МОДУЛЬНОЕ ПРОГРАММИРОВАНИЕ СИСТЕМ УПРАВЛЕНИЯ НА MCS48

Д. РЫЖОВ, г. Владимир

Известно, что один и тот же микроконтроллер может управлять как сложным технологическим оборудованием, так и бытовой кофемолкой или электронными часами. Адаптация к конкретному объекту осуществляется изменением программы микроконтроллера, аппаратные средства почти не затрагиваются. Предлагаемая статья посвящена приемам программирования микроконтроллеров серии MCS48, широко используемых в системах управления различного назначения. Ее основные положения справедливы и для более современных приборов.

Разработка и модернизация программ управления значительно облегчаются, если строить их по модульному принципу. В этом случае после накопления некоторого опыта, а главное – собственной библиотеки отлаженных модулей, программирование новой системы управления (СУ) сводится к замене некоторых модулей уже действующей и отлаженной программы и, возможно, дополнении ее фрагментами, учитывающими особенности конкретной системы.

Этот принцип заложен в структуру многих языков высокого уровня (PASCAL, C++), и программист буквально вынужден ему следовать. К сожалению, АССЕМБЛЕРы (в том числе для MCS48), предоставляя программисту большую свободу выбора средств и методов решения задач, как правило, совсем не следят за соблюдением дисциплины программирования. Это нередко приводит к созданию настолько запутанных программ, что даже их авторы не могут спустя некоторое время разобраться в том, что было сделано, не говоря уже об использовании отлаженных фрагментов в других программах. Сознательное соблюдение общих модульных концепций значительно облегчает и ускоряет программирование микроконтроллеров. Пример типичной модульной программы для СУ приведен в таблице. Ее синтаксис соответствует табличному АССЕМБЛЕРу TASM в варианте для микропроцессора 8048.

Как видно, в начале текста программы директивами EQU константам даются имена и присваиваются значения. Пользоваться именованными константами всегда предпочтительнее, чем указывать числовые значения непосредственно в исполняемых командах процессора. Например, выдержка времени, реализуемая одной из рассматриваемых ниже подпрограмм, определяется тремя числами. Они заданы константами N1, N2 и N3. Если нужно изменить выдержку, достаточно в операторах EQU указать новые значения. В противном случае пришлось бы разыскивать во всей программе команды с операндами, равными этим числам, решать, относится ли каждая из них к выдержке времени, и в нужных случаях указывать новые значения.

Очевидно, такая работа требует много времени и часто не обходится без ошибок. Особенно усложняет ее то, что в некоторых командах может использоваться не число целиком, а, например, его старший или младший байт. АССЕМБЛЕР уже на этапе трансляции программы способен вычислить некоторые константы, исходя из значений других. Эту возможность иллюстрирует вычисление старшего (N3H) и младшего (N3L) байтов числа N3.

Далее в программе выделяют память для переменных. Делают это теми же самыми директивами EQU, но в отличие от описаний констант задают не числовые значения переменных, а адреса занимаемых ими ячеек памяти.

Если позволяет АССЕМБЛЕР, не следует пренебрегать возможностью использования макрокоманд. Каждая из них представляет собой как бы новую команду, выполняющую операцию, прямо не предусмотренную системой команд процессора. Описывая макрокоманду, программист дает ей имя (конечно, не совпадающее с именем ни одной из «настоящих» команд) и задает требуемые действия в виде последовательности машинных команд. Каждый раз, встретив макрокоманду в программе, АССЕМБЛЕР заменит ее указанной последовательностью. В рассматриваемом примере используются две макрокоманды. Одна из них пересылает содержимое аккумулятора в заданную параметром макрокоманды ячейку памяти данных, а другая – обратно.

После включения питания (или подачи сигнала сброса) микроконтроллер начинает выполнять программу с нулевого адреса. По этому адресу обычно записывают команду безусловного перехода на действительную точку начала программы (в данном случае, на метку START). Это необходимо потому, что аппаратные прерывания всегда передают управление по фиксированным адресам 3 и 7 (у микроконтроллеров других типов адреса могут быть иными, но все равно они расположены в начале памяти программ). Находящиеся по этим адресам команды безусловного перехода на подпрограммы обслуживания соответствующих прерываний основная программа должна «обойти».

Следующий этап — установка режимов работы контроллера (например, выбор банков памяти и регистров), инициализация переменных и внешних устройств. Типичная ошибка начинающих программистов — считать, что сразу после пуска программы переменные уже имеют какие-то определенные значения. Укреплению этого заблуждения способствует предусмотренное в некоторых языках высокого уровня (например, в BASIC) автоматическое присвоение всем переменным начального нулевого значения. В программах на языке АССЕМБЛЕРа (и многих других языках) программист должен сам позаботиться, чтобы до первого считывания значения переменной в отведенную ей ячейку памяти уже было что-нибудь записано. Хороший стиль программирования требует, чтобы начальные значения были присвоены переменным в самом начале работы программы. В данном случае это делает подпрограмма INIT.

Раздел инициализации внешних устройств обычно выглядит как поочередный вызов подпрограмм, каждая из которых приводит в исходное состояние одно из них (аналого-цифровой преобразователь, светодиодный индикатор, кнопочный пульт и т. п.) и может быть легко заменена при доработке и совершенствовании системы. Нередко эти же подпрограммы проверяют работоспособность устройств.

Далее большинство управляющих программ входит в бесконечно повторяющийся основной цикл, выполнение которого приостанавливается только для обработки прерываний. Цикл состоит из подпрограмм опроса клавиатуры и других датчиков, проверки флагов, выставляемых подпрограммами обработки прерываний (например, флага истечения заданного интервала времени или окончания работы аналого–цифрового преобразователя), обработки поступившей информации в соответствии с заданным алгоритмом управления, вывода управляющих воздействий на исполнительные устройства, вывода информации о состоянии технологического процесса на жидкокристаллическое табло или другие индикаторы. Выход из основного цикла обычно предусматривается только в аварийных ситуациях, например, если для ликвидации последствий сбоя необходимо повторить инициализацию всех переменных и внешних устройств, а также при обработке прерываний.

Таким образом, программа, построенная по модульному принципу, представляет собой набор подпрограмм. Если в новой СУ применена, например, иная клавиатура, достаточно будет заменить подпрограмму BUTT. Для того чтобы такая замена была простой и безболезненной, следует выработать и всегда соблюдать определенные правила. Подпрограммы, по возможности, должны сохранять содержимое всех регистров контроллера, получать исходные данные и выдавать результаты работы в одних и тех же регистрах и ячейках памяти, пользоваться одной и той же кодировкой символов и т. п.

Следует бороться с естественным (особенно для программистов, преодолевших первые трудности и начинающих чувствовать себя профессионалами) стремлением упростить программу за счет отхода от строгих правил и применения нестандартных приемов. Кажущееся, на первый взгляд, неоправданным усложнение вполне окупится облегчением отладки и переработки программы в целом.

Рассмотрим некоторые особенности подпрограмм. INCREM и DECREM выполняют требуемые во многих случаях операции увеличения или уменьшения на заданную величину 16-разрядного двоичного числа (его старший и младший байты находятся соответственно в регистрах R6 и R5). Константы, задающие величину приращения, описаны в начале программы.

Так как любой микроконтроллер работает значительно быстрее технологическо-

```
;ПРОГРАММА WAZ_NIVA.ASM. РЫЖОВ Д. И., г. Владимир


;Константы:

DELL    .EQU 1          ;Младший и старший байты 16-разрядного
DELH    .EQU 0          ;числа DEL.
DEL1L   .EQU 1          ;То же DEL1.
DEL1H   .EQU 0
N1      .EQU 13         ;Константы, определяющие заданную
N2      .EQU 6          ;выдержку времени.
N3      .EQU 1200
N3L     .EQU N3 & $FF
N3H     .EQU N3 / 256
................
................

;Переменные:

ACC1    .EQU 20H        ;Сохранение аккумулятора.
INTT    .EQU 21H        ;Тип прерывания по INT.
DS      .EQU 22H        ;Рабочие ячейки обработчика прерываний
IST     .EQU 23H        ;от таймера.
TR2     .EQU 24H
TR3     .EQU 25H
TR4     .EQU 26H
TIMEL   .EQU 27H
TIMEH   .EQU 28H
FLT     .EQU 29H        ;Флаг истечения времени.
................
................

;Макроопределения:

#DEFINE  MOVA<@R0(xx)  mov r0,xx\ mov a,@r0
#DEFINE  MOVA>@R0(xx)  mov r0,xx\ mov @r0,a
................
................

;Исполняемая программа.

        .ORG $000
        JMP  START      ;Обход векторов прерываний.
        .ORG $003
        JMP  INTER      ;Прерывание по сигналу INT.
        .ORG $007
        JMP TIME        ;Прерывание от таймера.

START:  SEL MB0         ;Начало программы.
        SEL RB0
        CALL INIT       ;Инициализация регистров и переменных.
        CALL SBRSD      ;Сброс (установка в исходное состояние)
        CALL SBRTAB     ;светодиодов, табло и других устройств.
        ..........
        ..........
        CALL SET2M      ;Задание выдержки времени.

;Основной цикл.

CYCLE:  CALL BUTT       ;Опрос клавиатуры и датчиков.
        CALL SENS
        CALL LED        ;Отображение светодиодами состояния
                        ;клавиш и датчиков.
        CALL ALGO       ;Реализация алгоритма управления.
        CALL ACTUAT     ;Выдача управляющих воздействий.
        CALL TABLO      ;Отображение состояния технологичес-
                        ;кого процесса на табло.
        ........        ;Другие действия.
        ........
        JMP CYCLE       ;Конец основного цикла.

;Подпрограммы:

;Увеличение 16-разрядного числа в регистрах R6, R5
;на величину DEL

INCREM: MOV A,R5
        ADD A,#DELL
        MOV R5,A
        MOV A,R6
        ADDC A,#DELH
        MOV R6,A
        CLR C
        RET

;Уменьшение 16-разрядного числа в регистрах R6, R5
;на величину DEL1

DECREM: MOV A,#DEL1L
        CPL A
        ADD A,#01H
        MOV R0,A
        MOV A,#DEL1H
        CPL A
        ADDC A,#00H
        MOV R4,A
        MOV A,R0
        ADD A,R5
        MOV R5,A
        MOV A,R6
        ADDC A,R4
        MOV R6,A
        CLR C
        RET

;Установка начальных условий для отсчета выдержки
;времени 2 мин.

SET2M:  MOV A,#0        ;Очистить рабочие ячейки.
        MOVA>@R0(#DS)
        MOVA>@R0(#IST)
        MOVA>@R0(#TIMEL)
        MOVA>@R0(#TIMEH)
        MOVA>@R0(#FLT)  ;и опустить флаг.
        MOV A,#N1       ;Начальное значение счетчика/таймера.
        MOVA>@R0(#TR4)
        MOV A,#N3L      ;Заданное число повторений выдержки
        MOVA>@R0(#TR2)  ;длительностью 0,1 с.
        MOV A,#N3H
        MOVA>@R0(#TR3)
        STRT T          ;Запустить таймер и разрешить
        EN TCNTI        ;прерывания.
        RET

;Обработка прерывания от таймера

TIME:   STOP TCNT       ;Остановить таймер и запретить
        DIS TCNTI       ;прерывания.
        MOVA<@R0(#TR4)  ;Перенести значения из ячеек памяти
        MOV R4,A        ;в регистры:
        MOVA<@R0(#TR2)  ;R4 - число, загружаемое в регистр
        MOV R2,A        ;таймера;
        MOVA<@R0(#TR3)  ;R3 и R2 - заданное число повторений.
        MOV R3,A
        MOVA<@R0(#TIMEL)
        MOV R5,A
        MOVA<@R0(#TIMEH);R6 и R5 - счетчик повторений.
        MOV R6,A
        CALL INCREM     ;Увеличить текущее значение счетчика.
        MOV A,R5
        MOVA>@R0(#TIMEL)
        MOV A,R6
        MOVA>@R0(#TIMEH)
        MOV A,R6        ;Если заданное число еще не достигнуто,
        XRL A,R3        ;вновь запустить таймер.
        JNZ NEWSTART
        MOV A,R5
        XRL A,R2
        JNZ NEWSTART
        MOV A,#01H      ;Иначе - время истекло!
        MOVA>@R0(#FLT)  ;Выставляем флаг FLT=1.
        JMP RETURN
NEWSTART:
        MOVA<@R0(#IST)
        INC A
        MOVA>@R0(#IST)  ;IST=IST+1
        XRL A,#N2
        JNZ ESE         ;Если IST<N2, иначе IST=0 и DS=DS+1.
        MOV A,#0
        MOVA>@R0(#IST)
        MOVA<@R0(#DS)
        INC A
        MOVA>@R0(#DS)
ESE:    MOV A,R4
        MOV T,A
        STRT T          ;Запустить таймер и разрешить
        EN TCNTI        ;прерывания.
RETURN: RETR

;Обработка прерывания по сигналу INT

INTER:  MOVA>@R0(#ACC1) ;Сохранить аккумулятор.
        MOVA<@R0(#INTT) ;Проверить тип прерывания.
        XRL A,#01H
        JZ M1
        MOVA<@R0(#INTT)
        XRL A,#02H
        JZ M2
        JMP OUTE        ;Неверное значение типа.
M1:     CALL ISR1       ;INTT=1, вызывать ISR1.
        JMP OUTE
M2:     CALL ISR2       ;INTT=2, вызывать ISR2.
OUTE:   MOVA<@R0(#ACC1) ;Восстановить аккумулятор.
        RETR

ISR1:   ........        ;Первый вариант обработки.
        ........
        RET

ISR2:   ........        ;Второй вариант обработки.
        ........
        RET

;Инициализация регистров и переменных

INIT:   ........
        ........
        RET

;Другие подпрограммы

        ........
        ........

;Тексты некоторых подпрограмм могут находиться в отдельных
;файлах. Их подключают директивами:

#INCLUDE "LED.ASM"
#INCLUDE "ACTUAT.ASM"
#INCLUDE "TABLO.ASM"
................
................

        .END
```

го оборудования, очень важно уметь организовывать в программе выдержку времени. В данном случае использован внутренний счетчик/таймер процессора. Он имеет ограниченную емкость и переполняется за время, измеряемое миллисекундами. Каждое переполнение генерирует запрос прерывания. Подпрограмма обслуживания прерываний от таймера (TIME) подсчитывает их и при достижении заданного числа присваивает единичное значение флагу истечения времени FLT. Всем подпрограммам, работа которых зависит от времени, остается анализировать состояние этого флага. Так удается реализовать выдержки в несколько секунд и даже минут.

Для того чтобы начать отсчет нового интервала, необходимо занести исходные значения в рабочие ячейки подпрограммы TIME и включить таймер. Подпрограмма SET2M, например, задает выдержку времени, равную 2 мин. Расчет исходных значений имеет несколько тонкостей.

Известно, что в микроконтроллерах серии MCS48 на вход внутреннего счетчика/таймера импульсы поступают с частотой, в 480 раз меньшей частоты кварцевого генератора. Например, при частоте кварцевого резонатора 7 МГц число, записанное в счетчик, изменяется каждые 480/7000000 = 0,00006857 с = 68,57 мкс. Так что счетчик переполнится (и будет сформирован запрос прерывания) через 68,57·(256–N1) мкс, где N1 — число, первоначально записанное в счетчик. Если каждый раз начинать новый счет с этого числа, то за 0,1 с (минимальная выдержка времени) произойдет N2 = 0,1·7000000/[480·(256–N1)] переполнений.

Очевидно, одну и ту же выдержку времени можно получить при разных N1 и N2, но так как эти числа не могут быть дробными, она будет реализована с некоторой ошибкой. Задача состоит в подборе такой пары значений, при которых ошибка минимальна. В рассматриваемом случае наилучший вариант N1 = 13, N2 = 6. Выдержка времени, равная 2 мин, получается повторением описанной процедуры N3 = 1200 раз.

Часто бывает необходимо в разных режимах работы программы применять разные процедуры обработки одних и тех же аппаратных прерываний. Один из способов сделать это иллюстрирует подпрограмма INTER. Она анализирует код типа прерывания, занесенный основной программой в ячейку INTT, и в зависимости от его значения вызывает одну из подпрограмм обслуживания прерывания ISR1 или ISR2. Заметим, что обе они заканчиваются командой RET, а не RETR. Число вариантов обработки нетрудно увеличить и даже сделать так, что при некотором значении кода будут вызываться одна за другой несколько различных подпрограмм.

Вовсе не обязательно записывать все необходимые подпрограммы в текстовый файл основной программы. Отлаженные и неоднократно использованные в разных программах модули могут находиться в отдельных файлах и подключаться к основной программе директивами INCLUDE. Каждый включаемый файл может содержать одну или несколько подпрограмм. Недостаток такого способа заключается в том, что имена переменных, констант и меток во всех используемых модулях не должны повторяться. Лишенный этого дефекта метод раздельной трансляции модулей с последующим объединением их на уровне объектного кода, к сожалению, не поддерживается АССЕМБЛЕРом TASM. ■

МИКРОКОНТРОЛЛЕРЫ 8XC51GB ФИРМЫ INTEL

А. ФРУНЗЕ, г. Москва

СИСТЕМА ПРЕРЫВАНИЙ

Микроконтроллеры 8xC51GB поддерживают 15 векторов прерываний (табл. 6). Младшие пять из них аналогичны имеющимся во всех контроллерах семейства MCS51, шестой обслуживает третий таймер/счетчик (он появился только начиная с кристаллов семейства MCS52), седьмой, имеющийся только в 8xC51FX, 8xL51FX и 8xC51GB, поддерживает программируемую матрицу счетчиков (PCA). Последний дополнительно располагает прерываниями от пяти внешних входов (INT2—INT6), второй матрицы программируемых счетчиков, АЦП и расширенного последовательного порта.

Таблица 6

Адрес	Флаг-источник	Аппаратный сброс	Назначение
0003H	IE0	нет(уров.)/да(фронт)	Внешнее событие на входе INT0
000BH	TF0	да	Переполнение T/C0
0013H	IE1	нет(уров.)/да(фронт)	Внешнее событие на входе INT1
001BH	TF1	да	Переполнение T/C1
0023H	RI+TI	да	Прием/передача по последовательному каналу
002BH	TF2+EXF2	нет	Прерывание от T/C2
0033H	CF, CCFn (n=0-4)	нет	Прерывание от PCA
003BH	AIF	нет	Прерывание от АЦП
0043H	C1F, C1CFn (n=0-4)	нет	Прерывание от PCA1
004BH	SEPIF	нет	Прерывание от расширенного последовательного порта
0053H	IE2	да	Внешнее событие на входе INT2
005BH	IE3	да	Внешнее событие на входе INT3
0063H	IE4	да	Внешнее событие на входе INT4
006BH	IE5	да	Внешнее событие на входе INT5
0073H	IE6	да	Внешнее событие на входе INT6

Во всех контроллерах семейства MCS51 каждое прерывание может быть запрещено установкой в низкий уровень соответствующего бита в регистре IE. Естественно, это справедливо и для 8xC51GB. Однако поскольку он содержит вдвое больше источников прерывания, то для их разрешения/запрещения используется дополнительный регистр IEA (табл. 7). Как и в предыдущем случае, установка бита в 1 приводит к разрешению соответствующего прерывания, сброс в 0 запрещает его. Адрес регистра IEA — 0A7H.

Таблица 7

Бит	Позиция	Что разрешает/запрещает
EAD	IEA.7	Прерывание от АЦП
EX6	IEA.6	Внешн. прерывание - вход INT6
EX5	IEA.5	Внешн. прерывание - вход INT5
EX4	IEA.4	Внешн. прерывание - вход INT4
EX3	IEA.3	Внешн. прерывание - вход INT3
EX2	IEA.2	Внешн. прерывание - вход INT2
EC1	IEA.1	Прерывание от PCA1
ESEP	IEA.0	Прерывание от SEP

Отметим, что все прерывания, в том числе и описанные в табл. 7, могут быть одновременно запрещены установкой в 0 бита EA (IE.7) — старшего бита регистра IE.

Каждое прерывание может иметь свой собственный приоритет (от уровня 0 — низшего, до уровня 3 — высшего). Уровень приоритета определяется состоянием бит в регистровых парах IP, IPH и IPA, IPHA. Первая из них идентична имеющейся в более ранних контроллерах и подробно описана при рассмотрении группы 8xC51Fx. Вторая пара (адреса регистров соответственно 0B8H и 0B6H) имеется только в 8xC51GB и обслуживает прерывания, которые есть только в этих контроллерах. В табл. 8 показано соответствие между битами регистров и прерываниями, уровень которых они определяют, в табл. 9 — соответствие между уровнями приоритетов и состоянием бит в регистровых парах IP, IPH и IPA, IPHA.

Прерывания с низким приоритетом могут быть, в свою очередь, прерваны только событием более высокого приоритета (но не равного). Соответственно прерывание с высшим приоритетом прервано быть не может. Если процессор одновременно получил запросы на два или более прерываний с одинаковым приоритетом, то очередность их обработки определяется специальной последовательностью опроса флагов прерывания. У контроллеров 8xC51GB она выглядит следующим образом:

1 (высший)	IE0
2	SEPIF
3	IE2
4	TF0
5	C1F, C1CFn
6	IE3
7	IE1
8	AIF
9	IE4
10	TF1
11	CF, CCFn
12	IE5
13	RI, TI
14	TF2, EXF2
15 (низший)	IE6

Внешние прерывания INT0 и INT1 микроконтроллера 8xC51GB полностью соответствуют аналогичным прерываниям всех микросхем семейства MCS51 и в зависимости от состояния бит IT0 и IT1 регистра TCON могут фиксироваться как по уровню, так и по перепаду из 1 в 0.

Внешние выводы INT2 и INT3 могут реагировать как на положительный, так и на отрицательный фронт сигнала. Микросхема имеет регистр EXICON (0C6H), содержащий биты IT2 и IT3, определяющие активный фронт сигнала на выводах P5.2(INT2) и P5.3(INT3). При установке бита ITn в 0 преры-

(Окончание. Начало см. в «Радио», 1998, № 2.)

вание инициируется по отрицательному фронту, при ITn=1 — по положительному. Внешние события INT4—INT6 фиксируются только по положительному фронту на выводах P5.4(INT4)—P5.6(INT6).

Таблица 8

Младший бит	Позиция	Старший бит	Позиция	Регулируемое прерывание
PAD	IPA.7	PADH	IPHA.7	Прерывание от АЦП
PX6	IPA.6	PX6H	IPHA.6	Внешнее прерывание - вход INT6
PX5	IPA.5	PX5H	IPHA.5	Внешнее прерывание - вход INT5
PX4	IPA.4	PX4H	IPHA.4	Внешнее прерывание - вход INT4
PX3	IPA.3	PX3H	IPHA.3	Внешнее прерывание - вход INT3
PX2	IPA.2	PX2H	IPHA.2	Внешнее прерывание - вход INT2
PC1	IPA.1	PC1H	IPHA.1	Прерывание от PCA1
PSEP	IPA.0	PSEPH	IPHA.0	Прерывание от SEP

Все внешние прерывания генерируют соответствующие аппаратно устанавливаемые флаги. Для событий INT0, INT1 — это биты IE0 и IE1 регистра TCON. Флаги IE2—IE6 находятся в регистре EXICON. Их сброс осуществляется аппаратно в момент перехода процессора на подпрограмму обработки соответствующего прерывания.

За время машинного цикла опрос выводов

Таблица 9

IPH.x (IPHA.x)	IP.x (IHA.x)	Уровень приоритета прерывания
0	0	0
0	1	1
1	0	2
1	1	3

внешних прерываний осуществляется лишь однажды. Поэтому для того чтобы прерывание было зарегистрировано, длительность его активного уровня должна превышать продолжительность одного машинного цикла (12 периодов тактового генератора). Назначение бит регистра EXICON приведено в табл. 10.

Таблица 10

Бит	Позиция	Функция
-	EXICON.7	Зарезервировано
IE6	EXICON.6	Флаг прерывания INT6
IE5	EXICON.5	Флаг прерывания INT5
IE4	EXICON.4	Флаг прерывания INT4
IE3	EXICON.3	Флаг прерывания INT3
IE2	EXICON.2	Флаг прерывания INT2
IT3	EXICON.1	Бит управления INT3. При IT3=0 активный фронт прерывания - отрицательный, при IT3=1 - полохительный
IT2	EXICON.0	Бит управления INT2. При IT2=0 активный фронт прерывания - отрицательный, при IT2=1 - полохительный

РАСШИРЕННЫЙ ПОСЛЕДОВАТЕЛЬНЫЙ ПОРТ

Расширенный последовательный порт (SEP) располагает аппаратными средствами для реализации шины I C-bus, де-факто являющейся стандартом последовательного обмена. SEP допускает функционирование в четырех различных режимах, имеет три различных источника тактирования. Под его нужды задействовано два вывода микросхемы: P4.1 — ввода/вывода данных и P4.0 — для вывода тактирующего сигнала. Передаваемый или принимаемый пакет состоит из восьми бит данных. При этом используется восемь тактов работы SEP. В отсутствие принимаемой или передаваемой информации тактовый сигнал и данные неактивны.

За SEP закреплены три SFR-регистра: SEPCON (0D7H), SEPDAT (0E7H) и SEPSTAT (0F7H). Адресуются они только побайтно. Назначение бит в регистрах SEPCON и SEPSTAT приведено в табл. 11 и 12 соответственно.

На рис. 2 показаны отличительные особенности режимов работы SEP — активные уровни сигнала тактирования и фронты, используемые для приема или передачи. Как следует из табл. 11, режим работы SEP определяется состоянием бит CLKPOL и CLKPH, расположенных в регистре SEPCON.

Таблица 11

Бит	Позиция	Функция
-	SEPCON.7	Зарезервирован
-	SEPCON.6	Зарезервирован
SEPE	SEPCON.5	Разрешение SEP
SEPREN	SEPCON.4	Разрешение приема в SEP
CLKPOL	SEPCON.3	Полярность тактирования SEP
CLKPH	SEPCON.2	Фаза тактирования SEP
SEPS1	SEPCON.1	Бит 1 выбора скорости SEP
SEPS0	SEPCON.0	Бит 0 выбора скорости SEP
CLKPOL	CLKPH	Режим SEP
0	0	SEPMODE0
0	1	SEPMODE1 (только передача)
1	0	SEPMODE2
1	1	SEPMODE3 (только передача)
SEPS1	SEPS0	Скорость SEP
0	0	Fosc/12
0	1	Fosc/24
1	0	Fosc/48
1	1	Fosc/96

Таблица 12

Бит	Позиция	Функция
-	SEPSTAT.7	Зарезервирован
-	SEPSTAT.6	Зарезервирован
-	SEPSTAT.5	Зарезервирован
-	SEPSTAT.4	Зарезервирован
-	SEPSTAT.3	Зарезервирован
SEPFWR	SEPSTAT.2	Флаг ошибки записи SEP
SEPFRD	SEPSTAT.1	Флаг ошибки чтения SEP
SEPIF	SEPSTAT.0	Флаг прерывания SEP

Для приема или передачи байта пользователь должен выбрать режим работы порта (биты CLKPOL и CLKPH), скорость передачи (SEPS1 и SEPS0) и установить в 1 бит SEPE. Процесс передачи начинается сразу после загрузки байта в регистр SEPDATA. Прием инициируется установкой в 1 бита SEPREN в случае, когда регистр SEPDATA пуст и нет передачи. После приема восьми бит SEPREN аппаратно сбрасывается. Завершение приема или передачи приводит к установке в 1 бита SEPIF. Его сброс возможен только программным путем.

Если пользователь предпримет попытку записать информацию в регистр SEPDATA (или прочитать ее из него) в момент передачи или приема, устанавливается соответствующий бит ошибки. Флаг SEPFWR устанавливается при попытке сделать это в процессе передачи байта, а SEPFRD — в процессе приема. Прерывания, связанные с установкой этих бит отсутствуют, вследствие чего пользователь должен контролировать их состояние самостоятельно. Естественно, сброс этих флагов может быть осуществлен только программным путем.

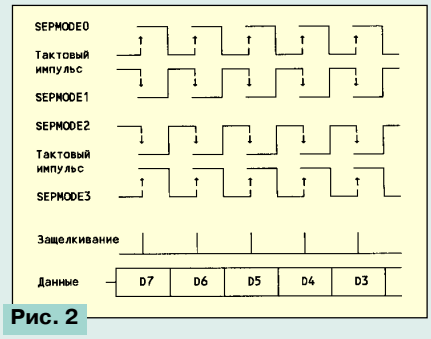

Рис. 2

АППАРАТНЫЙ СТОРОЖЕВОЙ ТАЙМЕР

Аппаратный сторожевой таймер (HWDT) сбрасывает микроконтроллер при своем переполнении, что является средством борьбы с зависанием системы (зацикливанием программы). На выполнение аналогичной функции может быть настроен и таймер/счетчик модуля 4 PCA, но такое его применение ограничивает возможности пользователя, в связи с чем в 8xC51GB появился самостоятельный WDT, не требующий использования PCA.

Аппаратный сторожевой таймер состоит из 14-битного счетчика, инкрементируемого в каждом машинном цикле, и SFR-регистра WDTRST (0A6H). Таймер всегда активен и при работающем тактовом генераторе непрерывно увеличивает содержимое счетчика. Средств остановки таймера нет. Если программа пользователя не записывает в WDTRST никакой информации, то через каждые 16 384 машинных цикла HWDT формирует сигнал RESET, который сбрасывает микроконтроллер. При этом счетчик обнуляется. Для предупреждения срабатывания HWDT пользовательская программа с промежутком не реже 16 383 машинных циклов должна заносить в регистр WDTRST последовательно два байта — 01EH и 0A6H. Отметим, что в WDTRST можно только записывать информацию, средства чтения его содержимого отсутствуют.

Не рекомендуется производить упомянутый перезапуск сторожевого таймера с помощью подпрограммы обработки прерывания от одного из таймеров/счетчиков, поскольку прерывания могут обрабатываться и при зависшей основной программе. Лучшее место для расположения команд обнуления сторожевого таймера — циклически выполняемый программный фрагмент, период повторения которого меньше времени срабатывания HWDT.

При переводе 8xC51GB в режим микропотребления внутренний тактовый генератор и HWDT останавливаются. Выведение контроллера из режима микропотребления, как и у всех его предшественников, может быть осуществлено двояко: сбросом или вызовом внешнего прерывания, разрешенного перед переводом 8xC51GB в названный режим. В

первом случае обнуляется HWDT, во втором при старте тактового генератора содержимое счетчика HWDT продолжит увеличиваться. Но поскольку для устойчивого запуска тактового генератора необходимо время около двух десятков его периодов, рекомендуется длительность импульса внешнего прерывания, выводящего контроллер из режима микропотребления, делать не меньшей упомянутого времени. Программа обработки прерывания начнет выполняться только после перехода уровня сигнала внешнего прерывания в 1, когда частота генерации стабилизируется. Тогда же и начнется инкрементирование счетчика HWDT, т. е. пока сигнал прерывания имеет нулевой уровень, HWDT не работает.

В режиме XX тактовый генератор контроллера не отключается. Вследствие этого содержимое счетчика HWDT непрерывно увеличивается и для предотвращения пересброса необходимо использовать таймерное прерывание, по которому будут осуществляться выход из этого режима, обнуление счетчика сторожевого таймера и возврат в режим Idle.

Ниже приводится фрагмент кода, использующего прерывание от T/C0 для периодического сброса HWDT. Правда, как отмечалось выше, использование такого прерывания — не лучшее место для обнуления счетчика, и подобную процедуру лучше встроить в периодически выполняемую часть программы — опрос клавиатуры или отображение информации. Поэтому приводимый фрагмент следует рассматривать как демонстрационный пример, а не как подпрограмму, которую нужно использовать в программах без каких-либо изменений.

```
; ПРОЦЕДУРА ИНИЦИАЛИЗАЦИИ T/C0
;
TC0_INIT:       SETB    EA              ;РАЗРЕШАЕМ ПРЕРЫВАНИЯ
                SETB    ET0             ;В ТОМ ЧИСЛЕ ОТ T/C0
                MOV     TMOD,#01H       ;УСТАНАВЛИВАЕМ РЕЖИМ 16-БИТНОГО ТАЙМЕРА
                MOV     TL0,#07FH       ;ЗАГРУЖАЕМ 0C17FH=0FFFH-03E80H, ЧТО
                MOV     TH0,#0C1H       ;ОБЕСПЕЧИТ СБРОС ЧЕРЕЗ 16000(03E80H) ТАКТОВ
                SETB    TR0             ;ЗАПУСКАЕМ ТАЙМЕР
                RET
;
; ОБРАБОТЧИК ПРЕРЫВАНИЙ ОТ T/C0
;
                ORG     000BH           ;АДРЕС ВЕКТОРА ПРЕРЫВАНИЯ
;
                CLR     TR0             ;ПРИОСТАНОВИМ ТАЙМЕР
                MOV     WDTCON,#01EH    ;СБРОСИМ HWDT
                MOV     WDTCON,#0E1H
                MOV     TL0,#07FH       ;СНОВА ЗАГРУЖАЕМ 0C17FH
                MOV     TH0,#0C1H
                SETB    TR0             ;ПЕРЕЗАПУСКАЕМ ТАЙМЕР
                RETI
;
```

ОБНАРУЖЕНИЕ СБОЯ ТАКТОВОГО ГЕНЕРАТОРА

Цепь обнаружения сбоя тактового генератора (OFD) предназначена для сброса микроконтроллера, если частота генератора окажется ниже предельного значения, заданного техническими условиями. Если после сброса тактовая частота не изменится (вернее, не возрастет до допустимого значения), контроллер так и останется в этом состоянии. Отметим, что превышение частоты сверх установленной границы не приводит к его сбросу.

Цепь OFD всегда включается после сброса или при выходе контроллера из режима микропотребления. Для ее отключения необходимо записать последовательно 0E1H и 01EH в регистр OSCR (0A5H). Это необходимо сделать, в частности, перед переходом в режим микропотребления, поскольку в нем тактовый генератор выключен. Разрешить работу цепи заново можно лишь пересбросом или выходом из режима микропотребления по внешнему прерыванию.

Состояние цепи OFD может быть определено путем чтения регистра OSCR. При OSCR=0FFH обнаружение сбоев разрешено, при OSCR=0FEH — запрещено.

ЗАКЛЮЧЕНИЕ

Итак, мы завершили рассмотрение особенностей восьмиразрядных микроконтроллеров семейства MCS51, разработанных и выпускаемых фирмой Intel. Они оказались настолько удачными, что тиражирование многих из них (с некоторыми технологическими усовершенствованиями) продолжается и поныне. Устойчивый спрос на эти контроллеры определяется тем, что сотни тысяч разработчиков привыкли к ним, наработали огромный объем программного обеспечения, обзавелись парком отладочных и кросс-средств. Во многих случаях новая разработка не требует замены микроконтроллера на что-то кардинально новое, в связи с чем целесообразнее выполнить ее на том, что уже знакомо и обеспечено средствами поддержки, а не тратить силы и средства на переход к иной элементной базе.

По этой причине Intel регулярно усовершенствовала свои контроллеры, чтобы расширить круг решаемых с их использованием задач. Более того, к подобному усовершенствованию присоединились фирмы, не имевшие отношения к первоначальной разработке. Так, сегодня совместимые с этим семейством микроконтроллеры выпускают фирмы Philips, Siemens, Dallas Semiconductor, Atmel, OKI и некоторые менее известные производители, в том числе и ряд предприятий на территории бывшего СССР. Все контроллеры имеют одинаковые набор команд и базовую архитектуру, как правило, совместимы по «цоколевке» и имеют схожие алгоритмы программирования.

Однако есть и существенные различия в наборе дополнительных регистров и аппаратных средств. Так, микроконтроллеры фирмы Dallas Semiconductor имеют два регистра DPTR и механизм их переключения, изделия Philips — АЦП повышенной разрядности, у контроллеров Siemens на кристалле нередко расположена внешняя память, адресуемая командами MOVX, и т. д. Со многими из этих особенностей автор планирует познакомить читателей в последующих статьях.

ПРИНЦИПИАЛЬНАЯ СХЕМА ПРОГРАММАТОРА

Ввиду того что прибор состоит из большого числа одинаковых узлов, не будем приводить целиком его принципиальную схему. Ограничимся лишь описанием схем и работы его основных блоков, а также порядка взаимодействия их друг с другом.

Uniprog подключают к порту принтера LPT1 компьютера. Необходимые для программирования данные поступают в блок регистров прибора, выполненный на микросхемах KP580BB55A. Все порты этих микросхем (за исключением одного, о котором будет сказано ниже) настраиваются на вывод. Выходы одних регистров соединены с управляющими входами многофункционального коммутатора, других — с аналогичными входами источников постоянного напряжения. Выходы коммутатора и источников соединяются в нужном порядке с выводами программируемой микросхемы. Таким образом, имеется возможность по командам компьютера формировать на этих выводах любые необходимые для программирования последовательности уровней напряжения.

Принципиальная схема узла связи блока регистров с компьютером показана на рис.1 (позиционные обозначения элементов на этой и последующих схемах условны). Для обеспечения нужного порядка обмена данными многие цепи LPT1 использованы нестандартно. Исключение составляют DATA1—DATA8, по которым через формирователь DD2 коды из компьютера поступают на шину данных блока регистров (цепи D0—D7). В какой именно порт и какой микросхемы KP580BB55A будет записана эта информация, зависит от кода, предварительно занесенного в регистр адреса DD5. Выходы двух младших разрядов этого регистра соединены с входами A0 и A1 микросхем KP580BB55A, а каждый из старших — с входом CS одной из них. Сигнал записи в DD5 подается по цепи $\overline{\text{AUTOFD}}$, а в порты KP580BB55A — по цепи $\overline{\text{INIT}}$.

Входы порта KP580BB55A, настроенного на ввод, соединены с шиной данных программируемой микросхемы, что позволяет прочитать записанный в нее код и сравнить его с требуемым. Цепи DATA1—DATA8 однонаправлены и не могут быть использованы для чтения. Поэтому компьютер читает байт, выведенный на шину данных блока регистров под воздействием сигнала $\overline{\text{SLCTIN}}$, в два приема по четыре разряда. С помощью мультиплексора DD1, управляемого сигналом $\overline{\text{STROBE}}$, они поочередно подключаются к цепям SLCT, PE, $\overline{\text{ACKNLG}}$ и BUSY, по которым компьютер обычно получает сигналы состояния принтера.

Аналогичным образом через буферный элемент DD6 можно прочитать и состояние восьми младших разрядов шины адреса программируемой микросхемы. Это бывает необходимо, если она имеет 16-разрядную шину данных либо мультиплексированную шину адреса/данных. Работа DD6 разрешается записью логического 0 во второй разряд DD5.

Многофункциональный коммутатор состоит из узлов двух типов. Для управ-

УНИВЕРСАЛЬНЫЙ ПРОГРАММАТОР UNIPROG

А. ЖАРОВ, г. Москва

Подключив программатор Uniprog к IBM-совместимому компьютеру через разъем принтера, можно заносить данные не только в обычные ПЗУ или память программ микроконтроллеров, но и в микросхемы программируемых логических матриц (ПЛМ). Программное обеспечение (оно названо Uniprog Plus) построено по принципу открытой архитектуры. Владея языком Си и применяя встроенные функции ядра Uniprog Plus, можно дополнить его своими собственными программирующими или тестирующими модулями.

ления шиной данных программируемой микросхемы имеется восемь коммутаторов, собранных по схеме, изображенной на рис.2, а. При нулевом уровне не входе УПР1 в зависимости от сигнала УПР2 на соответствующий разряд шины данных с выхода коммутатора подается напряжение одного из логических ТТЛ-уровней. Однако когда на входы УПР1 и УПР2 подана логическая 1, коммутируемая цепь через открывшийся транзистор VT1 соединяется с программируемым источником постоянного напряжения E. Диод VD2, закрываясь при значении E, меньшем напряжения питания +5 В, защищает транзистор VT1 от протекания тока в обратном направлении. В свою очередь, диод VD1 защищает цепь СЧИТЫВАНИЕ от напряжений, больших 5 В. В узле применен мощный транзистор КТ973А, способный пропустить импульсный ток до 1 А, что необходимо, например, для программирования микросхем серий К556, К1556.

Для управления шиной адреса и большинством других выводов программируемой микросхемы таких больших токов не требуется. Поэтому узел их коммутации (всего таких узлов — 20) несколько проще (рис. 2, б). Если на входы УПР1 и УПР2 одновременно подать напряжения с уровнем логического 0, откроются и VT1, и внутренний выходной транзистор элемента D1.2, но резистор R3 ограничит ток и не допустит повреждения транзисторов. Цепи СЧИТЫВАНИЕ с элементами VD1 и R4 имеются только в коммутаторах восьми младших разрядов шины адреса.

Четыре программируемых источника напряжений E1—E4 собраны по схеме, изображенной на рис. 3. Напряжение E1 через коммутаторы поступает на шины адреса и данных, остальные три можно подать на любые другие выводы программируемой микросхемы, в том числе и на вывод питания.

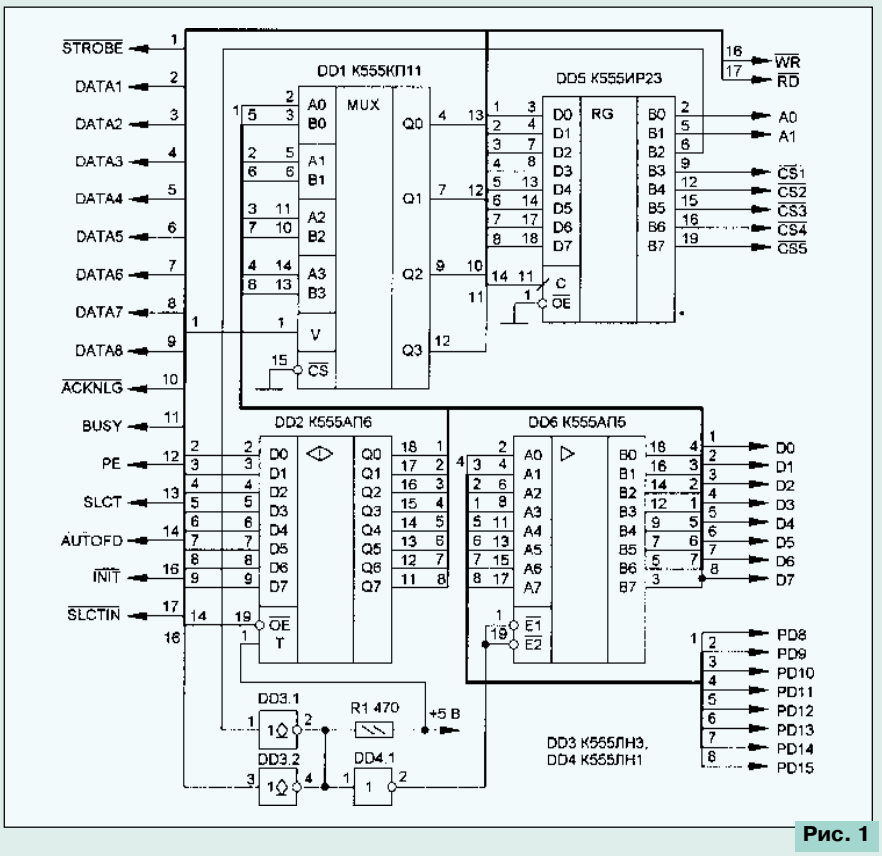

Рис. 1

Цифро-аналоговым преобразователем (ЦАП) на микросхеме DD2, включенной нестандартно, управляет код, поступающий из блока регистров. Сигнал УПР2 включает и выключает ЦАП, а УПР1 подключает к его выходу конденсатор C1, обеспечивающий плавное нарастание выходного напряжения после включения ЦАП или скачкообразного изменения кода (иногда это необходимо для правильного программирования). Источник образцового (опорного) напряжения и напряжения питания на стабилитронах VD1 и VD2 — общий для всех ЦАПов.

Напряжение от ЦАПа поступает на выход источника через усилитель мощности, выполненный на ОУ DA1 и транзисторах VT1—VT3. Последние должны иметь граничную частоту не менее 20 МГц, что необходимо для нормального функционирования обратной связи (а значит, стабильности выходного напряжения) в условиях переменной нагрузки, возникающей при работе с некоторыми микросхемами. Например, значения тока, потребляемого микросхемами ППЗУ серии К556, существенно различаются при чтении ячеек, в которые записаны коды 0xFF и 0x00.

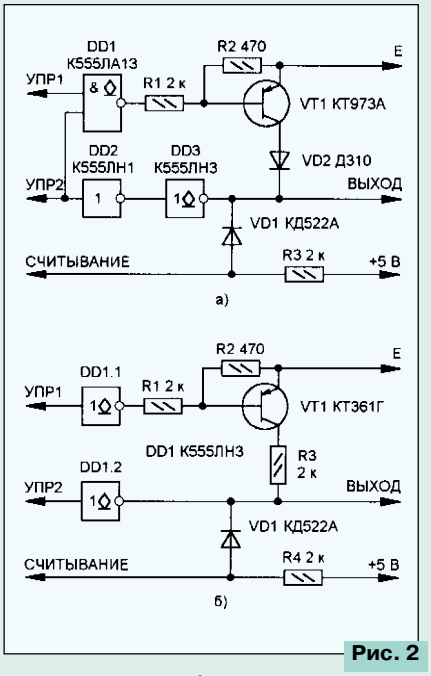

Рис. 2

На плате прибора предусмотрены посадочные места под панели для программируемых микросхем серий К556, К1556, 27xx, 28xx, 29xx, 8748 и 8749, 8x5x, а также К155РЕ3. Контактные площадки в нужном порядке соединены с выходами коммутаторов и программируемых источников напряжения. Имеется также кварцевый резонатор, подключенный к панелям тех микроконтроллеров, при программировании которых он необходим.

Некоторые микросхемы, не названные выше, тоже можно «уложить» в имеющиеся панели, но рациональнее воспользоваться специально предусмотренным разъемом, на который выведены все нужные цепи. К нему можно подключить плату с панелью под любую микросхему, например, в корпусе PLCC.

ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ

Поставляемый с прибором пакет программ Uniprog Plus представляет собой систему программирования самых различных микросхем ПЗУ, ПЛМ и т. д. Это открытая система: функции, отвечающие за обслуживание микросхем конкретного типа, реализуются внешними загружаемыми модулями. Для каждой из них обеспечивается выполнение всех операций установки режимов программирования и собственно программирования, предусмотренных в соответствующем модуле, а также набор операций контроля.

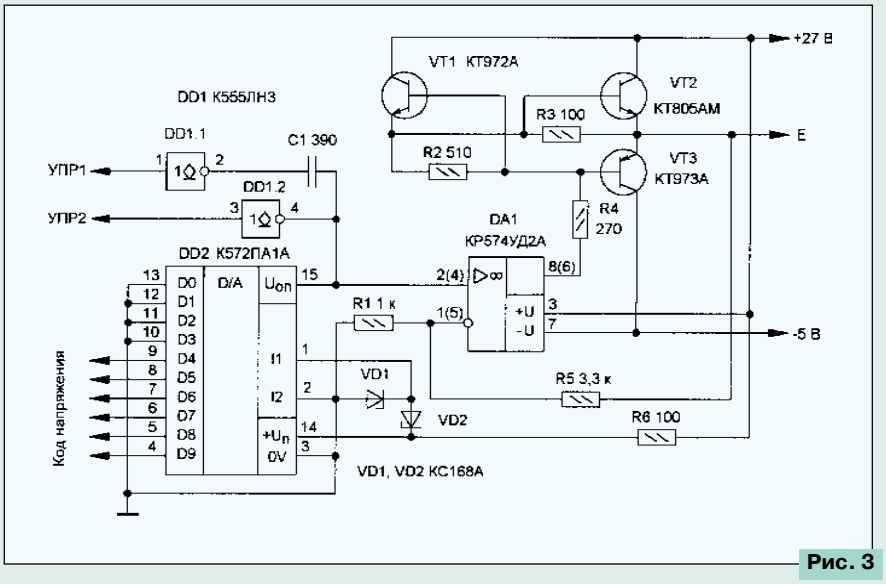

Рис. 3

В настоящее время в составе Uniprog Plus имеются следующие модули:

ROM.ED — редактор ПЗУ;
PAL.ED — редактор ПЛМ;
27XX.PRG — программирование УФ РПЗУ серий 27xx, 573, К573;
2728.ADT — автоматическое определение типа микросхем серий 27xx, 28xx, 29xx;
RTXX.PRG — программирование ППЗУ с плавкими перемычками серий К556, КР556;
1556X.PRG — программирование ПЛМ серии К1556;
RT1.PRG — программирование ПЛМ серии К556;
VE4X.PRG — программирование микроконтроллеров серий 874x;
VE51.PRG — программирование микроконтроллеров серий 875x, КР1816, КР1830, 89xx;
28XX.PRG — программирование микросхем FLASH-памяти серий 28xx, 29xx;
TEST.PRG — тестирование платы программатора.

В стадии разработки находятся модули программирования микроконтроллеров PIC, последовательных (битных) ППЗУ и проверки микросхем ОЗУ.

В комплект Uniprog Plus входит пакет программ Uniprog Developer's Kit (подробнее о нем рассказывается далее), позволяющий самостоятельно создавать новые программирующие модули. Кроме того, к Uniprog Plus можно подключить любые определяемые пользователем программы-конвертеры, преобразующие различные формы представления образа ПЗУ в вид, необходимый для программирования.

Экранный вариант программы Uniprog Plus выполняет следующие операции:

— **работа с файлами**: создание/загрузка/сохранение буфера редактирования, открытие/компиляция файла конвертером и т.д.;
— **редактирование**: отмена последнего изменения, начало/конец/снятие выделения, операции с «записной книжкой», заполнение блока значением, логические операции, поиск, различные переходы. Просмотр и редактирование содержимого буфера программирования. Данные могут быть представлены по выбору в виде массива четырехразрядных тетрад (младших и старших половин байта), байтов, слов или двойных слов. Каждый элемент массива изображается соответствующим символом кода ASCII и двоичным, восьмиричным, десятичным либо шестнадцатиричным числом. При работе с ПЛМ содержимое буфера программирования представляется набором матриц И, ИЛИ, НЕ;
— **выбор типа ПЗУ**: тип выбирается из экранного меню. Номенклатура программируемых микросхем соответствует заданной в конфигурационном файле. Функция Autodetect пытается определить тип ПЗУ автоматически;
— **действия с ПЗУ**: программирование, разные проверки (на чистоту, возможность допрограммирования, совпадение с содержимым буфера), стирание/запись бита защиты и т. д.;
— **настройка режимов и конвертеров**: установка режимов программирования, настройка оболочки Uniprog Plus, подключение/редактирование конвертеров. Режимы устанавливают в диалоге, полностью зависящем от конкретного модуля программирования, например, для УФ ППЗУ серии 27xxx имеется 18 различных вариантов. При необходимости можно включить или выключить контроль записи и дать произвольные значения всем переменным алгоритма программирования;
— **операции с окнами**: перемещение, масштабирование, раскрытие, восстановление, переход к следующему, закрытие, разложить/выстроить окна;
— **разные операции**: калькулятор, вызов внешних утилит, информация об Uniprog Plus.

Версия программы, параметры которой задаются в командной строке DOS, выполняет те же функции, что и экранная, за исключением интерактивных (просмотра и редактирования данных) и модификации буфера программирования. Она может быть полезна при постоянной работе с ПЗУ одного и того же типа, позволяя обойти утомительные операции ручной установки режимов при каждом запуске программы.

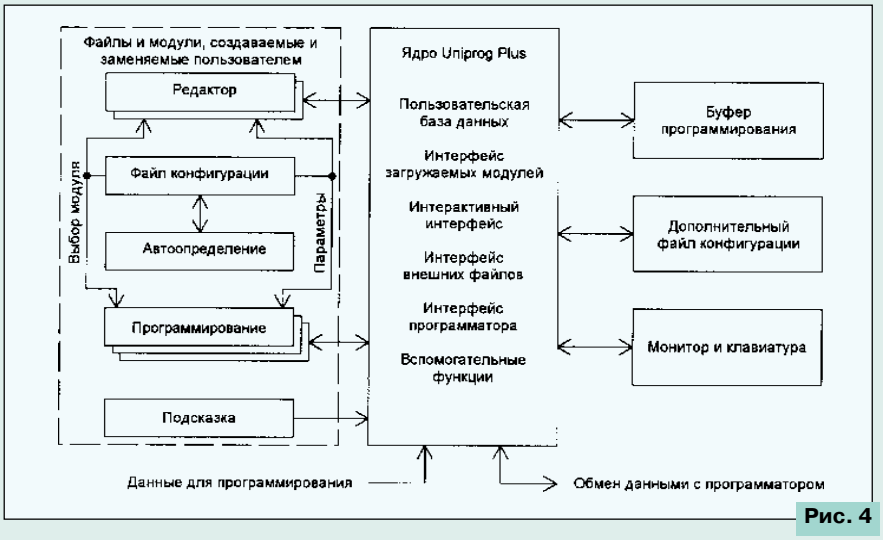

Рис. 4

ПАКЕТ UNIPROG DEVELOPER'S KIT

Как говорилось выше, пользователь имеет возможность создавать и подключать к Uniprog Plus собственные модули программирования и тестирования микросхем, редактирования данных для программирования, автоматического определения типа микросхемы и конфигурационные файлы. В этом ему поможет Uniprog Developer's Kit. Подробное

описание всех возможностей этого пакета потребовало бы слишком много места. Поэтому очень кратко остановимся лишь на общих принципах.

На рис. 4 показано взаимодействие ядра программы Uniprog Plus с модулями, подготовленными пользователем. Внутри ядра находятся основные интерфейсы, взаимодействующие с внешними (по отношению к нему) модулями и файлами данных, и другие неизменяемые части программы, обеспечивающие ее функционирование.

Модуль «Программирование» — собственно программа записи данных в микросхему, их чтения, сравнения и т. д., — реализует соответствующие временные диаграммы с учетом всевозможных параметров этих процессов. Пользователь может разработать собственный модуль для нужной ему микросхемы, не вникая в конкретное устройство программатора и пользуясь только логическими понятиями шины данных, шины адреса, управляющих сигналов. Для этого в ядре Uniprog Plus имеется ряд стандартных функций, к которым можно обращаться из любого модуля.

Модуль «Редактор» служит для отображения на экране монитора содержимого буфера программирования с данными, предназначенными для занесения в ПЗУ или прочитанными из него. Чаще всего бывает достаточно поставляемых с программатором бинарного редактора для ПЗУ с линейной структурой и редактора ПЛМ для логических матриц. Но если требуется создать на экране образ ПЗУ в каком-либо необычном виде, придется написать собственный редактор. Задача эта сложная, но выполнимая. Uniprog Developer's Kit предоставляет такую возможность.

Доступен пользователю и модуль «Автоопределение», по многим причинам отделенный от модуля «Программирование». А в модуль «Подсказка» можно поместить справочные данные, относящиеся к модулям собственной разработки.

Информация, необходимая для связи всех модулей с ядром программы и относящаяся к конкретным типам программируемых микросхем, находится в конфигурационном файле, который пользователь может дополнять и редактировать. В дополнительном конфигурационном файле автоматически фиксируются данные о настройках программы, сделанных уже во время работы с ней.

Тип ПЗУ задается пользователем вручную или определяется с помощью модуля «Автоопределение». После этого программа выбирает модули «Редактор» и «Программирование», нужные для работы с ПЗУ этого типа, и передает им из конфигурационного файла необходимые параметры. «Редактор» через ядро Uniprog Plus выдает образ ПЗУ на экран монитора и позволяет редактировать его, пользуясь клавиатурой и «мышью». Модуль «Программирование» через ядро управляет программатором, обеспечивая выполнение всех необходимых операций.

В заключение необходимо отметить, что программа Uniprog Plus бурно развивается как в сторону увеличения числа поставляемых с ней программирующих модулей, так и в сторону упрощения их самостоятельной разработки за счет «интеллектуализации» пакета Uniprog Developer's Kit.