отметим, что сигма-дельта АЦП являются в настоящее время самыми точными устройствами для оцифровки. Их разрядность в зависимости от типа составляет величину от 16 до 24. Они не требуют устройств выборки-хранения и практически не имеют внешних элементов, даже конденсатор интегратора изготавливается прямо на кристалле микросхемы, поскольку емкость его не превышает десятков пикофард.

По быстродействию сигма-дельта АЦП превосходят АЦП двойного интегрирования, но уступают АЦП последовательного приближения.

Практическое обслуживание микросхем сигма-дельта АЦП зависит от типа выходного цифрового интерфейса. Большинство из них имеют последовательный интерфейс типа SPI (см. п. 12.8.1) или I²C [25]. Быстродействующие микросхемы снабжены параллельным интерфейсом. Область применения – измерительное оборудование разного назначения и звуковая техника.

Список используемых сокращений

АЛУ – арифметико-логическое устройство
АЦП – аналого-цифровой преобразователь
БИС – большая интегральная схема
ВУ – внешнее устройство
ИУПТ – инвертирующий усилитель постоянного тока
ИОУ – интегральный операционный усилитель
ИУПТ – инвертирующий усилитель постоянного тока
КПД – коэффициент полезного действия
МОП – металл-окисел-полупроводник
НУПТ – неинвертирующий усилитель постоянного тока
ОБ – общая база
ОВМ – однокристалльная вычислительная машина.
ОЗУ – оперативное запоминающее устройство.
ОС – обратная связь
ООС – отрицательная обратная связь
ОЭ – общий эмиттер
ПЗУ – постоянное запоминающее устройство
ПЛМ – программируемая логическая матрица
ПОС – положительная обратная связь
РОН – регистр общего назначения
РПЗУ – репрограммируемое постоянное запоминающее устройство
СнК – система на кристалле
ТКН – температурный коэффициент напряжения
ТЛ – транзисторная логика
УГО – условное графическое обозначение
УПТ – усилитель постоянного тока
УВЧ – усилитель высокой частоты
УНЧ – усилитель низкой частоты
ЦАП – цифро-аналоговый преобразователь
ШИМ – широтно-импульсная модуляция
ЭДС – электродвижущая сила

Литература

1. БУКРЕЕВ, И. Н., Микроэлектронные схемы цифровых устройств, [Текст] / И. Н. Букреев, В. И. Горячев, Б. М. Мансуров. – 4-е изд., перераб. и доп. – М.: «Техносфера», 2009. 712 с. – ISBN 978-5-94836-197-0.
2. ВОЛОВИЧ, Г. И. Схемотехника аналоговых и аналого-цифровых электронных устройств [Текст] / Г. И. ВОЛОВИЧ. – М.: «Додэка-XX1», 2005. 528 с. – ISBN 5-94120-074-9.
3. ГОСТ 2.759-82. Обозначения условные графические в схемах. Элементы аналоговой техники. – М.: «Изд-во стандартов». 1988. – 13 с.
4. ГОСТ 2.743-91. Обозначения условные графические в схемах. Элементы цифровой техники. – М.: «Изд-во стандартов». 2003. – 44 с.
5. ГУСЕВ, В. Г., Электроника и микропроцессорная техника: Учеб. для вузов / В. Г. Гусев, Ю. М. Гусев. – 4-е изд., доп. – М.: «Высшая школа», 2006, 799 с. ISBN 5-06-005680-5.
6. ГУК, М. Интерфейсы ПК [Текст]: Справочник / М. ГУК. – СПб.: «Питер», 2000. 416 с. - ISBN 5-8046-0030-3.
7. ИЦХОКИ, Я. С., Импульсные и цифровые устройства [Текст] / Я. С. Ицхоки, Н. И Овчинников. – М. «Советское радио», 1972, 592 с.
8. ИНТЕГРАЛЬНЫЕ микросхемы: Микросхемы для аналого-цифрового преобразования и средств мультимедиа [Текст]. Выпуск 1. – М.: «ДОДЭКА», 1996. – 384 с. – ISBN 5-87835-008-4
9. ИНТЕГРАЛЬНЫЕ микросхемы: Перспективные изделия. Выпуск 2. – М.: «ДОДЭКА», 1996. – 96 с. ISBN 978-5-87835-006-8.
10. КАСПЕР, Э. Программирование на языке Ассемблера для микроконтроллеров семейства i8051 [Справ. изд.]. / Э. Каспер. – М.: «Горячая линия – Телеком», 2004. – 191 с. – ISBN 5-93517-104-X.
11. КОНОНОВ, Б. Н. Симметричные триггеры на плоскостных полупроводниковых триодах [Текст]. / Б. Н. Кононов. – М.: «Госэнергоиздат». 1960. 160 с.
12. ЛЕЙТМАН, М. Б. Нормирующие измерительные преобразователи электрических сигналов [Текст] / М. Б. Лейтман. – М.: «Энергоатомиздат», 1986. – 144 с
13. МИКУШИН, А. В. Занимательно о микроконтроллерах [Текст] / А. В. Микушин. – СПб.: «БХВ-Петербург», 2006. – 432 с. – ISBN 5-94157-571-8.

14. НЕФЕДОВ, А. В. Интегральные микросхемы и их зарубежные аналоги [Справочник]: Тома 1-12 / А. В. Нефедов. – М.: «ИП РадиоСофт», 1997-2001. – ISBN 5-93937-021-4, – ISBN 5-93937-053-2.
15. НОВИКОВ Ю. В., Разработка устройств для персонального компьютера типа IBM PC 2000 [Практ. пособие] / Ю. В. Новиков, О. А. Калашников, С. Э. Гуляев. – М.: «ЭКОМ», 2000. – 224 с. – ISBN 5-7163-0009-X.
16. НОВИКОВ Ю. В. Основы цифровой схемотехники. Базовые элементы и схемы. Методы проектирования [Текст]. / Ю. В. Новиков. – М.: «Мир», 2001. – 379 с. – ISBN 5-03-003449-8.
17. ОДНОКРИСТАЛЬНЫЕ микроЭВМ [Справ.]. / Боборыкин А. В. и др. – М.: «МИКАП», 1994. – 400 с. ISBN 5-85959-030-X.
18. ОПАДЧИЙ, Ю. Ф. Аналоговая и цифровая электроника [Учеб. для вузов] / Ю.Ф. Опадчий, О. П. Глудкин, А.И. Гуров. -М.: «Горная линия – Телеком», 2005. – 768 с. – ISBN 5-93517-002-7.
19. ОПЕРАЦИОННЫЕ усилители и компараторы [Справ.]. – М.: «Додэка-XXI», 2002. – 560 с., – ISBN 5-94120-004-8.
20. ОСНОВЫ теории цепей [Учеб. для вузов]. / Г. В. Зевеке и др., 4-е изд., перераб., – М.: «Энергия», 1975. – 752 с.
21. ПАВЛОВСКАЯ, Т. А., C/C++. Программирование на языке высокого уровня [Учеб.]. –СПб.; «Питер», 2003. – 461 с., ISBN 5-94723-568-4.
22. ПОТЕМКИН, И. С., Функциональные узлы на потенциальных элементах [Текст]. / И. С. Потемкин, – М.: «Энергия», 1976. – 104 с.
23. ПОТЕМКИН, И. С., Функциональные узлы цифровой автоматики [Текст]. / И. С. Потемкин, – М.:, «Энергоатомиздат», 1988. – 320 с.
24. САВЕЛЬЕВ, А. Я., Основы информатики [Учеб. для вузов]. / А. Я. Савельев. – М.: «Изд. МГТУ им. Баумана», 2001. – 328 с. – ISBN 5-7038-1515-0.
25. СЕМЕНОВ, Б. Ю. Шина I2C в радиотехнических конструкциях. – М.: «СОЛОН-Р», 2002. – 192 с. – ISBN 5-93455-168-X.
26. СМИТ, С., Цифровая обработка сигналов [Практ. рук.] / пер. с англ. А. Ю. Линовича, С. В. Витязева, И. С. Гусинского. – М.: «Додэка-XXI», 2008. – 720 с. – ISBN 978-5-94120-145-7.
27. СТАШИН, В. В., Проектирование цифровых устройств на однокристалльных микроконтроллерах / В. В. Сташин, А. В. Уру-

- сов, О. Ф. Мологонцева. – М.: «Энергоатомиздат», 1990. – 224 с. – ISBN 5-283-01543-2
28. СТЕПАНЕНКО, И. П., Основы теории транзисторов и транзисторных схем [Текст] / И. П. Степаненко. – М.: «Энергия», 1977. – 672 с.
29. СТЕШЕНКО В. Б. ПЛИС фирмы Altera: элементная база, система проектирования, и языки описания аппаратуры. [Текст] / В. Б. Стешенко. – М.: «Додэка-XX1». 2007. – 576 с. – ISBN 978-594120-112-9.
30. УГРЮМОВ, Е. П., Цифровая схемотехника [Учеб. пос. для вузов] / Е. П. Угрюмов. – СПб.: «БХВ-Петербург», 2005. – 800 с. – ISBN 5-94157-397-9.
31. ХАРТОВ, В. Я., Микроконтроллеры AVR [Практикум] / В. Я. Хартов. – М.: «Изд. МГТУ им. Баумана», 2007. – 240 с. – ISBN 978-5-7038-3051-2
32. SDCC – Small Device C Compiler [электронный ресурс]: Режим доступа: <http://sdcc.sourceforge.net/>, свободный. – Яз. англ.

Книги издательства «ДМК Пресс» можно заказать в торгово-издательском холдинге «АЛЬЯНС БУКС» наложенным платежом, вылав открытку или письмо по почтовому адресу: **123242, Москва, а/я 20** или по электронному адресу: **orders@alians-kniga.ru**.

При оформлении заказа следует указать адрес (полностью), по которому должны быть высланы книги; фамилию, имя и отчество получателя. Желательно также указать свой телефон и электронный адрес.

Эти книги вы можете заказать и в Internet-магазине: **www.alians-kniga.ru**.

Оптовые закупки: тел. **(499) 725-50-27, 725-54-09**.

Электронный адрес **books@alians-kniga.ru**.

Аверченков Олег Егорович

Схемотехника: аппаратура и программы

Главный редактор *Мовчан Д. А.*
dm@dmk-press.ru

Корректор *Синяева Г. И.*

Верстка *Паранская Н. В.*

Дизайн обложки *Мовчан А. Г.*

Подписано в печать 17.05.2012. Формат 60×90 ¹/₁₆.

Гарнитура «Петербург». Печать офсетная.

Усл. печ. л. 36,75. Тираж 500 экз.

№

Web-сайт издательства: www.dmk-press.ru

Схемотехника: аппаратура и программы

- Материал дозирован, снабжен большим количеством внутренних ссылок, что облегчает самостоятельное изучение и чтение “с середины”.
- Упор сделан на возможном сочетании аппаратных и программных средств при реализации заданных функций.
- Рассмотрены не только принципы действия подобных устройств, но и приведены конкретные примеры их реализации, что позволяет рекомендовать данное пособие для курсового проектирования.
- В примерах программ отсутствуют неименованные константы и связанные с ними громоздкие комментарии.
- В книге нет справочных данных, за ними следует обращаться к справочной и фирменной документации.

В основу книги положены материалы лекций, читаемых автором в Смоленском филиале Московского энергетического института студентам специальности “Вычислительные машины, комплексы, системы и сети” и “Автоматизированные системы обработки информации”.

Книгу можно рекомендовать студентам вузов, для которых схемотехника не является объектом специализации. Она также может быть полезна инженерам и радиолюбителям, работающим в области создания и эксплуатации программно-управляемых систем.

Internet-магазин:

www.aliants-kniga.ru

Книга – почтой:

e-mail: orders@aliants-kniga.ru

Оптовая продажа:

«Альянс-книга»

Тел./факс: (499) 725-5409

e-mail: books@aliants-kniga.ru



ISBN 978-5-94074-402-3



9 785940 744023 >