


**Федеральное государственное бюджетное образовательное  
учреждение высшего образования "Белгородский государственный  
технологический университет им. В.Г. Шухова"**

Кафедра программного обеспечения вычислительной техники и  
автоматизированных систем.

**Лабораторная работа работа № 1**  
Подключение внешней памяти и ее тестирование.  
Вариант 13

Выполнил:  
Студент группы КБ-211

  
\_\_\_\_\_ Коренев Д.Н.

Принял:

\_\_\_\_\_ Шамраев А.А.

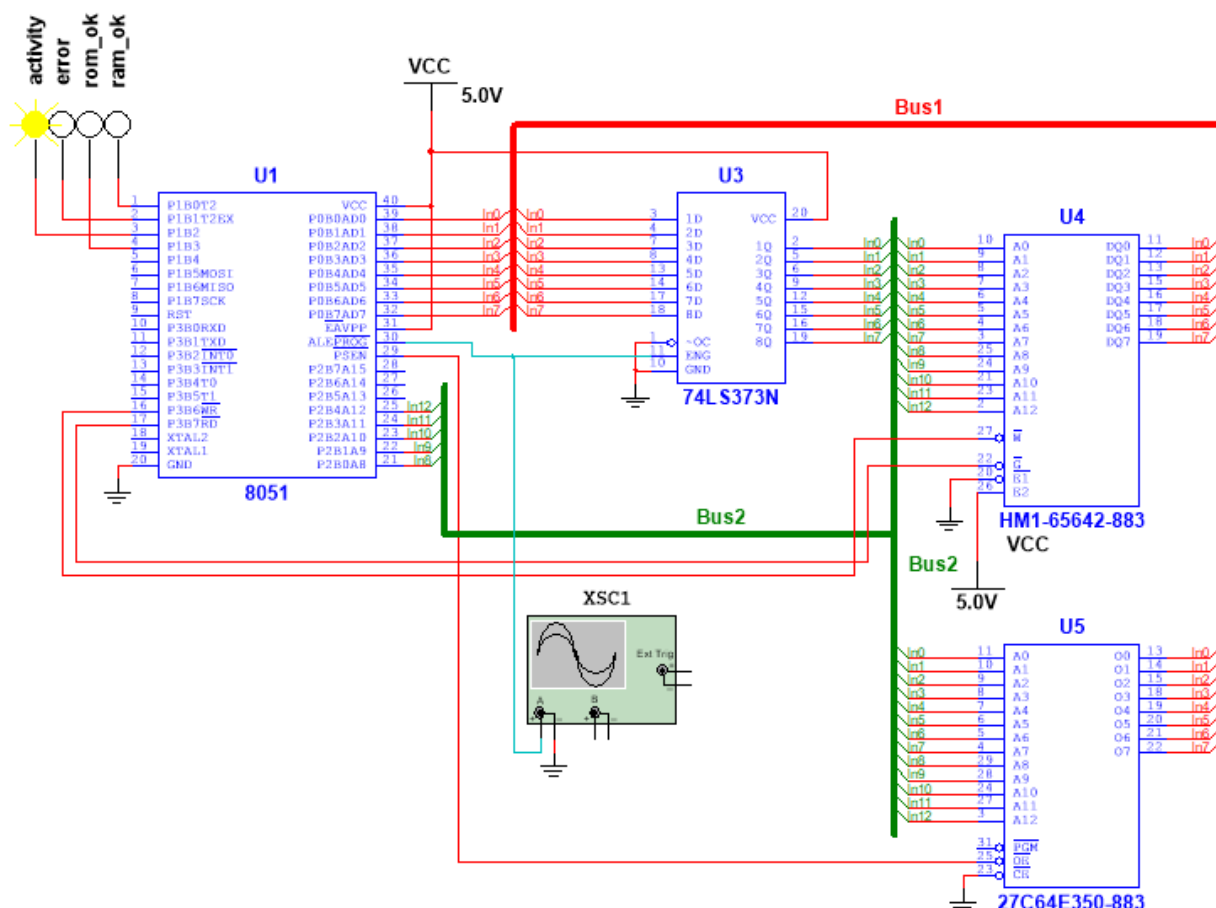
*Цель работы:* разработать схему подключения микроконтроллера с внешней памятью и протестировать память.

## Задание

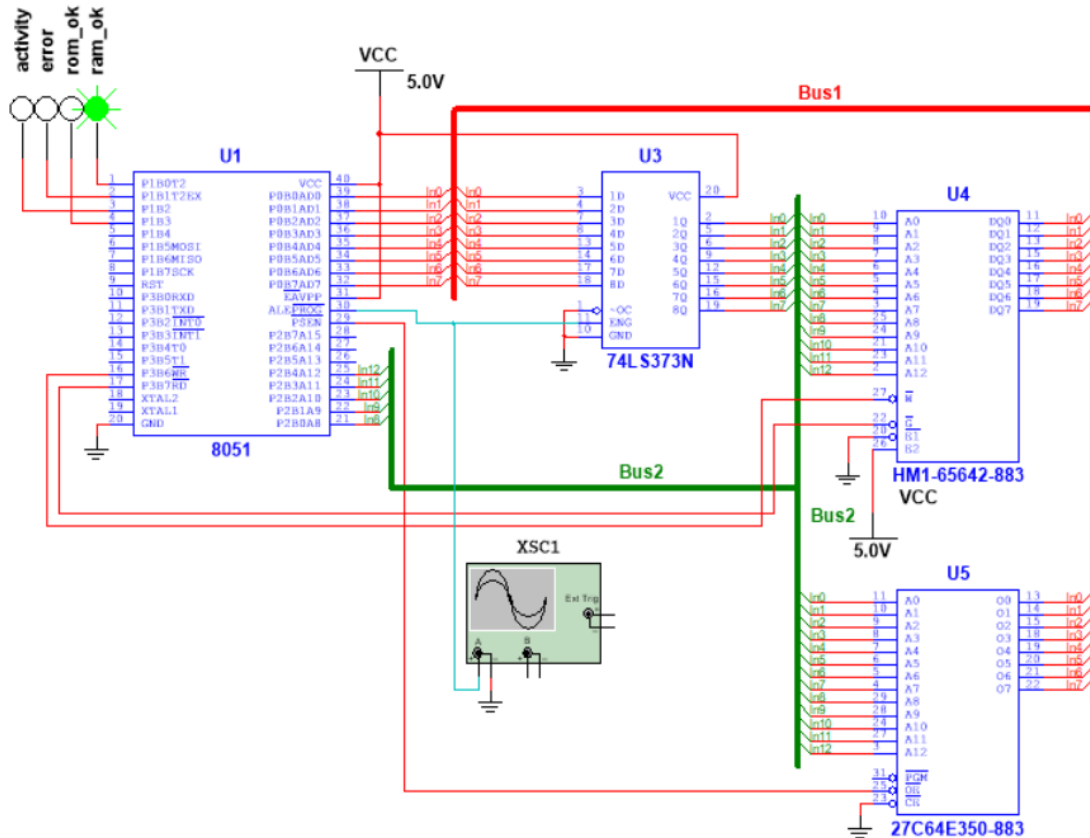
1. Подключить к МК внешнее ОЗУ емкостью N кбайт (табл. 2.9) и используя тестовый набор XX, произвести тестирование области памяти 1 кбайт, начиная с адреса ZZZ. Включить светоиндикатор, если записанный и считанный из ячейки памяти наборы не совпадают. Подключить осциллограф, как показано на рис. 2.28, и снять осциллограмму с вывода 0 ALE МК.

Параметры	1
N, кбайт	8
XX	0AAh
ZZZ	800h

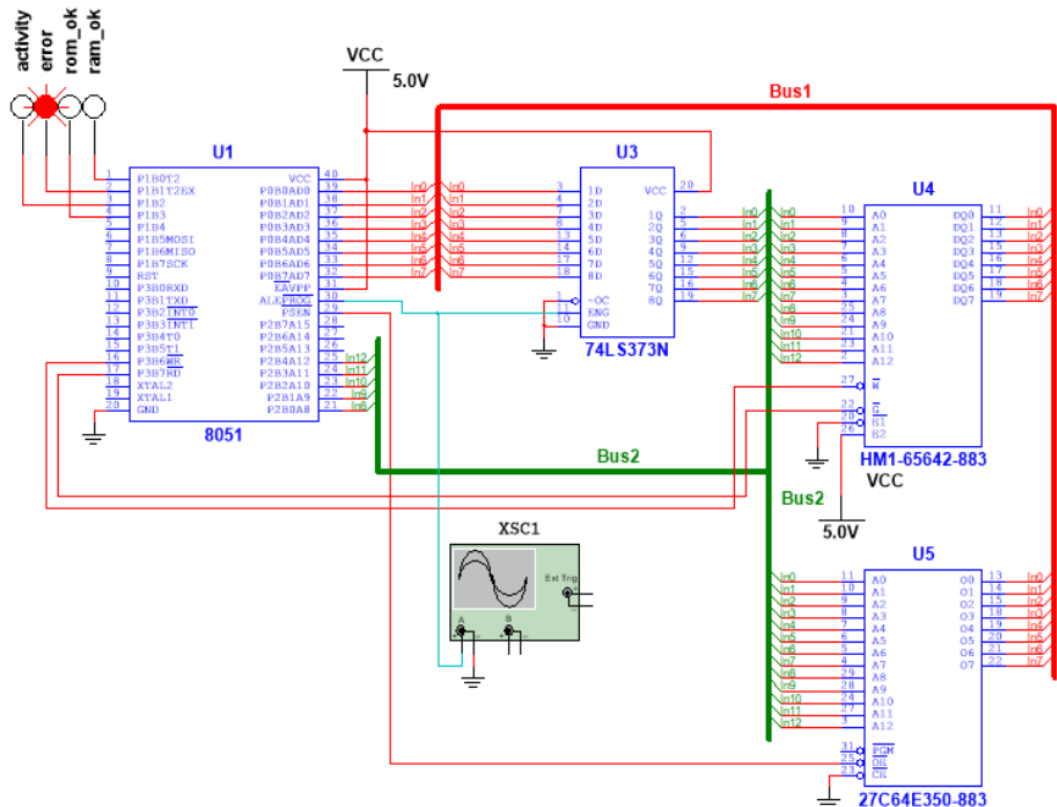
Выполняется тестирование (мигает желтая лампа):

[illegible]

Тест ОЗУ пройден успешно:

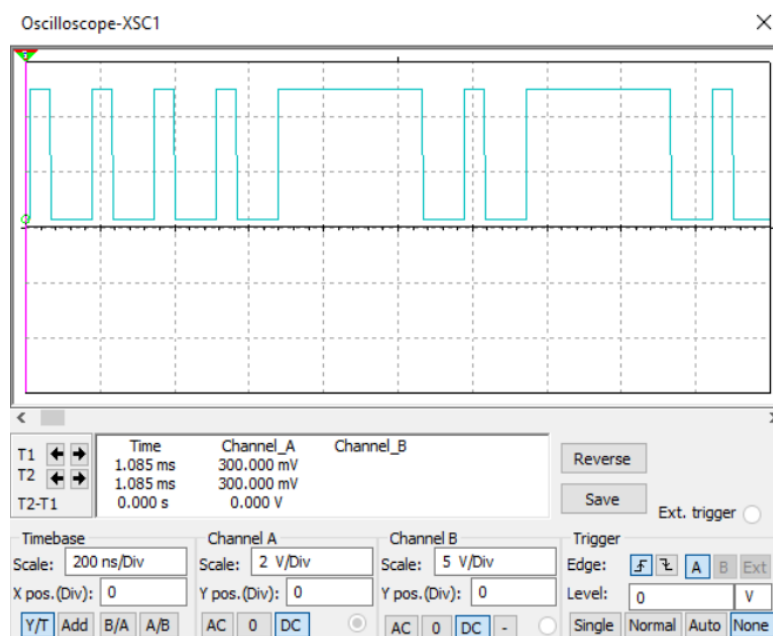


Искусственно сгенерировал ошибку:



07C0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
07E0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0800	AA	AA	AA	AA	AA	AA	AA	AA	55	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0820	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0840	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

Осциллограмма:



Код программы (с продолжением):

```

1  $MOD51
2
3  org 00h
4  test_ram:
5      mov dptr, #800h          ; Указатель на начало RAM для проверки
6      mov r7, #4              ; Кол-во повторений внешнего цикла
7      mov r6, #0              ; Кол-во повторений внутреннего цикла
8                              ; (кол-во байт в RAM для проверки)
9                              ; 4*256 = 1024 байт
10     anl p1, #00h            ; Очищаем порт P1 (все лампочки выключены)
11 loop_ram:
12     mov A, #0AAh            ; Записываем в аккумулятор 0xAA
13     movx @dptr, A            ; Записываем в RAM 0xAA
14     movx A, @dptr            ; Читаем из RAM в аккумулятор
15     xrl A, #0AAh            ; Сравниваем с 0xAA
16     jnz error                ; Если не равно, то ошибка
17     inc dptr                 ; Увеличиваем указатель RAM на 1
18     setb p1.2                ; Мигаем лампочкой активности
19     clr p1.2
20     djnz r6, loop_ram         ; Повторяем внутренний цикл 256 раз
21     djnz r7, loop_ram         ; Повторяем внешний цикл 4 раза
22     setb p1.0                ; Все хорошо, горит зеленая лампочка

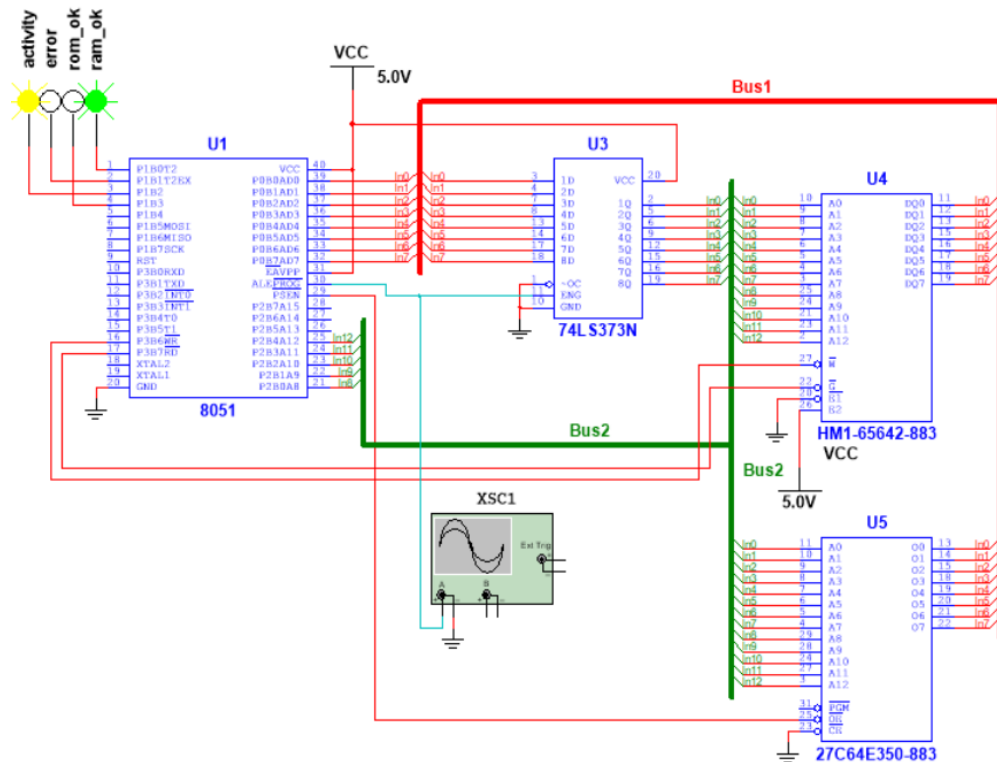
```

2. Подключить к МК внешнее ПЗУ емкостью N Кбайт (табл. 2.10). Подсчитать контрольную сумму области памяти емкостью Kбайт, начиная с адреса ZZZ и сравнить ее с константой, находящейся в резидентном ПЗУ. Если сравниваемые величины совпадают, тестирование ПЗУ прошло успешно,

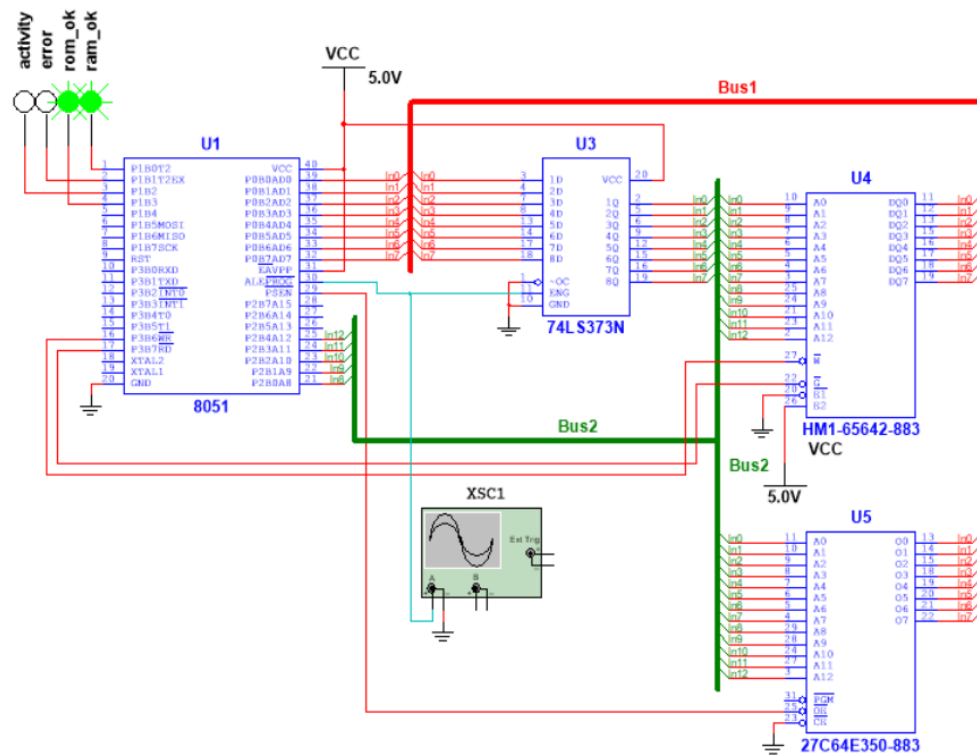
иначе включить светоиндикатор. Подключить осциллограф, как показано на рис. 2.28, и снять осциллограмму с вывода ALE МК.

Параметры	7
N, Кбайт	8
K, байт	100
ZZZ	800h

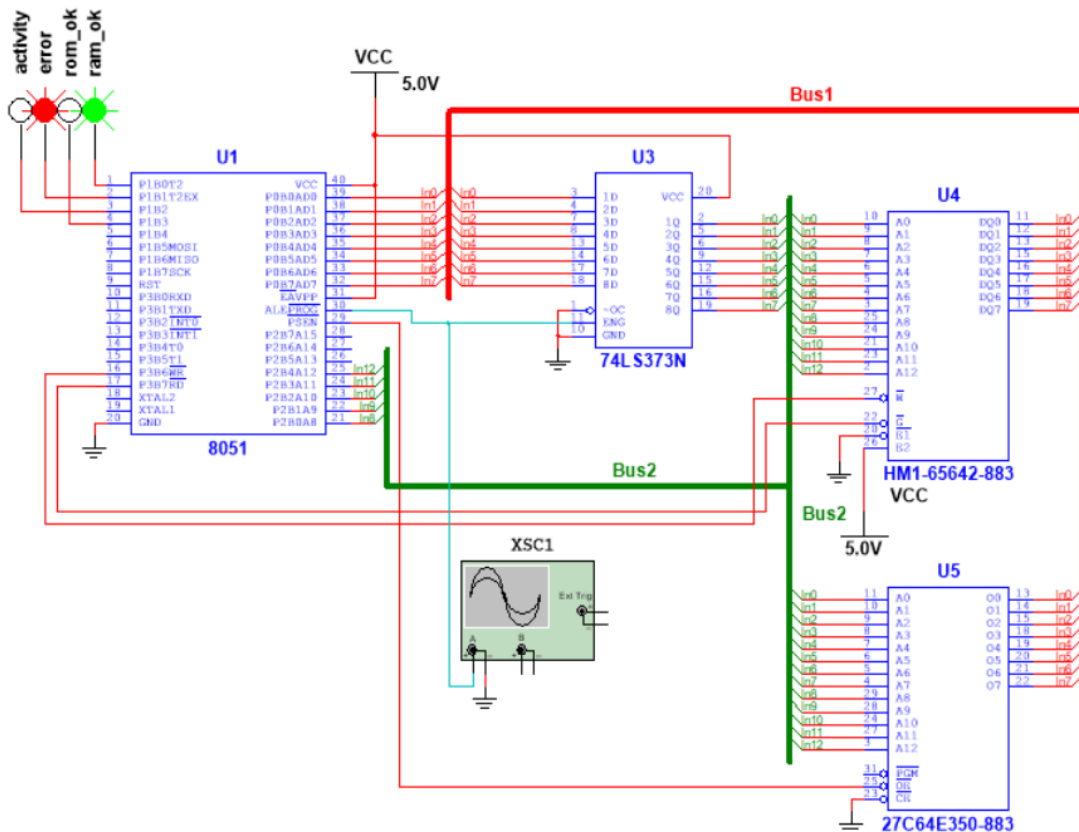
Выполняется тестирование (мигает желтая лампа):



Тест ПЗУ пройден успешно:



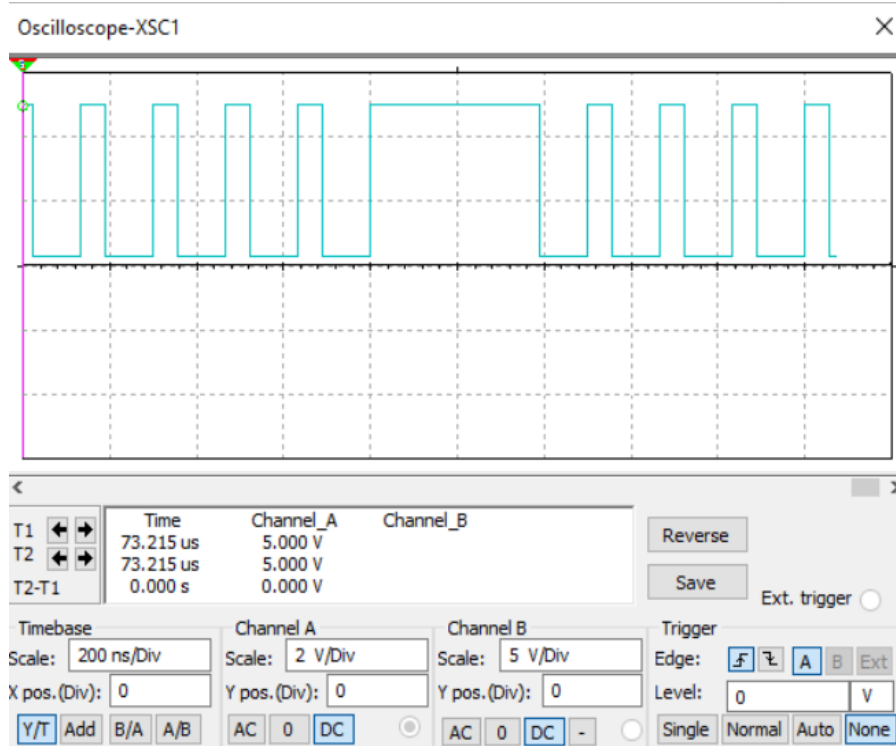
Искусственно сгенерировал ошибку:



```
clr pl.2
djnz r6, loop_rom
cjne r7, #63h, error
setb pl.3
sjmp $
```

, при 64h – правильное значение контрольной суммы

## Осциллограмма:



## Код программы (продолжение):

```

23 test_rom:
24     mov dptr, #800h           ; Указатель на начало ROM для проверки
25     mov r7, #0                ; Счетчик суммы
26     mov r6, #64h              ; Кол-во циклов (кол-во байт в ROM для проверки)
27 loop_rom:
28     clr A                     ; A = 0
29     movc A, @A+dptr           ; Читаем из ROM в аккумулятор
30     add A, r7                 ; Складываем с суммой
31     mov r7, A                 ; Записываем результат в r7
32     inc dptr                  ; Увеличиваем указатель ROM на 1
33     setb p1.2                 ; Мигаем лампочкой активности
34     clr p1.2
35     djnz r6, loop_rom         ; Повторяем цикл 100 раз
36     cjne r7, #64h, error      ; Если сумма не равна 64h (контрольная сумма),
37                               ; то ошибка
38     setb p1.3                 ; Все хорошо, горит зеленая лампочка
39     sjmp $                    ; Бесконечный цикл
40
41 error:
42     setb p1.1                 ; Ошибка, горит красная лампочка
43     sjmp $                    ; Бесконечный цикл
44
45 org 800h                     ; ROM
46 db 41, 35, 190, 132, 225, 108, 214, 174, 82, 144
47 db 73, 241, 241, 187, 233, 235, 179, 166, 219, 60
48 db 135, 12, 62, 153, 36, 94, 13, 28, 6, 183
49 db 71, 222, 179, 18, 77, 200, 67, 187, 139, 166
50 db 31, 3, 90, 125, 9, 56, 37, 31, 93, 212
51 db 203, 252, 150, 245, 69, 59, 19, 13, 137, 10
52 db 28, 219, 174, 50, 32, 154, 80, 238, 64, 120
53 db 54, 253, 18, 73, 50, 246, 158, 125, 73, 220
54 db 173, 79, 20, 242, 68, 64, 102, 208, 107, 196
55 db 48, 183, 50, 59, 161, 34, 246, 34, 145, 157
56
57 END

```

## **Вывод:**

Цель работы успешно достигнута. В ходе выполнения задачи были выполнены следующие шаги:

1. Создана схема подключения микроконтроллера к внешней памяти, включая оперативную (ОЗУ) и постоянную (ПЗУ) память.
2. Разработана программа для тестирования памяти, которая загружена в постоянную память (ПЗУ) микроконтроллера.
3. Проведено тестирование памяти, включая оперативную (ОЗУ) и постоянную (ПЗУ) память. Тестирование выполнено успешно, и ошибки не обнаружены.
4. Результаты тестирования проанализированы, и подтверждено, что внешняя память работает без сбоев.



## Контрольные вопросы:

1. Какие функции выполняет регистр-защелка при подключении к МК внешней памяти?

Регистр-защелка (latch) при подключении внешней памяти к микроконтроллеру MCS-51 (или любому другому микроконтроллеру) выполняет следующие функции:

- 1) Буферизация данных: Регистр-защелка может использоваться для временного хранения данных, передаваемых между микроконтроллером и внешней памятью. Это позволяет более эффективно управлять потоком данных и синхронизировать их передачу.
- 2) Управление сигналами: Регистр-защелка может использоваться для формирования или преобразования сигналов управления, таких как сигналы адреса и управления памятью. Это позволяет адаптировать сигналы, генерируемые микроконтроллером, для совместимости с интерфейсом внешней памяти.
- 3) Уменьшение нагрузки на шины: Использование регистра-защелки может уменьшить нагрузку на шины данных и адреса, так как данные могут быть временно сохранены в регистре, прежде чем быть переданными на шины. Это помогает в управлении таймингами и минимизации возможных конфликтов на шинах.
- 4) Синхронизация данных: Регистр-защелка может использоваться для синхронизации данных между микроконтроллером и внешней памятью, что помогает избегать ошибок при передаче данных.

2. Назовите альтернативную функцию, выполняемую портом P2.

Порт P2 на микроконтроллерах семейства MCS-51 имеет альтернативную функцию. В основном, P2 используется как порт ввода/вывода, но также может выполнять функцию внешнего счетчика/таймера.

В качестве внешнего счетчика/таймера P2 может быть использован для различных операций, таких как генерация синхросигналов, задержки, измерения времени и других временных задач. Это позволяет микроконтроллеру выполнять задачи тайминга и управления временем.

Для переключения порта P2 между режимами ввода/вывода и внешнего счетчика/таймера, требуется настройка специальных битов в регистрах управления таймерами и портов ввода/вывода.

3. Объясните назначение сигнала PSEN (37 вывод МК-51).

Сигнал PSEN (Program Store Enable) на микроконтроллерах семейства MCS-51 (например, 8051) имеет следующее назначение:

- 1) Чтение программы из внешней памяти: PSEN используется для управления доступом микроконтроллера к внешней памяти, в которой хранится программа (код) для выполнения. Когда микроконтроллер нуждается в чтении инструкции (команды) из памяти для выполнения следующей операции, он активирует сигнал PSEN.
- 2) Генерация сигнала чтения: PSEN является сигналом чтения, который передается внешней памяти. При активации PSEN внешняя память должна быть настроена на предоставление данных (инструкций) микроконтроллеру. Этот сигнал позволяет внешней памяти понимать, что микроконтроллер хочет прочитать данные.
- 3) Использование внешней памяти: PSEN позволяет микроконтроллеру использовать внешнюю память для хранения программы вместо внутренней памяти. Это особенно полезно в случаях, когда требуется большой объем памяти для хранения программы.
- 4) Обеспечение исполнения кода: Активация PSEN позволяет микроконтроллеру получать инструкции из внешней памяти и, следовательно, выполнять программу, хранящуюся в этой памяти.

4. Поясните, в чем отличия в работе процессора при обращении к внешней памяти данных по сравнению с резидентной?

Работа процессора при обращении к внешней памяти данных (например, внешней RAM или внешней Flash-памяти) отличается от работы с резидентной (внутренней) памятью в нескольких аспектах:

- 1) Скорость доступа: Внутренняя память, как правило, имеет более высокую скорость доступа по сравнению с внешней памятью. Процессоры обычно имеют кэшированную внутреннюю память, что позволяет им получать данные из внутренней памяти гораздо быстрее, чем из внешней.
- 2) Ширина шины данных: Часто внутренняя память имеет более широкую шину данных, что позволяет процессору одновременно читать или записывать больше данных. Внешняя память может иметь меньшую ширину шины данных, что делает доступ к данным медленнее.
- 3) Доступность: Внутренняя память всегда доступна для процессора, и он может ее использовать без каких-либо задержек. Внешняя память, с другой стороны, может иметь задержки в доступе из-за физических характеристик, таких как время доступа к Flash-памяти или задержки при доступе к удаленной RAM.

- 4) Управление: Обращение к внутренней памяти происходит стандартным образом с использованием адресов, которые генерирует процессор. В случае внешней памяти может потребоваться дополнительное управление и сигнализация для выбора и доступа к нужной области внешней памяти.
- 5) Сложность программирования: Работа с внешней памятью требует более сложной логики программирования, чем работа с внутренней памятью. Это связано с необходимостью управления адресами внешней памяти, обработкой возможных ошибок доступа и другими аспектами.
- 6) Размер памяти: Внутренняя память ограничена размером, который встроен в процессор, в то время как внешнюю память можно расширить или заменить с большими объемами, что делает ее более подходящей для приложений, требующих большого объема данных.

#### 5. Каково назначение вывода EA/Vpp МК?

Вывод EA/Vpp (External Access/Voltage Programming) на микроконтроллерах семейства MCS-51, таких как 8051, имеет двойное назначение:

- 1) Режим внешнего доступа (EA): Если вывод EA/Vpp находится в состоянии "0" (заземлен), микроконтроллер будет использовать внутреннюю программную память для выполнения кода. Это означает, что микроконтроллер будет исполнять программу из своей встроенной Flash-памяти или ROM (в зависимости от конкретной модели).
- 2) Режим программирования напряжения (Vpp): Если вывод EA/Vpp находится в состоянии "1" (подключен к высокому напряжению), микроконтроллер переходит в режим программирования. Это означает, что вы можете программировать (записывать) внутреннюю Flash-память микроконтроллера с использованием специального программатора и соответствующего программного обеспечения. В этом режиме вы можете обновлять программное обеспечение микроконтроллера, загружая новый код в его память.

6. Каким образом можно использовать внешнее ОЗУ для обращения и к данным и к программе?

- 1) Данные во внешней памяти: Можно использовать внешнее ОЗУ для хранения данных, которые будут обрабатываться вашей программой. Для этого нужно правильно настроить адресацию и доступ к данным во внешней памяти. Процессор MCS-51 имеет команды для чтения и записи данных во внешнюю память. Нужно определить адреса и

местоположение данных во внешней памяти и использовать эти команды для доступа к данным.

- 2) Переадресация данных: Можно использовать внутренние регистры для временного хранения данных, прежде чем записать их во внешнее ОЗУ, или наоборот, считать данные из внешней памяти и сохранить их во внутренних регистрах для обработки. Это позволяет более эффективно управлять данными между внутренней и внешней памятью.
- 3) Загрузка программы: Как уже упоминалось, программа (код) должна храниться во внутренней программной памяти микроконтроллера. Мы не можем выполнять программу напрямую из внешней памяти. Однако мы можем организовать механизм загрузки программы из внешней памяти во внутреннюю, если это необходимо. Это может потребовать специфических процедур, включая копирование кода из внешней памяти во внутреннюю при инициализации микроконтроллера.<sup>7</sup> Как подсчитывается контрольная сумма заданной области памяти?

8. Объясните назначение выводов CE (CS), OE микросхем памяти.

Выводы CE (Chip Enable) или CS (Chip Select) и OE (Output Enable) на микросхемах памяти имеют следующие назначения:

- 1) CE (Chip Enable) или CS (Chip Select): Этот вывод используется для выбора конкретной микросхемы памяти на автономной шине данных и адреса. Когда сигнал CE/CS активен (обычно в низком состоянии), это означает, что микросхема памяти активирована и готова к выполнению операций чтения или записи данных. Когда CE/CS неактивен (обычно в высоком состоянии), микросхема находится в отключенном состоянии, и никакие операции с памятью не выполняются.
- 2) OE (Output Enable): Этот вывод управляет состоянием выходных данных микросхемы памяти. Когда сигнал OE активен (обычно в низком состоянии), микросхема разрешает выход данных, и данные могут быть считаны с выходных выводов микросхемы. Когда OE неактивен (обычно в высоком состоянии), выходные данные отключены, и микросхема не выводит данные на шину данных.

9. Каковы конструктивно-технологические отличия однократнопрограммируемых микросхем ПЗУ (PROM) по сравнению с репрограммируемыми ПЗУ (EPROM – с ультрафиолетовым стиранием)?

- 1) Способ программирования:

- PROM: Однократнопрограммируемые микросхемы ПЗУ программируются во время производства с помощью специального устройства, которое применяет напряжение

источника тока (высокого напряжения) к выбранным ячейкам памяти, чтобы "прожечь" биты в них. Программирование PROM является необратимым процессом, и данные нельзя стереть или изменить после записи.

- EPROM: Репрограммируемые микросхемы EPROM также записываются с помощью высокого напряжения (ультрафиолетового света), но в отличие от PROM, они могут быть стерты и перезаписаны. Для стирания данных в EPROM необходимо выставить микросхему на ультрафиолетовый свет (обычно под ультрафиолетовой лампой) в течение некоторого времени. После этого данные могут быть заново записаны в EPROM.

## 2) Устойчивость данных:

- PROM: Данные, записанные в PROM, являются стабильными и не могут быть изменены или стерты. Это делает PROM подходящими для хранения постоянной информации, которая не должна изменяться в процессе эксплуатации.
- EPROM: Данные в EPROM могут быть стерты и перезаписаны, что делает их более гибкими и позволяет обновлять программное обеспечение или данные в микросхеме. Однако EPROM менее устойчивы к излучению ультрафиолетового света и могут потерять данные при длительном воздействии ультрафиолетовых лучей.

## 3) Затраты на производство:

- PROM: Изготовление PROM более дешево и проще, поскольку их можно запрограммировать только один раз, и это не требует сложных устройств для стирания и перезаписи.
- EPROM: EPROM требуют более сложного производственного процесса, который включает в себя добавление окна кварцевого оксида для ультрафиолетового стирания, что делает их более дорогими.

10. Поясните особенности тестирования микросхем ОЗУ по сравнению с ПЗУ.

## 1) Цель тестирования:

- ОЗУ: Основной целью тестирования ОЗУ является проверка корректности операций чтения и записи данных. Тесты направлены на обнаружение ошибок в считывании и записи

данных в ячейки памяти, а также на поиск сбоев в адресации и временных характеристиках памяти.

- ПЗУ: В случае ПЗУ, главной целью тестирования является проверка целостности и точности хранения данных в памяти. Это включает в себя убеждение в том, что данные, записанные в ПЗУ, могут быть корректно считаны в будущем без ошибок.

## 2) Обнаружение ошибок:

- ОЗУ: Тесты ОЗУ могут обнаруживать ошибки в операциях чтения и записи, такие как сбои в ячейках памяти, проблемы с аппаратными линиями данных и адресов, а также проблемы с временными характеристиками памяти.
- ПЗУ: Тестирование ПЗУ более ориентировано на проверку целостности данных. Это включает в себя сравнение данных, записанных в ПЗУ, с эталонными данными, чтобы убедиться, что они не искажены или не повреждены.

## 3) Объем данных:

- ОЗУ: Тесты ОЗУ могут потребовать большого объема данных для выполнения проверки на чтение и запись во всех ячейках памяти. Обычно тесты ОЗУ выполняются с использованием специальных тестовых шаблонов.
- ПЗУ: Тесты ПЗУ могут быть более ограниченными в объеме данных, поскольку целью является проверка сохранности данных, а не каждой ячейки в памяти. Тесты ПЗУ также могут быть ориентированы на проверку корректности записи определенных данных.

## 4) Перепрограммирование:

- ОЗУ: ОЗУ является внутренней памятью, которая обновляется при каждом включении и выключении устройства, поэтому в большинстве случаев ее тестирование не включает в себя процедуры перепрограммирования.
- ПЗУ: ПЗУ может быть перепрограммировано, и тестирование может включать в себя проверку процедуры стирания и перезаписи данных в ПЗУ.

## Приложения

### Приложение 1. Полный код программы.

```
1  $MOD51
2
3  org 00h
4  test_ram:
5      mov dptr, #800h          ; Указатель на начало RAM для проверки
6      mov r7, #4              ; Кол-во повтерений внешнего цикла
7      mov r6, #0              ; Кол-во повторений внутреннего цикла
8                              ; (кол-во байт в RAM для проверки)
9                              ; 4*256 = 1024 байт
10     anl p1, #00h            ; Очищаем порт P1 (все лампочки выключены)
11 loop_ram:
12     mov A, #0AAh            ; Записываем в аккумулятор 0xAA
13     movx @dptr, A           ; Записываем в RAM 0xAA
14     movx A, @dptr           ; Читаем из RAM в аккумулятор
15     xrl A, #0AAh            ; Сравниваем с 0xAA
16     jnz error               ; Если не равно, то ошибка
17     inc dptr                ; Увеличиваем указатель RAM на 1
18     setb p1.2               ; Мигаем лампочкой активности
19     clr p1.2                ;
20     djnz r6, loop_ram        ; Повторяем внутренний цикл 256 раз
21     djnz r7, loop_ram        ; Повторяем внешний цикл 4 раза
22     setb p1.0               ; Все хорошо, горит зеленая лампочка
23
24 test_rom:
25     mov dptr, #800h          ; Указатель на начало ROM для проверки
26     mov r7, #0              ; Счетчик суммы
27     mov r6, #64h            ; Кол-во циклов (кол-во байт в ROM для проверки)
28 loop_rom:
29     clr A                   ; A = 0
30     movc A, @A+dptr         ; Читаем из ROM в аккумулятор
31     add A, r7               ; Складываем с суммой
32     mov r7, A               ; Записываем результат в r7
33     inc dptr                ; Увеличиваем указатель ROM на 1
34     setb p1.2               ; Мигаем лампочкой активности
35     clr p1.2                ;
36     djnz r6, loop_rom        ; Повторяем цикл 100 раз
37     cjne r7, #64h, error     ; Если сумма не равна 64h (контрольная сумма),
38                              ; то ошибка
39     setb p1.3               ; Все хорошо, горит зеленая лампочка
40     sjmp $                  ; Бесконечный цикл
41
42 error:
43     setb p1.1               ; Ошибка, горит красная лампочка
44     sjmp $                  ; Бесконечный цикл
45
46 org 800h                    ; ROM
47 db 41, 35, 190, 132, 225, 108, 214, 174, 82, 144
48 db 73, 241, 241, 187, 233, 235, 179, 166, 219, 60
49 db 135, 12, 62, 153, 36, 94, 13, 28, 6, 183
50 db 71, 222, 179, 18, 77, 200, 67, 187, 139, 166
51 db 31, 3, 90, 125, 9, 56, 37, 31, 93, 212
52 db 203, 252, 150, 245, 69, 59, 19, 13, 137, 10
53 db 28, 219, 174, 50, 32, 154, 80, 238, 64, 120
54 db 54, 253, 18, 73, 50, 246, 158, 125, 73, 220
55 db 173, 79, 20, 242, 68, 64, 102, 208, 107, 196
56 db 48, 183, 50, 59, 161, 34, 246, 34, 145, 157
57
58 END
```