Микроконтроллеры? Это же просто!

Глава 4. Сопряжение микроконтроллера с индикаторами различных типов

Сопряжение с жидкокристаллическими индикаторами на основе контроллеров типа HD44780 фирмы Hitachi

Эти индикаторы принадлежат к числу наиболее распространенных — я встречал более полудюжины их производителей, таких как Data Vision, Bolymin, SII, Optrex, Batron, Picvue, EDT, Varitronix или Povertip. Индикаторы бывают одно, двух- и четырехстрочными. Каждая строка отображает 8, 12, 16, 20, 24 или 40 символов, формируемых матрицей формата 5·7 или 5·10 точек.

В состав индикатора входит микроконтроллер HD44780 или совместимый с ним, производимый фирмами Hitachi, Epson, Toshiba, Philips, Samsung или Sanyo. Он управляет точками ЖК-дисплея и интерфейсной частью индикатора. Как и HT1610, этот индикатор представляет собой печатную плату (ее размеры для однострочного 16-символьного индикатора составляют 80x36x10 мм), на которой смонтирован ЖК-дисплей, контроллер и необходимые дополнительные электронные компоненты. Внешний вид одного из таких индикаторов, DV-16100 от Data Vision, приведен на рис. 29.

На этом сходство DV-16100 с рассмотренным выше HT1610 заканчивается, и начинаются различия.

Интерфейс DV-16100 — параллельный. Для соединения индикатора с микроконтроллером нужно использовать 11 линий — восемь для передачи данных (DB0-DB7), одну для информирования индикатора о направлении обмена (R/W; R/W=1 — чтение, R/W=0 — запись), одну для информирования о типе передаваемых данных (RS; RS=1 — данные, RS=0 — команда), и одну в качестве строб-сигнала, по перепаду которого из 1 в 0 осуществляется запись данных в индикатор или чтение из него. Естественно, на индикатор нужно подать также питающее напряжение (+ 5 В), соединить его с общим проводом, а также подать на соответствующий вывод некий потенциал от 0 до 5 В, который регулирует контраст формируемого индикатором изображения. Схема соединения индикатора DV-16100 с микроконтроллером приведена на рис. 30.

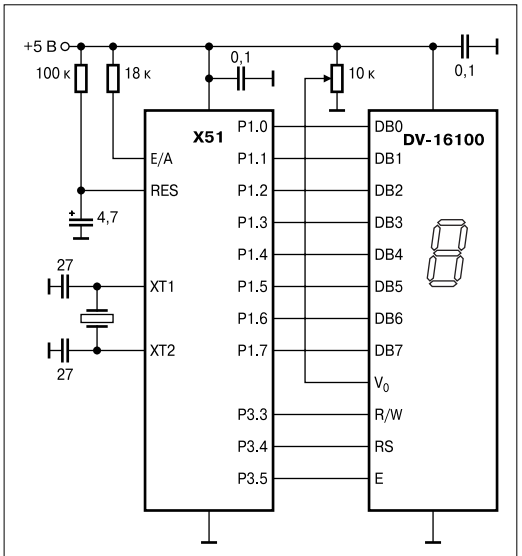

Рис. 30. Схема сопряжения индикатора DV-16100 с микроконтроллером

Отмечу, что DV-16100 допускает возможность работы с использованием не только восьми, но и четырех линий данных (DB4-DB7).

Однако в настоящей статье мы этот режим рассматривать не будем. Желающие, опираясь на приведенную ниже информацию, могут попробовать реализовать этот вариант самостоятельно, так сказать, в качестве домашнего задания.

Индикаторы на основе HD44780 — чрезвычайно гибкие изделия, позволяющие использовать различные режимы ввода в них информации и ее просмотра. Они формируют изображения не только цифр, но и букв латинского (а многие — еще и русского) алфавита, а также псевдографических символов. Однако подобная гибкость имеет и свою оборотную сторону — более сложные по сравнению с HT1610 алгоритмы инициализации и ввода в них информации. В связи с этим вам придется затратить определенные усилия, чтобы понять, что нужно сделать для реализации тех или иных режимов ввода информации и ее отображения. Минимальные начальные сведения об этом, без которых нельзя написать простейшую программу работы с таким индикатором, вы получите из материала, содержащегося в настоящей главе. Всю информацию по тому или иному конкретному индикатору вы найдете в фирменном описании на него. А поскольку даже одинаково организованные индикаторы от различных производителей иногда в деталях отличаются друг от друга, я настоятельно рекомендую перед использованием того или иного индикатора найти описание на него и ознакомиться с ним — это поможет свести к минимуму проблемы с отладкой собранного изделия.

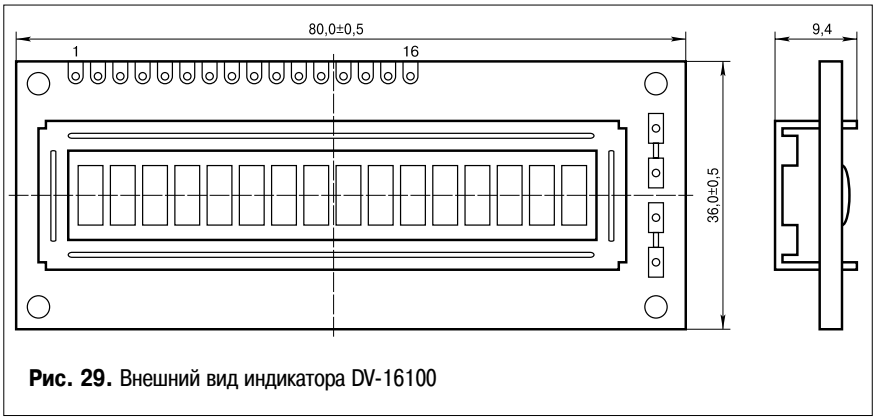

Рис. 29. Внешний вид индикатора DV-16100

Теперь давайте вернемся к конкретному примеру и рассмотрим более подробно уже упоминавшийся DV-16100. Как уже говорилось, это однострочный индикатор, отображающий в строке 16 символов. Для того, чтобы понять, что же он отображает и как его программировать, полезно представить себе его внутреннюю структуру. В очень упрощенном виде она изображена на рис. 31.

Внутри индикатора есть 80 ячеек памяти, называемых обычно видеопамятью. При помощи соответству-

ющих команд, о которых я скажу чуть ниже, мы можем в любую из этих ячеек занести любое 8-битное число (т. е. число от 0 до 0FFH). Каждому из этих чисел взаимно однозначно соответствует при отображении определенный символ: например, числу 32H соответствует цифра 2, числу 47H — заглавная латинская буква G, и т. д. Таблица соответствия, иногда называемая таблицей кодов символов, таблицей знакогенератора или таблицей фонтов (кто во что горазд!) приводится в описании на каждый конкретный индикатор. Для DV-16100 она приведена на рис. 32.

Не вдаваясь в подробности, отмечу, что наибольшие различия между одинаковыми индикаторами различных производителей лежат именно в таблицах кодов символов, точнее в правых их половинах, соответствующих числам от 0A0H до 0FFH. В так называемых русифицированных индикаторах среди символов, соответствующих этим числам, есть символы нашей родной кириллицы. Поэтому, если вы собираетесь на индикаторе отображать русские буквы, лучше приобретать индикаторы именно с такой таблицей кодов символов. Если же вы планируете обойтись только цифрами и латинскими буквами, то вам подойдет индикатор с любой таблицей кодов символов.

Итак, повторю, что внутри индикатора есть 80 ячеек видеопамяти, в которые для отображения мы должны занести 8-битные коды, соответствующие тем символам, которые нам нужно отобразить. Но ведь индикатор-то 16-разрядный! Естественно, возникает вопрос: какие 16 из этих 80 символов будут отображены на нем?

Ответ прост. Любые 16 идущих последовательно символов. А вот какие конкретно, зависит от нас. Очень часто для определенности первой командой перед занесением в индикатор информации ставят команду сброса дисплея. В ходе ее выполнения контроллер индикатора заносит во все ячейки видеопамяти код пробела (20H) и настраивает дисплей так, чтобы в крайнем левом разряде отобразился символ, код которой находится в ячейке видеопамяти с адресом 0, во втором слева разряде — символ, код которого в ячейке с адресом 1, и т. д. Таким образом, сразу после выполнения команды сброса дисплея отображаются

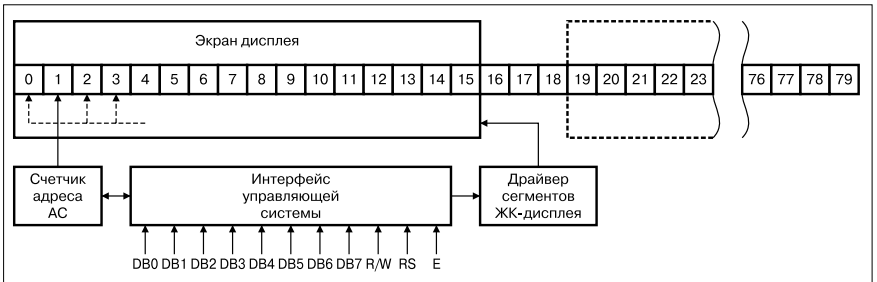

Рис. 31. Упрощенная внутренняя структура индикатора DV-16100

Младшие 4 бита (D0...D3) кода символа в HEX - представлении	Старшие 4 бита (D4...D7) кода символа в HEX - представлении															
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	CG RAM (1)															
1	CG RAM (2)															
2	CG RAM (3)															
3	CG RAM (4)															
4	CG RAM (5)															
5	CG RAM (6)															
6	CG RAM (7)															
7	CG RAM (8)															
8	CG RAM (1)															
9	CG RAM (2)															
A	CG RAM (3)															
B	CG RAM (4)															
C	CG RAM (5)															
D	CG RAM (6)															
E	CG RAM (7)															
F	CG RAM (8)															

Рис. 32. Таблица кодов символов для индикатора DV-16100

символы, коды которых хранятся в первых 16 ячейках видеопамяти.

Правда, как я только что сказал, эта команда заносит во все ячейки видеопамяти пробелы, поэтому для того, чтобы увидеть на дисплее что-то помимо пустой строки, нам после сброса дисплея нужно занести в эти 16 ячеек видеопамяти коды символов, которые нам хотелось бы отобразить. Для этого надо установить в 0 сигнал на входе R/W индикатора. Далее надо установить в 1 сигнал на входе RS дисплея. Как уже говорилось, при RS=1 HD44780 воспринимает передаваемую информацию как данные, а при RS=0 — как команду. На DB7-DB0 нужно вывести передаваемые в индикатор 8 бит, и после того, как все сказанное будет сделано, подать на вход E индикатора строб-сигнал — положительный импульс длительностью не менее 500 нс. По перепаду строб-сигнала из 1 в 0 осуществится запись данных в индикатор. Временные диаграммы операций

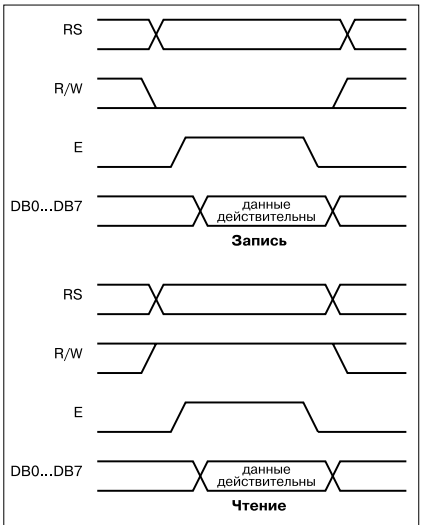

Рис. 33. Временные диаграммы операций чтения и записи для индикатора DV-16100

чтения и записи приведены на рис. 33.

У самых внимательных из читателей уже может возникнуть следующий вопрос: а в какую именно ячейку видеопамяти мы таким образом занесем информацию? Ответ таков : в ту, на которую указывает так называемый счетчик адреса памяти индикатора AC. После команды сброса дисплея AC устанавливается в 0. Следовательно, первые после этого сброса записываемые данные попадут в нулевую ячейку видеопамяти.

А как записать данные во вторую, третью и т. д. ячейки? В нашем распоряжении есть две возможности. Первая — перед каждой записью данных заставить индикатор выполнить команду установки AC в требуемое положение (от 0 до 79), и после этого передавать данные. Это универсальный способ, с помощью которого можно записать данные в любую ячейку видеопамяти в любой последовательности. Но при этом запись данных в 16 ячеек видеопамяти требует 32 циклов записи, ведь перед каждой передачей данных в индикатор должна идти команда установки AC.

Вторая возможность — настроить соответствующей командой HD44780 таким образом, чтобы после каждой записи в видеопамять AC увеличивался бы на 1. Если мы сделаем это, то, как уже говорилось, первая после сброса дисплея запись данных осуществится в ячейку видеопамяти с адресом 0. При этом в момент завершения записи AC увеличится на 1, то есть будет содержать уже не 0, а 1. Следовательно, у нас нет необходимости заставлять индикатор выполнять команду установки AC в 1, и мы сразу вслед за упомянутой пересылкой кода символа в нулевую ячейку можем записать в индикатор код символа, который должен оказаться в ячейке 1. Когда мы осуществим эту запись, AC снова увеличится на 1 и станет равным 2. Значит, следующие записываемые данные попадут во вторую ячейку. А как AC? Правильно, увеличится до 3. Еще раз запишем в индикатор данные (теперь уже в третью ячейку), и снова AC возрастет на 1, указывая на следующую, теперь уже 4-ю ячейку видеопамяти. И так далее.

Надеюсь, то, что я рассказал в предыдущем абзаце, понятно всем. Если нет — еще раз внимательно прочитайте его, так как именно таким образом мы будем в рассматриваемой программе сопряжения МК с DV-16100 передавать данные в индикатор.

Мы рассмотрели самый простой способ вывода данных на индикатор типа DV-16100: предварительно настраиваем его на работу в режиме увеличения AC на 1 после каждой записи данных в видеопамять, даем команду сброса дисплея и после этого записываем в индикатор один за другим коды всех 16 символов, которые мы хотим отобразить. Кстати замечу, что такую же последовательность действий для отображения данных на этом индикаторе должен выполнить любой контроллер — будь то x51, AVR, или PIC-контроллер. Так что если вы разберетесь с описанным алгоритмом, реализуете его на x51, а спустя некоторое время захотите перейти на работу с другим контроллером, вам понадобится лишь переписать вашу программу на языке этого контроллера. А это не так уж сложно, гораздо проще, чем разбираться с работой неизвестного вам доселе устройства.

Еще одно замечание. Вернемся к рис. 31. На нем помимо ячеек видеопамяти вы видите прямоугольное окошко длиной в 16 ячеек. Как вы, наверное, догадались, прямоугольник этот — не что иное как экран дисплея. На рисунке он расположен таким образом, что вбирает в себя (то есть отображает) первые 16 ячеек видеопамяти. Это его положение после выполнения команд сброса дисплея или возврата (последняя отличается от первой тем, что не меняет содержимого ячеек видеопамяти). Но в системе команд HD44780 есть и такие команды, которые сдвигают это окошко влево или вправо. В результате можно отобразить на дисплее не только с 0 по 15 ячейки видеопамяти, но и, к примеру, с 8 по 23-ю, или с 19-й по 34 (положение окошка, соответствующее послед нему случаю, отмечено пунктиром). Вы можете занести информацию, к примеру, во все 80 ячеек видеопамяти, а затем отображать то одно, то другое, то третье, переводя это окошко туда, где оно выхватит те 16 ячеек, которые вам нужно отобразить в текущий момент. Я плохо представляю себе, зачем нужен подобный режим, но наверняка для кого-то это покажется весьма удобно, и он с удовольствием его реализует. Правда, ему придется реализовывать такой режим работы индикатора самостоятельно, без моих подсказок, так как я не планирую рассмотрение реализации этого режима.

Несколькими абзацами выше я сказал, что индикатор нужно настроить таким образом, чтобы счетчик AC увеличивался на 1 после каждой записи данных в видеопамять. А какие еще настройки нужно осуществить?

Как уже упоминалось, индикатор может вести обмен информацией с микроконтроллером как по 4-разрядной, так и по 8-разрядной шине

R/W	RS	D7	D6	D5	D4	D3	D2	D1	D0	Название
0	0	0	0	0	0	0	0	0	0	Сброс дисплея
0	0	0	0	0	0	0	0	1	*	Возврат дисплея
0	0	0	0	0	0	0	1	I/D	S	Автоинкремент/сдвиг
0	0	0	0	0	0	1	DL	C	B	Управление включением/выключением
0	0	0	0	0	1	S/C	R/L	*	*	Сдвиг курсора/экрана
0	0	0	0	1	D/L	N	F	*	*	Параметры развертки и шины данных
0	0	0	1	AG	AG	AG	AG	AG	AG	Установка АС в ОЗУ знакогенератора
0	0	1	AD	AD	AD	AD	AD	AD	AD	Установка АС в ОЗУ видеопамяти

* - безразлично

Рис. 34. Команды индикатора DV-16100

данных. Следовательно, мы должны указать ему, по какой же шине мы планируем передавать ему информацию. Далее, индикатор может быть как однострочным, так и многострочным, и мы должны сообщить ему, с каким количеством строк он работает. Нам нужно также выбрать размер матрицы, формирующей символы на индикаторе — 5·7 или 5·9 точек. Также подлежит уточнению наличие и форма отображения курсора — в виде линии подчеркивания или в виде мигающего символа. Все эти и некоторые другие настройки осуществляются соответствующими командами, посылаемыми индикатору нашим МК.

Перечень этих команд приведен на рис. 34. Передача каждой из них осуществляется записью в индикатор (при RS=0) соответствующего ей байта, в котором в зависимости от требуемых нам настроек установлены в нули или в единицы биты I/D, S, D, C, B, S/C, R/L, DL, N, F. Что они означают?

I/D=1 как раз и означает, что после записи в видеопамять каждого байта данных АС должен увеличиваться на 1. S=1 разрешает упомянутый выше сдвиг окошка дисплея, а S=0 запрещает сдвиг. D=1 включает дисплей, C=1 включает курсор в виде символа подчеркивания, B=1 — в виде мигающего символа. Установка D, C и B в 0 отключает соответственно дисплей и отображение всех видов курсора.

DL=1 устанавливает режим работы с 8-разрядной шиной данных, N=0 предписывает дисплею работать с одной строкой, F=1 — с матрицей 5x9 точек, а F=0 — 5·7 точек. Биты S/C, R/L относятся к одновременным сдвигам курсора и дисплея, и мы здесь рассматривать их не будем.

AD — адрес ячейки видеопамяти. В последней команде вместо 7 битов AD вы должны разместить адрес ячейки видеопамяти, на которую будет указывать счетчик АС. Этот адрес должен быть в пределах от 0 до 79 (от 0000000B до 1001111B).

AG — адрес ячейки памяти в так называемом знакогенераторе (здесь мы его не рассматриваем, поэтому я воздержусь от объяснений, что это такое).

Обычно команды настройки посылают индикатору сразу после подачи на него напряжения питания. Мы пошлем ему следующие команды: команду установки функций, команду управления включением/выключением дисплея и команду автоинкремента. При этом мы зададим 8-разрядную шину данных (DL=1), работу индикатора с одной строкой (N=0) с матрицей 5·7 точек (F=0), с отключенным изображением курсора (C=0, B=0). Также установим I/D=1 (после записи в видеопамять каждого байта данных АС должен увеличиваться на 1) и S=0 (запретим сдвиг окошка дисплея). Все это выполнит подпрограмма INICDV16, которую при использовании DV-16100 нужно вызвать в самом начале вашей программы, до того, как микроконтроллер должен будет начать отображать результаты выполнения каких-то действий, вычислений или измерений.

```
;
;
INICDV16:
;
     MOV    P1,#11111111B     ; УСТАНОВИЛИ ПОРТ P1
     MOV    P3,#11000111B     ; УСТАНОВИЛИ ПОРТ P3
;
;КОМАНДА УСТАНОВКИ ФУНКЦИЙ С DL=1, N=0, F=0
;
     MOV    P3,#11000111B                ;RS=0, E=0, R/W=0
     MOV    P1,#00110000B                ;DL=1, N=0, F=0
     LCALL  IMPULS1                      ;ИМПУЛЬС E
;
;КОМАНДА УПРАВЛЕНИЯ ВКЛЮЧЕНИЕМ/ВЫКЛЮЧЕНИЕМ ДИСПЛЕЯ
;C D=1, C=0, B=0
;
     MOV    P3,#11000111B                ;RS=0, E=0, R/W=0
     MOV    P1,#00001100B                ;D=1, C=0, B=0
     LCALL  IMPULS1                      ;ИМПУЛЬС E
;
;КОМАНДА АВТОИНКРЕМЕНТА С I/D=1, S=0
;
     MOV    P3,#11000111B                ;RS=0, E=0, R/W=0
     MOV    P1,#00000110B                ;I/D=1, S=0
     LCALL  IMPULS1                      ;ИМПУЛЬС E
;
     LCALL  DEL16MS                      ;ЗАДЕРЖКА 16 МС
;
            RET
;
;ПОДПРОГРАММА ЗАДЕРЖКИ 64 МС
;
DEL64MS:LCALL  DEL16MS                   ;ЗАДЕРЖКА 64 МС
     LCALL  DEL16MS
     LCALL  DEL16MS
     LCALL  DEL16MS
     RET
;
;ПОДПРОГРАММА ЗАДЕРЖКИ 16 МС
;
DEL16MS:LCALL  DEL4MS                    ;ЗАДЕРЖКА 16 МС
     LCALL  DEL4MS
     LCALL  DEL4MS
     LCALL  DEL4MS
     RET
;
;ПОДПРОГРАММА ЗАДЕРЖКИ 4 МС
;
DEL4MS: LCALL  DEL1MS                    ;ЗАДЕРЖКА 4 МС
     LCALL  DEL1MS
     LCALL  DEL1MS
     LCALL  DEL1MS
     RET
;
;ПОДПРОГРАММА ЗАДЕРЖКИ 1 МС
;
DEL1MS:                                  ;ЗАДЕРЖКА 1 МС
     MOV    R1,#25      ; ПОВТОРЯЕМ 25 РАЗ
```

```
;
LREX:  MOV    R2,#18
LRIN:  DJNZ   R2,LRIN      ; 36+1 МКС НА 12
МГЦ
;
      DJNZ   R1,LREX
      RET
;
;ПОДПРОГРАММА, ФОРМИРУЮЩАЯ ИМ-
ПУЛЬС НА ВХОДЕ E
;
IMPULS: SETB   P3.5        ; ИМПУЛЬС E
      NOP
      NOP
      NOP
      NOP
      CLR    P3.5
      RET
;
;ПОДПРОГРАММА ИМПУЛЬСА НА E С
;ПОСЛЕДУЮЩЕЙ ЗАДЕРЖКОЙ 1 МС
;
IMPULS1:ACALL  IMPULS      ;ИМПУЛЬС E
      ACALL  DEL1MS      ;ПОСЛЕД.ЗАДЕРЖКА
1 МС
      RET
;
```

Обращаю ваше внимание на следующие моменты. Как следует из рис. 30, входы E, RS и R/W индикатора соединены соответственно с линиями P3.5, P3.4 и P3.3 нашего микроконтроллера, поэтому перед тем, как подать на индикатор строб-сигнал записи (положительный импульс на вход E), мы устанавливаем эти линии в нули. P3.5=0 для того, чтобы можно было сформировать положительный импульс, P3.4=0 — мы передаем в индикатор команду, P3.3=0 — будет операция записи. Остальные линии порта P3 в данном применении не используются, в связи с чем мы устанавливаем на них единицы.

Далее, для упрощения работы я не проверяю, завершил ли индикатор выполнение той или иной операции, а ставлю после этих операций задержки, длительность которых гарантированно больше, чем время выполнения предшествующих им операций. Это задержка на 16 мс после завершения подпрограммы INICDV16 (LCALL DEL16MS), и 1 мс после окончания строб-сигнала записи (ACALL DEL1MS в составе подпрограммы IMPULS1). Более подробно о том, как организованы эти подпрограммы задержек поговорим чуть ниже. Пока же вернемся к подпрограмме инициализации индикатора.

Первыми командами (MOV P1,#11111111B и MOV P3,#11000111B) мы устанавливаем в 1 все линии портов P1 и P3, кроме упомянутых P3.5, P3.4 и P3.3. Далее этими же командами выводим на линии порта P3 нули на все тех же P3.5, P3.4 и P3.3, а на P1 — коды, соответствующие командам установки функций, управления включением/выключением дисплея и автоинкремента с выбранными значениями DL, N, F, D, C, B, I/D, S. После вывода на линии P1 кода каждой из упомянутых команд вызываем подпрограмму IMPULS1, формирующую на входе E индикатора положительный импульс длительностью несколько микросекунд с последующей миллисекундной задержкой. Собственно импульс формирует подпрограмма IMPULS, а далее стоящая команда ACALL DEL1MS вызывает формирующую миллисекундную задержку подпрограмму DEL1MS.

Теперь непосредственно о том, как устроена подпрограмма DEL1MS. До сих пор мы формировали задержки, вставляя в нужные места программы команды NOP, время выполнения каждой из которых при тактовой частоте нашего МК 12 МГц составляет ровно 1 мкс. Пока речь шла о задержках длительностью не более 10–20 мкс, это было вполне приемлемо. Но сейчас нам нужна задержка на 1 мс = 1000 мкс. Не писать же подпрограмму, состоящую из 1000 команд NOP. Как быть?

Чаще всего подобную задачу решают следующим образом. Рассмотрим приведенный ниже фрагмент подпрограммы.

```
;
LREX:  MOV    R2,#18
LRIN:  DJNZ   R2,LRIN ;36+1 МКС НА 12 МГЦ
;
```

Первая команда, идущая после метки LPEX, очевидна — занесение в регистр R2 числа 18. Команда DJNZ нам не знакома. Познакомимся же с ней.

Аббревиатура этой команды состоит из первых букв словосочетания Decrement and Jump Not Zero, что в переводе означает декрементировать (то есть уменьшить на 1) и перейти, если то, что мы декрементировали, не равно 0. Как теперь нетрудно догадаться, команда DJNZ R2,LRIN означает буквально следующее: уменьшить R2 на 1, и если содержимое этого регистра после уменьшения не равно 0, то перейти на выполнение команды, идущей после метки, имя которой (LRIN) стоит через запятую после наименования нашего регистра (R2).

Но ведь после этой метки в нашей подпрограмме стоит именно сама команда DJNZ R2,LRIN! Следовательно, уменьшив R2 на 1 и обнаружив, что регистр содержит все еще ненулевой результат, микроконтроллер снова вернется к выполнению этой же самой команды. Выполнив ее еще раз, и вновь получив ненулевой R2, МК вынужден будет повторять команду до тех пор, пока R2 не станет равным 0. Сколько же раз он ее повторит? Естественно, 18 — ведь именно это число мы поместили в R2 перед тем, как заставили контроллер выполнять команду DJNZ R2,LRIN .

Теперь я забегу немного вперед и скажу, что у МК·51 при тактовой частоте 12 МГц команда занесения числа в какой-либо из регистров осуществляется за 1 мкс, а команда DJNZ R2,LRIN — за 2 мкс. Однократное выполнение первой из них и последующее 18-кратное выполнение второй осуществится за 37 мкс, ни больше, ни меньше. Запомним это и двинемся дальше.

```
MOV    R1,#25 ;ПОВТОРЯЕМ 25 РАЗ
LREX:  MOV    R2,#18
LRIN:  DJNZ   R2,LRIN ;36+1 МКС НА 12 МГЦ
      DJNZ   R1,LREX
      RET
```

Перед вами приведены команды подпрограммы миллисекундной задержки. Со второй и третьей командами мы уже разобрались. Первая, как вы понимаете, заносит в регистр R1 число 25. А вот четвертая заставляет МК именно эти 25 раз выполнить фрагмент из второй и третьей команды.

Как это происходит? Наверное, многие уже догадались, но я все же прокомментирую. Вначале мы заносим в регистр R1 число 25. Далее выполняем вторую и третью команды. Как мы уже несколько раз говорили, их выполнение длится до тех пор, пока R2 не станет равным 0, и будет это продолжаться 37 мкс. После того, как R2 достигнет нуля, наступит черед выполнения последней команды. Она уменьшит R1 на 1 (то есть до 24), и поскольку 24 не равно 0, то отправит МК на выполнение команды, идущей после метки LREX:. А там... да, да, мы опять заносим в R2 18 и в течение

37 мкс командой DJNZ R2,LRIN уменьшаем его до 0. Затем снова наступит черед выполнения последней команды. Она опять уменьшит R1 на 1 (теперь уже до 23), и опять отправит МК на выполнение команды, идущей после метки LREX:. И так будет продолжаться, пока R1 не достигнет 0, то есть 25 раз.

Обратите внимание, чтокаждое выполнение фрагмента, состоящего из второй и третьей команды, занимает 37 мкс. Далее, каждый раз после этого фрагмента МК выполняет четвертую команду, длительность которой, как вы догадываетесь, 2 мкс. Итого, 25-кратное выполнение блока из второй, третьей и четвертой команд отнимет у микроконтроллера 25·37+25·2=975 мкс. Приплюсуем сюда 1 мкс на выполнение первой команды, 2 мкс на выполнение команды RET, и 2 мкс, которые микроконтроллер тратит на выполнение команды вызова подпрограммы миллисекундной задержки ACALL DEL1MS. В сумме получаем 980 мкс. Это чуть меньше обещанной миллисекунды, но в настоящем примере 980 мкс вполне достаточно, чтобы индикатор гарантированно завершил процесс записи. По этой причине я не стал добиваться того, чтобы выполнение подпрограммы DEL1MS занимало ровно миллисекунду, ни микросекундой больше или меньше. Но если такая задача возникла бы, то решить ее не составило бы труда — нужно было бы всего-навсего перед командой RET поставить 20 команд NOP.

Кажется, с подпрограммой миллисекундной задержки все. Если кому что осталось неясно, еще раз попытайтесь внимательно перечитать последние восемь абзацев. Там достаточно информации, чтобы понять и идею построения подпрограммы, и реализацию этой идеи. Перейдем теперь к подпрограммам 4-, 16- и 64-миллисекундной задержек.

Как следует из приведенного выше, подпрограмма 4-миллисекундной задержки (DEL4MS) устроена предельно просто — она всего-навсего четырежды вызывает только что рассмотренную подпрограмму DEL1MS, и все! Не сложнее ее подпрограммы 16-миллисекундной (DEL16MS) и 64-миллисекундной (DEL64MS) задержек — они, в свою очередь, четырежды вызывают подпрограммы соответственно 4-миллисекундной и 16-миллисекундной задержек. В общем, теперь в ваших руках есть средства для построения задержек любой длительности — от единиц микросекунд до десятка секунд. Напомню только, что в приводимом примере упомянутые задержки чуть-чуть меньше 1, 4, 16 и 64 миллисекунд. Если же вам понадобятся задержки, точно равные 4 или, к примеру, 10 мс, вам придется дополнять рассмотренные фрагменты необходимым числом команд NOP.

Итак, мы научились инициализировать индикатор DV-16100 и ему аналогичные, а заодно познакомились с методом формирования задержек средней и большой длительности. Теперь надо разобраться с главным: как же выводить требуемую нам информацию на экран индикатора.

Давайте представим себе, что сразу после включения питания нашего устройства, содержащего рассматриваемый индикатор, мы хотим вывести на его экран надпись, содержащую название прибора и номер версии работающей в микроконтроллере программы. Кстати, настоятельно рекомендую придерживаться по возможности такой практики. Дело в том, что написанные нами программы практически всегда содержат некоторое количество ошибок, выявляемых не сразу, а с течением времени. Также иногда по тем или иным причинам в изделии приходится делать некоторые аппаратные доработки, и как следствие этого, так или иначе корректировать программу. По этим причинам с течением времени у вас будет накапливаться несколько слегка различающихся версий одной и той же программы. И когда вы столкнетесь с необходимостью доработки или ремонта устройства, вам необходимо будет точно знать, какая же из них зашита в памяти программ стоящего в нем микроконтроллера. Так что если при включении он сам об этом вам скажет, выведя на экран название прибора и номер версии работающей в нем программы, вам это не покажется ненужным излишеством.

Будем считать, что я убедил вас в необходимости вывода на экран индикатора такой информации. Предположим, что наше изделие является оптимизатором какого-либо процесса, и номер версии стоящей в нем программы 11. Поскольку рассматриваемый индикатор — однострочный 16-разрядный, выводимая информация не должна содержать более 16 символов. Этому условию удовлетворяет, например, такая надпись: OPTIMIZER VER.11 . Надеюсь, не надо объяснять, что с тем же успехом это могла быть и другая надпись — AREOMETER VER.33 или VOLTMETER VER.07, и для их формирования нам нужно было бы просто занести в индикатор коды не тех букв и цифр, которые сформируют надпись OPTIMIZER VER.11 , а несколько иных.

Для того, чтобы решить поставленную задачу, программист должен сделать следующее. После обычно идущей в самом начале любой программы настройки линий портов и занесения требуемой информации в регистры он должен поставить вызов рассмотренной выше подпрограммы инициализации индикатора, а вслед за ним — вызов приведенной ниже подпрограммы отображения на экране индикатора надписи OPTIMIZER VER.11 .

```
;ИНДИКАТОР НАСТРОЕН. ТЕПЕРЬ ВЫВОД НА ЭКРАН
;НАДПИСИ OPTIMIZER VER.11
;
OTB:  MOV   P3,#11000111B  ;Clear display
      MOV   P1,#00000001B
      LCALL IMPULS1
;
      LCALL DEL16MS
;
      MOV   P3,#11010111B
      MOV   P1,#01001111B          ;»O»
      LCALL IMPULS1
;
      LCALL DEL16MS
;
      MOV   P3,#11010111B
      MOV   P1,#01010000B          ;»P»
      LCALL IMPULS1
;
      LCALL DEL16MS
;
      MOV   P3,#11010111B
      MOV   P1,#01010100B          ;»T»
      LCALL IMPULS1
;
      LCALL DEL16MS
;
      MOV   P3,#11010111B
```

```
    MOV   P1,#01001001B          ;»I»
    LCALL IMPULS1
;
    LCALL DEL16MS
;
    MOV   P3,#11010111B
    MOV   P1,#01001101B          ;»M»
    LCALL IMPULS1
;
    LCALL DEL16MS
;
    MOV   P3,#11010111B
    MOV   P1,#01001001B          ;»I»
    LCALL IMPULS1
;
    LCALL DEL16MS
;
    MOV   P3,#11010111B
    MOV   P1,#01011010B          ;»Z»
    LCALL IMPULS1
;
    LCALL DEL16MS
;
    MOV   P3,#11010111B
    MOV   P1,#01000101B          ;»E»
    LCALL IMPULS1
;
    LCALL DEL16MS
;
    MOV   P3,#11010111B
    MOV   P1,#01010010B          ;»R»
    LCALL IMPULS1
;
    LCALL DEL16MS
;
    MOV   P3,#11010111B
    MOV   P1,#00100000B          ;» «
    LCALL IMPULS1
;
    LCALL DEL16MS
;
    MOV   P3,#11010111B
    MOV   P1,#01010110B          ;»V»
    LCALL IMPULS1
;
    LCALL DEL16MS
;
    MOV   P3,#11010111B
    MOV   P1,#01000101B          ;»E»
    LCALL IMPULS1
;
    LCALL DEL16MS
;
    MOV   P3,#11010111B
    MOV   P1,#01010010B          ;»R»
    LCALL IMPULS1
;
    LCALL DEL16MS
;
    MOV   P3,#11010111B
    MOV   P1,#00101110B          ;».»
    LCALL IMPULS1
;
    LCALL DEL16MS
;
    MOV   P3,#11010111B
    MOV   P1,#00110001B          ;»1»
    LCALL IMPULS1
;
    LCALL DEL16MS
;
    MOV   P3,#11010111B
    MOV   P1,#00110001B          ;»1»
    LCALL IMPULS1
;
    LCALL DEL16MS
;
    RET
;
```

Как нетрудно заметить, по структуре она очень похожа на подпрограмму инициализации индикатора. Мы настраиваем линии порта P3, управляющие сигналами на входах E, RS и R/W индикатора (соответственно P3.5, P3.4 и P3.3), затем выводим на линии порта P1 код команды или отображаемого символа, вызываем формирующую строб-импульс записи подпрограмму IMPULS1 и 16-миллисекундную задержку. И так пока не выведем всю необходимую информацию.

Но есть и отличия. Вначале подпрограммы идет команда сброса дисплея, устанавливающая счетчик адреса AC в 0 и проецирующая окошко дисплея на первые 16 ячеек видеопамяти. При этом на линии порта P1 выводится код команды сброса дисплея (00000001B), а на линии порта P3 — код 11000111B, устанавливающий нулевые сигналы на входах E, RS и R/W индикатора. Далее мы последовательно начинаем пересылать в индикатор коды отображаемых символов. При этом — обратите внимание! — на линии порта P3 мы выводим не 11000111B, а 11010111B, то есть бит, управляющий входом RS, мы устанавливаем в 1. Зачем? Затем, что если сигнал на входе RS индикатора нулевой, то последний воспринимает передаваемую ему по входам DB7...DB0 (то есть по линиям P1) информацию как код команды, а если RS =1 — код отображаемого символа. Поэтому при пересылке кодов отображаемых символов на линии порта P3 мы выводим 11010111B.

Остальное предельно просто. 01001111B — это, как следует из приведенной на рис. 32 таблицы кодов символов, код буквы «O», 01010000B — буквы «P», 01010100B — буквы «T», и т. д. Поняв это, вы теперь сможете сформировать на нашем индикаторе любую надпись. Как видите, все действительно элементарно. Нужно лишь придумать, какой текст вы хотите вывести на экран, выписать без ошибок коды составляющих его символов да подставить их в подпрограмму OTB. И любоваться творением рук своих.

В общем, вы узнали об индикаторах на основе HD44780 почти все, что нужно для того, чтобы успешно начать работать с ними. Осталась самая малость — научиться выводить на экран цифровую информацию, получаемую микроконтроллером в процессе измерений или вычислений. Я думаю, что многие из вас уже и самостоятельно смогли бы справиться с этой задачей. Но тем не менее я приведу еще один пример — подпрограмму отображения на экране индикатора цифр, хранящихся в ОЗУ микроконтроллера в ячейках с адресами от AD00+1 до AD00+11 (см. ниже). Этот пример интересен тем, что цифровая информация перед выводом в индикатор требует определенной перекодировки, так как коды цифр от 0 до 9 (см. рис. 32) отличаются от

двоичного или двоично-десятичного представления, в котором они, как правило, хранятся в ОЗУ МК.

О том, в какой из разрядов индикатора выводится содержимое той или иной ячейки, ясно из комментариев. Также отмечу, что в 4-й, 7-й, 14-й, 15-й и 16-й разряды индикатора эта подпрограмма выводит пробелы (SPACE). Как следует из таблицы на рис. 32, код пробела — 20H. Но в теле подпрограммы, как несложно заметить, я ставлю не этот код, а его символическое имя (то есть не MOV P1,#020H, а MOV P1,#SPACE). Поэтому в начале вашей основной программы, где идет описание адресов регистров и символических имен, не забудьте поставить строчку SPACE .EQU 20H. Если этого не сделать, ассемблер не поймет, что вы от него хотите, и в тех строках, где он найдет это имя, он даст сообщения об ошибках (см. гл. 2).

```
;
;
;
IZOBR1:
    MOV   P3,#11000111B          ;СБРОС ДИСПЛЕЯ
    MOV   P1,#00000001B
    LCALL IMPULS1
    LCALL DEL16MS
;
    MOV   P3,#11010111B
    MOV   A,AD00+1
    CJNE  A,#0FH,IZR1
    MOV   A,#SPACE
    SJMP  IZR2
IZR1: ORL   A,#00110000B
IZR2: MOV   P1,A
    LCALL IMPULS1                ;ОТОБРАЗИЛИ AD00+1 В
;1-Й РАЗРЯД ИНДИКАТОРА
;
    MOV   P3,#11010111B
    MOV   A,AD00+2
    CJNE  A,#0FH,IZR3
    MOV   A,#SPACE
    SJMP  IZR4
IZR3: ORL   A,#00110000B
IZR4: MOV   P1,A
    LCALL IMPULS1                ;ОТОБРАЗИЛИ AD00+2 ВО
;2-Й РАЗРЯД ИНДИКАТОРА
;
    MOV   P3,#11010111B
    MOV   A,AD00+3
    CJNE  A,#0FH,IZR5
    MOV   A,#SPACE
    SJMP  IZR6
IZR5: ORL   A,#00110000B
IZR6: MOV   P1,A
    LCALL IMPULS1    ;ОТОБРАЗИЛИ AD00+3 В
;3-Й РАЗРЯД ИНДИКАТОРА
;
    MOV   P3,#11010111B
    MOV   P1,#SPACE
    LCALL IMPULS1    ;ОТОБРАЗИЛИ SPACE В
;4-Й РАЗРЯД ИНДИКАТОРА
;
    MOV   P3,#11010111B
    MOV   A,AD00+4
    CJNE  A,#0FH,IZR7
    MOV   A,#SPACE
    SJMP  IZR8
IZR7: ORL   A,#00110000B
IZR8: MOV   P1,A
    LCALL IMPULS1    ;ОТОБРАЗИЛИ AD00+4 В
;5-Й РАЗРЯД ИНДИКАТОРА
;
    MOV   P3,#11010111B
    MOV   A,AD00+5
    CJNE  A,#0FH,IZR9
    MOV   A,#SPACE
    SJMP  IZR10
IZR9: ORL   A,#00110000B
IZR10: MOV   P1,A
    LCALL IMPULS1    ;ОТОБРАЗИЛИ AD00+5 В
;6-Й РАЗРЯД ИНДИКАТОРА
;
    MOV   P3,#11010111B
    MOV   P1,#SPACE
    LCALL IMPULS1    ;ОТОБРАЗИЛИ SPACE В
;7-Й РАЗРЯД ИНДИКАТОРА
;
    MOV   P3,#11010111B
    MOV   A,AD00+6
    CJNE  A,#0FH,IZR11
    MOV   A,#SPACE
    SJMP  IZR12
IZR11: ORL   A,#00110000B
IZR12: MOV   P1,A
    LCALL IMPULS1    ;ОТОБРАЗИЛИ AD00+6 В
;8-Й РАЗРЯД ИНДИКАТОРА
;
    MOV   P3,#11010111B
    MOV   A,AD00+7
    CJNE  A,#0FH,IZR13
    MOV   A,#SPACE
    SJMP  IZR14
IZR13: ORL   A,#00110000B
IZR14: MOV   P1,A
    LCALL IMPULS1    ;ОТОБРАЗИЛИ AD00+7 В
;9-Й РАЗРЯД ИНДИКАТОРА
;
    MOV   P3,#11010111B
    MOV   A,AD00+8
    CJNE  A,#0FH,IZR15
    MOV   A,#SPACE
    SJMP  IZR16
IZR15: ORL   A,#00110000B
IZR16: MOV   P1,A
    LCALL IMPULS1    ;ОТОБРАЗИЛИ AD00+8 В
;10-Й РАЗРЯД ИНДИКАТОРА
;
    MOV   P3,#11010111B
    MOV   A,AD00+9
    CJNE  A,#0FH,IZR17
    MOV   A,#SPACE
    SJMP  IZR18
IZR17: ORL   A,#00110000B
IZR18: MOV   P1,A
    LCALL IMPULS1    ;ОТОБРАЗИЛИ AD00+9 В
;11-Й РАЗРЯД ИНДИКАТОРА
;
    MOV   P3,#11010111B
    MOV   A,AD00+10
    CJNE  A,#0FH,IZR19
    MOV   A,#SPACE
    SJMP  IZR20
IZR19: ORL   A,#00110000B
```

```
IZR20:  MOV    P1,A
        LCALL  IMPULS1     ;ОТОБРАЗИЛИ AD00+10 В
;12-Й РАЗРЯД ИНДИКАТОРА
;
        MOV    P3,#11010111B
        MOV    A,AD00+11
        CJNE   A,#0FH,IZR21
        MOV    A,#SPACE
        SJMP   IZR22
IZR21:  ORL    A,#00110000B
IZR22:  MOV    P1,A
        LCALL  IMPULS1     ;ОТОБРАЗИЛИ AD00+11 В
;13-Й РАЗРЯД ИНДИКАТОРА
;
        MOV    P3,#11010111B
        MOV    P1,#SPACE
        LCALL  IMPULS1     ;ОТОБРАЗИЛИ SPACE В
;14-Й РАЗРЯД ИНДИКАТОРА
;
        MOV    P3,#11010111B
        MOV    P1,#SPACE
        LCALL  IMPULS1     ;ОТОБРАЗИЛИ SPACE В
;15-Й РАЗРЯД ИНДИКАТОРА
;
        MOV    P3,#11010111B
        MOV    P1,#SPACE
        LCALL  IMPULS1     ;ОТОБРАЗИЛИ SPACE В
;16-Й РАЗРЯД ИНДИКАТОРА
;
        RET
;
;
```

Как и для ранее рассмотренных индикаторов, мы предполагаем, что цифры в ячейках AD00+1–AD00+11 хранятся в обычном двоичном представлении: 00000000B — это 0, 00000001B — это 1, ..., 00001001B — это 9. Также аналогично предыдущим случаям 00001111B в AD00+1–AD00+11 — это погашенный символ, то есть пробел. Как уже упоминалось, при выводе информации на индикатор и цифры, и пробел требуют перекодировки. Осуществляется это следующим образом. Вначале каждый символ помещается в аккумулятор, и командой CJNE A,#0FH,IZRxx сравнивается с 0FH. В случае равенства МК понимает, что выводимый в индикатор символ — пробел, и в аккумулятор помещается для дальнейшего вывода в порт P1 код пробела. В противном случае перекодировка осуществляется с использованием команды ORL A,#00110000B.

Эта команда осуществляет побитовое логическое ИЛИ содержимого аккумулятора и числа, в данном случае 00110000B. В результате этой операции 7-й, 6-й, 3-й, 2-й, 1-й и 0-й биты аккумулятора останутся неизменными, так как соответствующие биты числа 00110000B содержат нули, а 5-й и 4-й биты аккумулятора установятся в 1, независимо от того, какими они были до выполнения рассматриваемой команды.

Зачем это нам нужно? Обратимся еще раз к таблице символов на рис. 32. Заметьте, что код нуля — 30H или 00110000B, код единицы — 31H или 00110001B, код двойки — 32H или 00110010B, и т. д., вплоть до девятки (39H или 00111001B). Иными словами, чтобы вывести на индикатор DV-16100 хранящийся в AD00+n нуль (00000000B), нужно переслать в него код нуля, или 00110000B, для отображения единицы (00000001B) — код единицы (00110001B), и т. д. А поскольку код любой и цифра от 0 до 9 отличается от хранящейся в AD00+n в двоичном представлении самой цифры именно содержимым 5-го и 4-го бита, то преобразование цифры в соответствующий ей код осуществляется простой установкой этих самых битов в 1. Что, собственно и делает нам команда ORL A,#00110000B.

На этом мы закончим знакомство с индикаторами на основе контроллеров типа HD44780. Теперь работа с ними вам вполне по плечу. Следующий объект нашего