(Продолжение. Начало — № 4–11/2001)

Микроконтроллеры? Это же просто!

Сопряжение со светодиодными индикаторами типа АЛС318

АЛС318 — наиболее удобный для рассмотрения многоразрядный 7-сегментный светодиодный индикатор. Разобравшись с тем, как его использовать, вы без особых усилий адаптируете описываемые в статье аппаратные и программные средства под любой 7-сегментный индикатор, будь-то панель, набранная из десятка одиночных индикаторов с большими цифрами, или малогабаритный 5-разрядный АЛС328 в 14-выводном DIP-корпусе.

Напомню, что 7-сегментные светодиодные индикаторы выпускаются либо с объединенными анодами, либо с объединенными катодами. АЛС318 принадлежит к последним. Его анодами обычно управляет дешифратор типа КР514ИД1. Управление катодами можно организовать двумя способами — с использованием второго дешифратора или напрямую от микроконтроллера. Мы рассмотрим первый вариант — он требует использования меньшего числа выводов микроконтроллера.

Схема сопряжения нашего МК с индикатором АЛС318 приведена на рис. 32. Она включает в себя два дешифратора DD3 (КР514ИД1) и DD4 (К555ИД7) и половинку микросхемы с открытым коллектором DD2 (КР155ЛЛ2). Последняя, как будет показано ниже, управляет десятичной запятой. Вместо всех этих трех микросхем вполне возможно использовать запрограммиров анную соответствующим образом ПЛИС, которая «вберет» их в себя и будет выполнять те же функции. Но при этом потеряется ясность, почему именно так, а не иначе мы построили нашу программу связи МК с индикатором, и как изменить программу, если что-то изменено в схеме сопряжения. Поэтому я и рассматриваю схему на дискретных элементах, пусть даже несколько архаичную, но наиболее удобную для первоначального знакомства. А разобравшись с ней, вы будете делать то, что для вас легче, проще, элегантнее — когда знаешь, что и как сделать, придумать десяток вариантов на любой вкус несложно.

Итак, рассмотрим схему на рис. 24. Для работы с индикатором используется порт P1. Четыре его младшие линии (P1.0-P1.3) выводят на дешифратор DD3 код отображаемой цифры: 0000B — 0; 0001B — 1; 1001B — 9. Выходы дешифратора DD3 соединены с одноименными входами индикатора соответствующими анодами сегментов. Катоды сегментов каждого разряда, как уже упоминалось, объединены внутри индикатора и управляются выходами второго дешифратора DD4. На информационные входы последнего поступают сигналы с трех старших линий порта P1 — P1.5, P1.6 и P1.7. Они позволяют управлять индикатором, имеющим до 8 индицируемых разрядов. Оставшаяся линия (P1.4) используется для управления десятичной запятой — разрядом h индикатора. Установка этой линии в 1 зажигает запятую в том разряде, катодный вывод которого установлен в 0 соответствующим выходом дешифратора DD4.

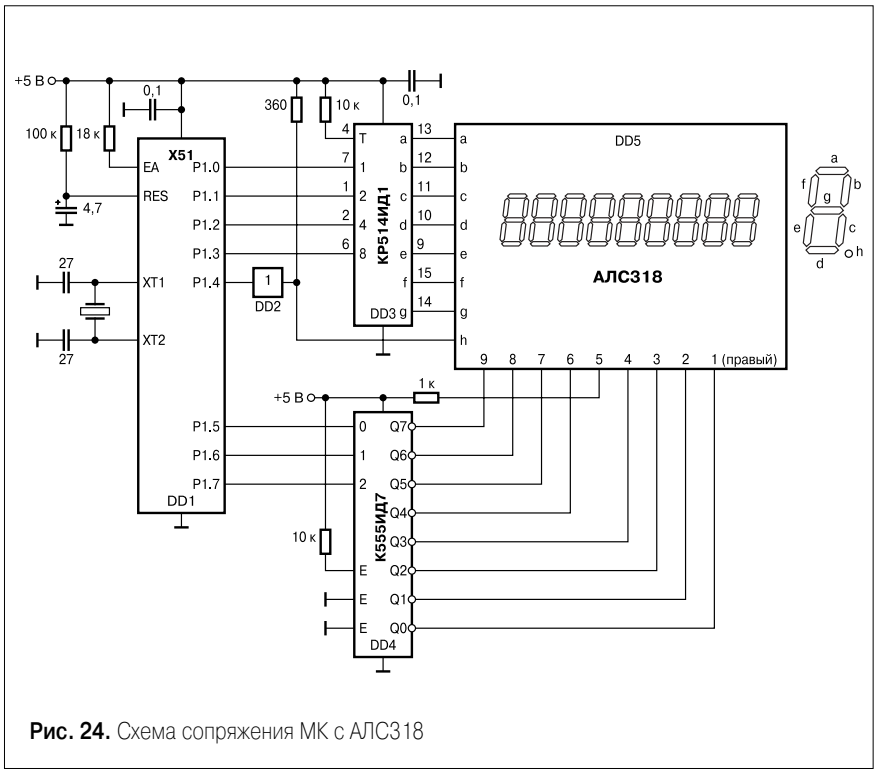

Рис. 24. Схема сопряжения МК с АЛС318

Как видите, для управления 8-разрядным 7-сегментным светодиодным индикатором нам понадобилось 8 линий вывода — весь порт P1. В предыдущем случае, при использовании HT1610, линии ввода/вывода использовались более экономно. Но ничего не поделаешь, это плата за отсутствие внутри индикатора АЛС318 дополнительного микроконтроллера.

Наверное, вы уже догадались, что если вам нужно управлять не 8-, а 16-разрядным индикатором, в качестве DD4 необходимо использовать дешифратор «4 в 16». Соответственно, для управления им понадобятся не 3, а 4 линии порта. Логичнее всего использовать для этого P1.4-P1.7, внеся соответствующие изменения в приведенную ниже программу. Ну, а управление десятичной запятой, если она вам необходима, придется осуществить по какой-либо линии другого порта, например, по P3.0.

Будем считать, что перед нами стоит та же задача, что и в предыдущем случае — отобразить подпрограммой IZOBR на индикаторе два 4-разрядных числа, хранящихся в двоично-десятичном представлении во внутреннем ОЗУ МК в ячейках памяти с адресами от AD00+3 (старший разряд первого числа) до AD00 (младший разряд). В данном примере, как и ранее, символическому адресу AD00 я присвоил численное значение 30H, AD00+1 — 31H, ... AD00+4 — это 34H, AD00+7 — 37H. Первое число я вывожу в четыре правых разряда индикатора, второе — в четыре левых, а в средний (индикатор-то 9-разрядный) вывожу пробел.

Таким образом, если ячейка с адресом 30Н содержит число 07Н, 31Н — число 02Н, 32Н — число 05Н, 33Н — число 04Н, а в следующих четырех ячейках с адреса 34Н по 37Н хранятся, соответственно, числа 08Н, 00Н, 01Н и 02Н, то при запуске подпрограммы IZOBR на индикаторе АЛС318 вы увидите (слева направо) числа 2108 и 4527, разделенные пробелом.

А как быть, если нам нужно отобразить не 2108 и 4527, а 2,108 и 45,27? Очень просто. Вспомним, что отображаемые цифры, хранящиеся в адресах с AD00+0 по AD00+7, — 4-битные (от 0000В до 1111В), а старшие четыре бита каждой из этих цифр хранят незначащие нули. Давайте договоримся, что пятый бит в каждой из ячеек памяти AD00+0-AD00+7 отвечает за десятичную запятую, идущую непосредственно за соответствующей отображаемой цифрой. В нашем примере в числе 2,108 запятая должна отображаться вместе с двойкой, то есть в ячейке с адресом 37Н должно храниться не 02Н, а 12Н. Соответственно, в числе 45,27 десятичная запятая идет после пятерки, то есть она хранится в виде единички в пятом по счету разряде числа в ячейке 32Н — там вместо 05Н должно находиться 15Н. В остальных же ячейках, хранящих цифры 1, 0 и 8 числа 2,108, и 4, 2, 7 числа 45,27, все четыре старших бита по-прежнему должны хранить незначащие нули.

Соответственно, при отображении на индикаторе той или иной цифры мы в соответствующей подпрограмме должны не только перенести ее из четырех младших битов ячейки AD00+n в четыре младших разряда порта P1 (P1.0-P1.3), но перенести также и пятый бит ячейки AD00+n в пятый разряд P1, т. е. в P1.4. Вот тогда мы сможем на индикаторе увидеть не только 2108 и 4527, но и 2,108 и 45,27.

И еще один момент, на который я хочу обратить ваше внимание. Как мы договорились, дешифратор DD4 может управлять восемью разрядами индикатора. В то же время АЛС318 — 9-разрядный, и мы хотим, чтобы средний (пятый с любого края) разряд был всегда погашен. Как погасить разряд, на управление которым у нас уже нет свободного выхода дешифратора? Правильно, подать на его катод единичный уровень. А если бы мы захотели погасить самый старший (левый) разряд индикатора, а в младших 8 разрядах отображать число от 00000000 до 99999999? Естественно, нужно было бы подать единичку на катод старшего разряда, а с выходами дешифратора соединить оставшиеся восемь катодов и при этом не забыть сделать соответствующие изменения в приводимой ниже программе.

Ну а теперь перейдем непосредственно к программе IZOBR:

```
;
AD00            .EQU
30H
CNSKIND         .EQU
40H       ;счетчик 0-255
;
;ПОДПРОГРАММА IZOBR ОБЕСПЕЧИВАЕТ
ВЫВОД ИНФОРМАЦИИ
;НА ЭКРАН 8-РАЗРЯДНОГО ДИСПЛЕЯ
АЛС318
;
IZOBR:
                MOV     CNSKIND,#0
IZOBR1:
                MOV     A,CNSKIND
                INC     A
                JZ      GASH
                MOV     CNSKIND,A
                ANL
A,#00000111B
                MOV     R0,A
                LCALL   DISPLEY
                SJMP    IZOBR1
GASH:   MOV     P1,#11101111B
                RET
;
;ПРИ ВХОДЕ В ЭТУ П/П R0=0...7 (СКАНИ-
;РОВАНИЕ). НАДО ПОСЧИТАТЬ AD00+R0,
;ПРОЧИТАТЬ ПО ЭТОМУ АДРЕСУ ЧИС-
ЛО,ПОМЕС-
;ТИТЬ ЕГО В МЛ. 5 БИТ P1, А СОДЕРЖИ-
МОЕ
;R0 — В СТАРШИЕ 3 БИТА
DISPLAY:
                MOV     A,#AD00
                ADD     A,R0
                MOV     R1,A
;R1=AD00+R0
                MOV     A,@R1
                ANL
A,#00011111B
                MOV     R1,A
                MOV     A,R0
                RR      A
                RR      A
                RR      A
                ADD     A,R1
                MOV     P1,A
                NOP
                NOP
                NOP
                NOP
                NOP
                NOP
                NOP
                NOP
                NOP
                NOP
        RET
```

Она реализует отображение методом динамической индикации. Напомню, что при этом одноименные (в нашем случае анодные) сегменты разных разрядов индикатора объединены между собой, и, вследствие этого, код отображаемого символа одновременно подается на аноды всех разрядов. Отображается же этот символ в том разряде, катоды которого находятся под нулевым потенциалом, при этом все остальные разряды индикатора оказываются погашены. Следовательно, если мы на дешифратор DD3 подадим код символа, который должен быть отображен в первом разряде, то для того, чтобы он высветился в нужном месте, на входы второго дешифратора (DD4) нужно подать код, при котором бы на его выходе, соединенном с катодом первого разряда, появился лог. 0. Далее, дав небольшую задержку, на DD3 подадим код символа, который должен отобразиться во втором разряде, а на входы DD4 — такой код, чтобы лог. 0 появился на его выходе, соединенном с катодом второго разряда. Процедуру будем повторять до тех пор, пока не переберем все разряды индикатора и не высветим в каждом из них соответствующую цифру, затем еще раз, еще, еще и т. д.

Обратимся снова к схеме на рис. 24. Заметьте, что крайний правый разряд я соединил с выходом 0 дешифратора DD4, второй справа — с выходом 1, третий — с выходом 2 и т. д. Это означает, что при отображении цифры, стоящей в самом правом разряде, на входах DD4 должен присутствовать код 000В, при отображении цифры во втором справа разряде — код 001В, в третьем справа — код 010В, в четвертом — 011В, в шестом (пятый отключен подачей на его катоды единицы) — 100В, в седьмом — 101В, в восьмом — 110В, и в девятом, крайнем слева — 111В. Вспомним также, что цифра, которая должна отобразиться в крайнем справа разряде, хранится в ячейке с адресом AD00+0, следующая — с адресом AD00+1, и т. д. Таким образом, для реализации отображения методом динамической индикации мы,

во-первых, должны где-то в микроконтроллере иметь счетчик, последовательно перебирающий значения 0, 1, 2, 3, 4, 5, 6, 7, 0, 1... Во-вторых, когда значение этого счетчика (назовем его счетчиком сканирования) равно 0, то нужно отображать цифру из ячейки AD00+0, когда 1 — из ячейки AD00+1, и т. д. Иными словами, если значение счетчика равно n, то отображаемая цифра должна быть извлечена из ячейки AD00+n. И в-третьих, вместе с отображаемой цифрой мы должны на входы дешифратора DD4 (т. е. на линии P1.5-P1.7) вывести текущее значение упомянутого счетчика. Вот, собственно, и все, что должна сделать наша программа.

Как это реализовано в IZOBR? В ячейке памяти CNSKIND организован счетчик, значение которого в начальный момент устанавливается в 0 командой MOV CNSKIND,#0. В ходе выполнения программы оно непрерывно увеличивается — команда MOV A,CNSKIND переносит его содержимое в аккумулятор, затем команда INC A увеличивает его на 1, а команда MOV CNSKIND,A возвращает инкрементированное значение обратно в ячейку CNSKIND. Идущая затем команда ANL A,#00000111B зануляет пять старших битов этого счетчика, точнее оставшейся в аккумуляторе его копии. Зачем? Если вы еще не догадались, поясню. После выполнения этой команды в аккумуляторе остается либо 0, либо 1, либо 2, либо 3, либо 4, либо 5, либо 6, либо 7. Никакой другой цифры там остаться не может. Чтобы осознать это, представьте, что до выполнения команды ANL A,#00000111B в нем была восьмерка. Вспомним, что 8 — это 00001000B. Занулите ее 5 старших бит, и получите 0. От девятки (00001001B) после зануления 5 старших бит останется 1, от десятки (00001010B) — двойка, и т. д., до 15 (00001111B), которые при этом оставят семерку. А вот от идущего после 15 числа 16 (00010000B) — (обратите на это внимание) — зануление 5 старших бит оставит снова 0. От 17 останется единица, от 18 — двойка, и т. д.

Таким образом, в результате выполнения вышеупомянутых команд в аккумуляторе последовательно, друг за другом, оказываются 0, 1, 2, 3, 4, 5, 6, 7. Затем эти цифры снова повторяются в той же последовательности, затем еще и еще, пока не завершится выполнение программы. А ведь именно это нам и нужно! Далее мы должны это хранящееся в аккумуляторе число перенести в три старших разряда порта P1 и помимо этого прибавить его к AD00; найденную сумму мы возьмем в качестве адреса ячейки, откуда извлечем отображаемую цифру (она находится в младших четырех битах этой ячейки) и выведем ее в четыре младших разряда P1. Вспомним еще, что в пятом бите упомянутой ячейки, как мы договаривались, хранится разряд десятичной запятой. Его нужно вывести на ту линию порта, которая управляет сегментом h индикатора, то есть в P1.4 (см. программу).

Рис. 25. Последовательность выполнения трех команд RR A

Полученное после выполнение команды ANL A,#00000111B значение реализованного программным путем счетчика мы сохраняем в регистре R0 (команда MOV R0,A) и затем вызываем подпрограмму DISPLAY. Последняя загружает в аккумулятор число #AD00 и командой ADD A,R0 суммирует его со значением счетчика, хранимого в R0. Полученная сумма сохраняется в регистре R1 (MOV R1,A).

Далее идет пока еще не очень нам знакомая команда MOV A,@R1. Что она делает? Напомню, что в предыдущей главе, рассматривая регистр DPTR, мы узнали, что команда MOVX A,@DPTR предписывает микроконтроллеру прочитать в аккумулятор данные из ячейки внешней памяти, адрес которой хранится в DPTR. Вспомнив это, вы догадаетесь, что MOV A,@R1 заставит наш МК прочитать в аккумулятор данные из ячейки памяти, адрес которой хранится ... где? Правильно, в R1. Вспомните также, что при рассмотрении команды MOVX A,@DPTR я отметил, что X на конце команды MOVX говорит о том, что чтение должно осуществляться из внешней памяти данных. В команде же MOV A,@R1 этого X нет, что говорит о том, что чтение должно быть осуществлено из внутренней памяти данных. Следовательно, MOV A,@R1 предписывает МК найти, какой адрес хранится в регистре R1, после чего перенести в аккумулятор данные из той ячейки внутреннего ОЗУ, адрес которой найден в R1. Попутно отмечу, что команда MOV @R1,A вынуждает МК совершить обратное действие — перенести данные из аккумулятора в ячейку внутренней памяти, адрес которой хранится все в том же R1. Кстати, подобный метод адресации, когда адрес ячейки памяти, участвующей в обмене данными, находится в каком-либо регистре, носит название косвенной адресации (адрес мы находим косвенно, при помощи R1 или DPTR), в отличие от прямой адресации (например, MOV A,R1), где адрес (регистр R1) в явном виде указан в команде.

Теперь вернемся к нашей программе, от которой мы слегка отвлеклись. Чуть раньше мы нашли адрес ячейки памяти, где хранится цифра, соответствующая значению счетчика сканирования (счетчика, последовательно перебирающего значения 0, 1, 2, 3, 4, 5, 6, 7, 0, 1...). Этот адрес, равный AD00+n, мы сохранили в R1. Затем мы вызвали команду MOV A,@R1. Думаю, что всем, читающим эти строки, очевидно, что после выполнения последней в аккумуляторе будет находиться та самая цифра, которую нам надо отобразить. Но цифра эта хранится в четырех младших битах аккумулятора, а в пятом хранятся 1 или 0, зажигающие или гасящие десятичную запятую после отображаемой цифры. Три старших бита пока не несут полезной информа-

Аккумулятор	a2	a1	a0	0	0	0	0	0
Регистр R1	0	0	0	b4	b3	b2	b1	b0
Аккумулятор	a2	a1	a0	b4	b3	b2	b1	b0

Рис. 26. Сложение аккумулятора и регистра R1

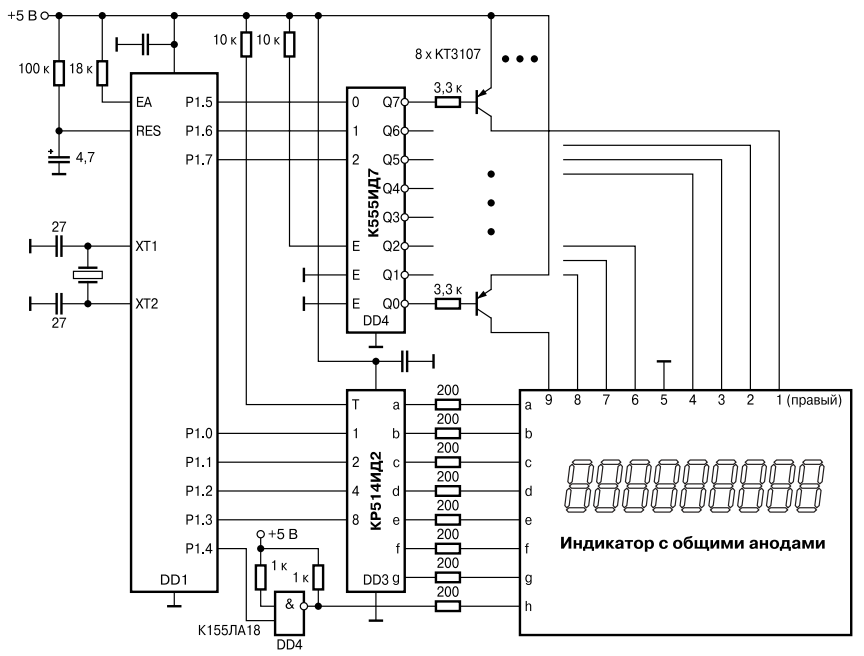

Рис. 27. Схема сопряжения с индикатором с общими анодами

ции. Занулим их для надежности командой ANL A,#00011111B , после чего разместим в этих старших трех битах значение счетчика сканирования (напомню, что оно хранится в R0 и в нашем случае равно 000, 001, 010, ..., 110 или 111).

Операция занесения счетчика сканирования в старшие биты аккумулятора также довольно проста. Сначала мы сохраняем значение аккумулятора все в том же регистре R1 командой MOV R1,A. Взамен этого числа мы переносим в аккумулятор счетчик сканирования из R0 командой MOV A,R0. Зачем? Да потому что значение счетчика сканирования хранится в трех младших битах R0, а нам нужно перенести его в три старших бита. А подобное преобразование мы можем сделать только в аккумуляторе, в связи с чем нам и пришлось его вначале освободить, а затем перенести в него значение счетчика сканирования.

Перенос из младших битов в старшие мы осуществим при помощи трех команд циклического сдвига аккумулятора вправо (RR A). При выполнении этой команды старший, седьмой бит аккумулятора, переместится в шестой, шестой переедет в пятый, и т. д., вплоть до первого. Первый же переместится на место нулевого бита, а как бы «вытолкнутый» из аккумулятора нулевой бит пересылается на место старшего, седьмого. Сказанное поясняет рис. 25, показывающий состояние аккумулятора до выполнения команды RR A, после выполнения первой из них, затем второй и третьей. Нетрудно убедиться, что, применив ее три раза, мы добьемся желаемого эффекта — счетчик сканирования из трех младших бит переместится в три старших. При этом в пяти младших битах будут находиться нули.

Теперь нам осталось только сложить содержимое аккумулятора и регистра R1. Делается это при помощи уже знакомой нам команды ADD A,R1. Нули в трех старших разрядах R1 не исказят при сложении с аккумулятором хранящиеся в его старших трех разрядах биты счетчика сканирования. Пять младших разрядов аккумулятора, хранивших нули, при сложении с пятью битами R1, содержащими отображаемую цифру, в результате суммирования будут именно ее и хранить (рис. 26). После этого нам ничего не остается, как вывести это число в порт P1 командой MOV P1,A, и требуемая цифра загорится в нужном разряде АЛС318. Дав ей погореть какое-то время (пока МК будет выполнять поставленные для реализации задержки команды NOP), контроллер осуществит возврат из подпрограммы DISPLAY, инкрементирует счетчик сканирования и приступит к отображению следующей цифры.

А как быть, если мы решили использовать индикаторы не с общими катодами, а с общими анодами? Схема сопряжения с таким индикатором приведена на рис. 27. Нам нужно лишь заменить дешифратор 514ИД1 на 514ИД2 и управлять общими анодами не напрямую с дешифратора DD4, а через буферные транзисторы. Поскольку информация для дешифратора DD3 по-прежнему выводится через младшие четыре линии порта P1, а для DD4 — через его старшие три линии, как и в схеме, рассмотренной на рис. 24, то для схемы на рис. 27 годится та же программа, что и для рис. 24, без каких-либо изменений.

Последний рассматриваемый в качестве примера случай — сделать 16-разрядное табло из одиночных индикаторов, например с общими катодами. Фактически эта задача почти полностью сводится к случаю, который проиллюстрирован на рис. 24. Объединим, как показано на рис. 28, у всех одиночных индикаторов одноименные аноды a, b, c, d, e, f, g, а для управления (необъединяемыми!) 16 ка-

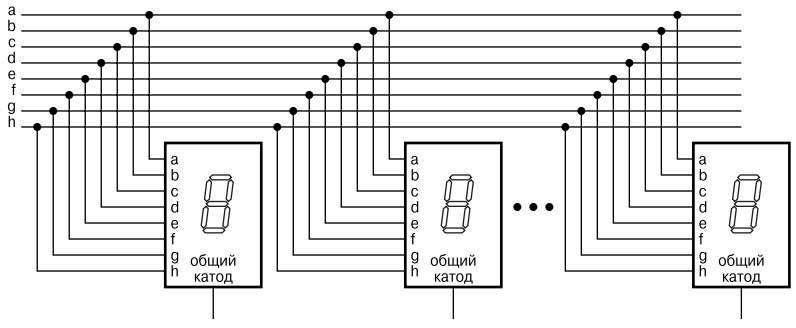

Рис. 28. Многоразрядный индикатор из отдельных одноразрядных

тодами используем дешифратор 155ИД3 (4 в 16). Как говорилось выше, для управления последним дешифратором используем линии P1.4-P1.7 порта P1, а для десятичной запятой — линию P3.0. В остальном же схема остается без изменений.

Программа, управляющая этим 16-разрядным табло, приведена ниже. Она аналогична той, что рассмотрена нами чуть выше, но учитывает отличия в аппаратной части — использование 16-разрядного индикатора и управление десятичной запятой по другой линии порта. Комментировать эту программу я не буду — вам должно быть вполне по силам самостоятельно найти отличия в программах и разобраться, чем вызваны эти отличия.

```
   ;
   AD00              .EQU
30H
   CNSKIND           .EQU
40H         ;счетчик 0-255
   ;
   ;ПОДПРОГРАММА IZOBR ОБЕСПЕЧИВАЕТ
ВЫВОД 16 ЦИФР ИЗ AD00+0...
   ;...AD00+15   НА  ЭКРАН  16-РАЗРЯДНОГО
ДИСПЛЕЯ
   ;
   IZOBR:
                     MOV       CNSKIND,#0
   IZOBR1:
                     MOV       A,CNSKIND
                     INC       A
                     JZ        GASH
                     MOV       CNSKIND,A
                     ANL
A,#00001111B
                     MOV       R0,A
                     LCALL     DISPLEY
                     SJMP      IZOBR1
   GASH:  MOV        P1,#11111111B
                     CLR       P3.0
                     RET
   ;
   ;ПРИ ВХОДЕ В ЭТУ П/П R0=0...15 (СКАНИ-
   ;РОВАНИЕ).НАДО ПОСЧИТАТЬ AD00+R0,
   ;ПРОЧИТАТЬ ПО ЭТОМУ АДРЕСУ
   ЧИСЛО,ПОМЕС-
   ;ТИТЬ ЕГО В МЛ. 4 БИТА P1, А
   СОДЕРЖИМОЕ
   ;R0 — В СТАРШИЕ 4 БИТА. ЗАПЯТУЮ В P3.0
   ;
   DISPLAY:
                     MOV       A,#AD00
                     ADD       A,R0
                                         MOV       R1,A
                  ;R1=AD00+R0
                                         MOV       A,@R1
                                         MOV       C,ACC.4
                                         MOV       P3.0,C
                                   ANL   A,#00001111B
                                   MOV   R1,A
                                         MOV       A,R0
                                         RR        A
                                   RR    A
                                         RR        A
                                         RR        A
                                         ADD       A,R1
                                         MOV       P1,A
                                         NOP
                                         NOP
                                         NOP
                                         NOP
                                         NOP
                                         NOP
                                         NOP
                                         NOP
                                         NOP
                                   RET
```

Александр Фрунзе,
alex.fru@dian.ru

Продолжение следует