Санкт-Петербургский Поли­технический Университет Петра Великого

Инс­титут компьютерных наук и технологий

Кафедра компьютерных систем и программных технологий

Отчет по лабораторной работе

по дисциплине «Микропроцессорные системы»

«Изучение таймеров и системы прерываний»

|  |
| --- |
| \_\_\_\_\_\_\_\_\_\_\_  *подпись*  \_\_\_\_\_\_\_\_\_\_\_  *подпись* |
| \_\_\_\_\_\_\_\_\_\_\_  *подпись* |
| \_\_\_\_\_\_\_\_\_\_\_  *подпись* |

Работу выполнили студенты группы № 43501/3 Бояркин Н.С.

Кан В.С.

Работу приняли преподаватели Кузьмин А.А.

Павловский Е.Г.

Санкт-Петербург

2016

1. **Цель работы**
2. Приобретение практических навыков программирования таймеров.
3. Изучение принципов программного деления частоты.
4. Знакомство с организацией и использование многоуровневой системы прерываний
5. **Программа работы**
6. Ознакомление со структурой и основными режимами работы счетчиков/таймеров МК SAB 80C515, а также с принципом организации системы прерываний этого МК.
7. Разработать структуру информационных связей МК с подключаемыми внешними устройствами (клавиатурой, модулем ЖКИ, осциллографом) в соответствии с номером варианта.
8. Разработать и выполнить программу, генерирующую на заданном выводе порта МК меандр с частотой N\*100 Гц, где N — номер нажатой клавиши.
9. Разработать и выполнить программу формирования ШИМ-сигнала с управляемой скважностью и заданной частотой F=N\*100 Гц, где N — номер рабочего места.
10. Модифицировать программу формирования ШИМ-сигнала, используя для управления скважностью ШИМ-сигнала цифровые коды управляющего воздействия, формируемого инструментальной ЭВМ при работе с вкладкой «Окна управления».
11. Разработать программу «Электронные часы» с отображением на ЖКИ текущего времени с точностью 0,1 с.
12. Разработать программу «Электронный секундомер», определяющую интервал времени между внешними прерываниями int0 и int1, генерируемыми при нажатии двух клавиш разных столбцов клавиатуры стенда.
13. Исследовать работу системы (электронных часов) при наличии нескольких источников прерываний. Для этого произвести модификацию программы «электронные часы», дополнив её обработчиком прерывания от приёмопередатчика последовательного порта.
14. Разработать программу простейшей многозадачной операционной системы с разделением времени.
15. **Теоретические сведения**

**3.1. Блок таймеров/счетчиков МК SAB 80С515**

МК SAB 80C515 содержит в своём составе три программируемых 16-разрядных таймера-счётчика – T/C0, T/C1, T/C2. Эти таймеры реализуются на основе 16-разядных суммирующих счётчиков со схемами управления. Всё три таймера могут работать либо в режиме таймера, либо в режиме счётчика событий. Программно доступными регистрами таймеров T/C0, T/C1 и T/C2 МК являются регистры блока SFR: TH0 и TL0 (таймер 0), TH1 и TL1 (таймер1), TH2 и TL2 (таймер2).

Таймеры T/C0, T/C1 и T/C2 не идентичны. Таймеры T/C0 и T/C1 имеют одинаковую внутреннюю структуру и могут использоваться в качестве программируемого управляемого таймера, генератора программируемой частоты, счётчика внешних событий. Отличительной особенностью таймера T/C1 является то, что он используется для синхронизации работы приёмопередатчика последовательного порта SP (для управления скоростью передачи).

Таймер T/C2 может функционировать как управляемый таймер или использоваться в качестве счётчика внешних событий. Однако основное назначение таймера 2 связано с реализацией на его основе функций быстрого вода (фиксации входных событий) и функций быстрого вывода (формирования требуемых выходных событий).

Обобщённая структура таймеров T/C0 и T/C1 представлена на Рис. 1.

При работе в режиме таймера содержимое T/Cx (x = 0,1,2) увеличивается в каждом машинном цикле, то есть через каждые 12 периодов тактовой частоты fCR. При работе в качестве счётчика событий содержимое T/Cx (x = 0,1,2) инкрементируется при переходе сигнала из 1 в 0 на внешнем выводе Tx (x = 0,1,2) соответствующего таймера. На распознавание перехода требуется два машинных цикла, и максимальная частота счёта входных событий равна fCR/24.

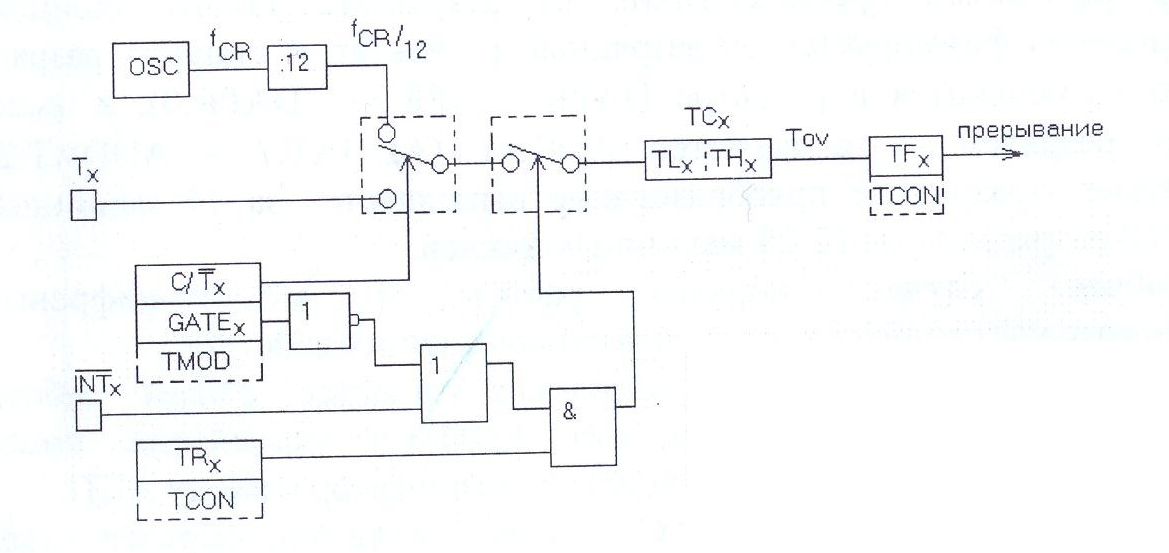


Рис. 1. Обобщённая структура таймеров T/C0 и T/C1

Для задания режимов работы таймера T/Cx (x = 0,1,2) и его управления используются два регистра блока SFR – регистр режимов TMOD (Рис. 2) и регистр управления TCON (Рис. 3).

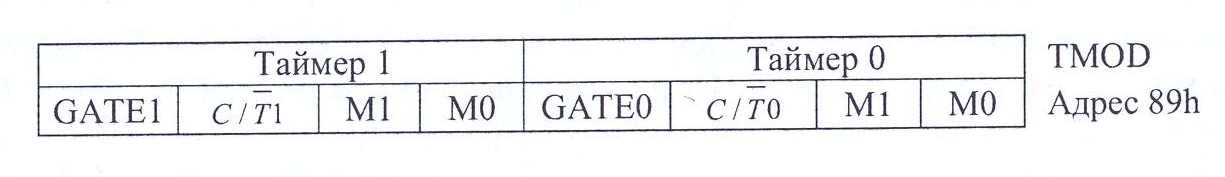


Рис. 2. Регистр режимов TMOD

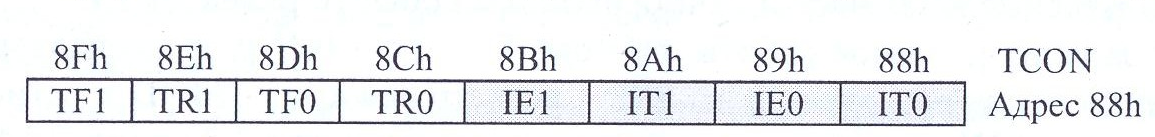


Рис. 3. Регистр управления TCON

GATEx – бит управления блокировкой таймера T/Cx

C/x – бит, определяющий функцию T/Cx: таймер (C/x = 0) или счётчик (C/x = 1)

M1, M0 – биты, задающие режим работы T/Cx (x = 0,1)

Режимы работы таймеров представлены на Рис. 4.

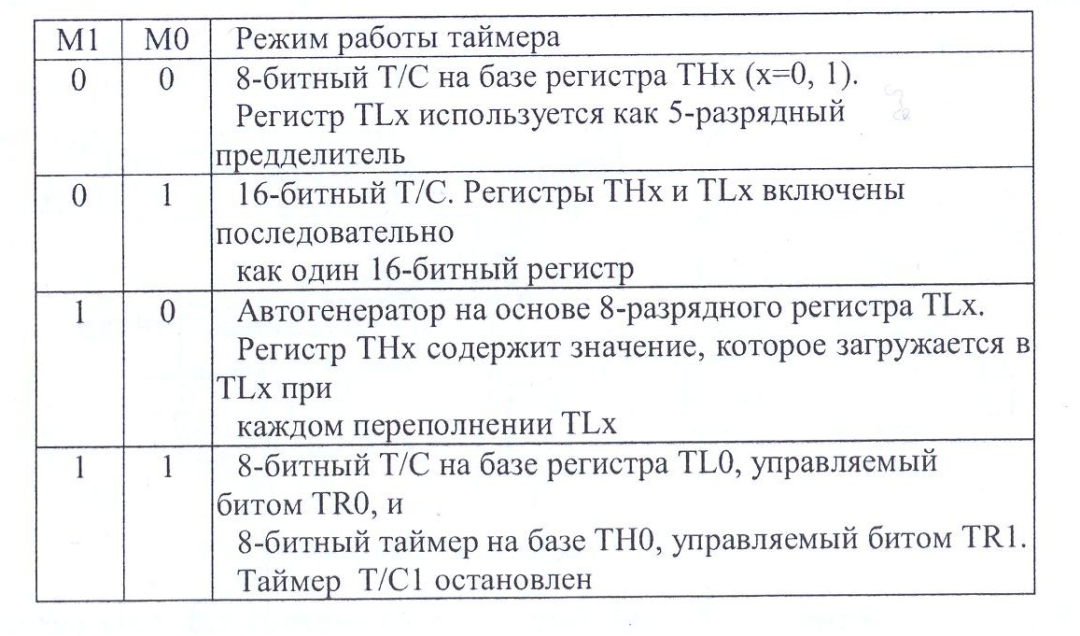


Рис. 4. Режимы работы таймеров T/C1 и T/C2

Регистр TCON для каждого таймера T/Cx (x = 0,1) содержит бит управления режимом работы TRx, с помощью которого при необходимости осуществляют пуск или останов таймера, и флаг переполнения TFx. Четыре младших бита регистра TCON к работе таймера отношения не имеют. Два управляющих бита ITx специфицируют вид сигнала на входе : при ITx = запрос прерывания вызывает уровень (логический 0) сигнала на входе , а при ITx = 1 запрос прерывания вызывает спад (переход из 1 в 0) сигнала на входе .

Важнейшим использованием таймера является формирование «тика» - временного интервала программируемой длительности. «Тики» используются в программах реализации всех временных функций (часы реального времени, определение длительности временных интервалов, формирование сигналов требуемой частоты, определение моментов наступления входных событий, генерация выходных событий в заданные моменты времени и т.п.).

Аппаратный «тик», формируемый таймером, фиксируется при переполнении счётчика таймера. При разрешённых прерываниях в момент формирования импульса переполнения запускается обработчик прерывания, реализующий функции, определяемые пользователем. Обработчик должен быть коротким, поскольку время на выполнение команд обработчика увеличивает длительность «тика».

**3.2. Блок быстрого ввода-вывода МК SAB 80C515**

16-разрядный таймер/счётчик (Рис. 5), наряду с реализацией функций счёта внешних событий и формирования сигналов требуемой частоты, является базовым таймером устройства быстрого ввода/вывода HSIO, предназначенного для регистрации входных и генерации выходных событий в реальном времени.

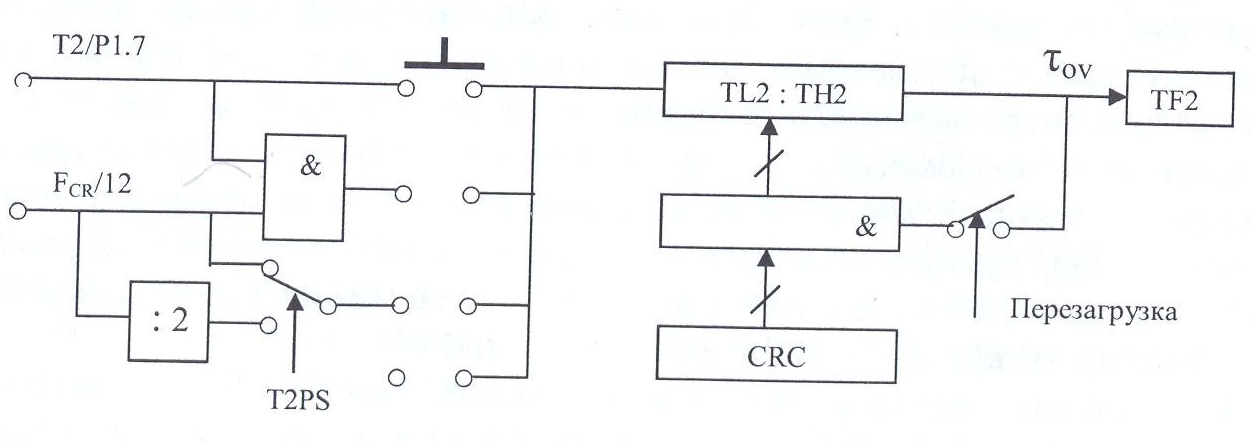


Рис. 5. Обобщённая структура таймера T/C2

Рассмотрим режимы работы таймера T/C2. На его счётный вход могут поступать импульсы либо от внутреннего источника тактовых импульсов (режим таймера), либо сигналы с внешнего входа T2/P1.7 (режим счётчика внешних событий). В режиме 0 таймер остановлен (счётные импульсы на его вход не поступают).

Важной характеристикой таймера T/C2 является возможность его использования в качестве базового таймера устройства быстрого ввода-вывода HSIO. В состав HSIO, кроме таймера T/C2, входит набор из 4-х многофункциональных регистров CC1–CC3, CRC. Названные регистры используются для сравнения с текущим временем (в режиме быстрого вывода и при формировании ШИМ-сигналов), захвата (при фиксации времени наступления события) в режиме быстрого ввода и перезагрузки таймера T/C2 (только регистр CRC).

Формирование ШИМ-сигналов.ШИМ-сигналы – это сигналы с фиксированной частотой и регулируемой скважностью (переменной длительностью сигнала). Формирование ШИМ-сигналов в современных МК чаще всего осуществляется с помощью ШИМ-генераторов. Основными узлами ШИМ-генератора являются таймер, запрограммированный для работы в режиме программируемого делителя частоты, регистр задания скважности, n-разрядный цифровой компаратор и формирователь выходного ШИМ-сигнала. Временная диаграмма работы ШИМ-генератора показана на Рис. 6.



Рис. 6. Временная диаграмма работы ШИМ-генератора

При работе таймера на выходах счётчика таймера TH:TL формируется цифровой пилообразный сигнал. С помощью n-разрядного цифрового компаратора этот сигнал сравнивается со значением, загружаемым в регистр задания скважности, и в момент равенства формируется сигнал, устанавливающий RS-триггер, выступающий в роли формирователя выходного ШИМ-сигнала. Импульс переполнения таймера сбрасывает триггер. Частота поступления импульсов переполнения определяет частоту формируемого ШИМ-сигнала. Длительность импульса и скважность ШИМ-сигнала определяется значением в регистре задания скважности.

Широкое распространение получили ШИМ-сигналы не постоянной, а переменной скважности, которые используются при формировании управляющих сигналов с изменяемой амплитудой. Управляя длительностью генерируемых ШИМ-сигналов, нетрудно сформировать управляющие сигналы требуемой интенсивности или мощности. Такие сигналы обычно используются при работе с объектами с аналоговым управлением. Кодовое (цифровое) управление длительностью ШИМ-сигнала обычно реализуется с помощью задатчиков с цифровым входом, например, с помощью АЦП, на вход которого поступают аналоговые сигналы от датчика объекта управления.

**3.3. Система прерываний МК SAB 80C515**

Наряду с таймерами работу встраиваемых систем управления в реальном времени поддерживает система прерываний МК. При её использовании можно отказаться от непрерывного контроля состояния большого количества датчиков, требующего существенных затрат процессорного времени, и перейти к обработке значительной части информации по прерываниям. Система прерываний является универсальным интерфейсом, обеспечивающим автоматический запуск различных процедур обслуживания по запросу от внешних или внутренних устройств МК-системы. Система прерываний МК SAB80C515 реализована с помощью размещённого на кристалле контроллера прерываний. Последний представляет собой устройство, устанавливающее однозначное соответствие между запросом прерывания и адресом подпрограммы обработки этого запроса.

Система прерываний МК SAB80C515 является многовекторной и многоуровневой. Она объединяет 6 внутренних и 8 внешних источников запросов, которые с помощью логических схем МК преобразуются в 12 векторов прерывания, однозначно задающих адреса программ обслуживания прерывания. Система прерываний устанавливает последовательность обработки одновременно поступающих запросов от нескольких источников и обеспечивает приоритетное обслуживание наиболее ответственных запросов. Очерёдность обслуживания прерывания определяется механизмом 4-уровневой системы приоритетного прерывания.

Все источники прерываний МК SAB80C515 объединены попарно, образуя 6 пар (Рис. 7).

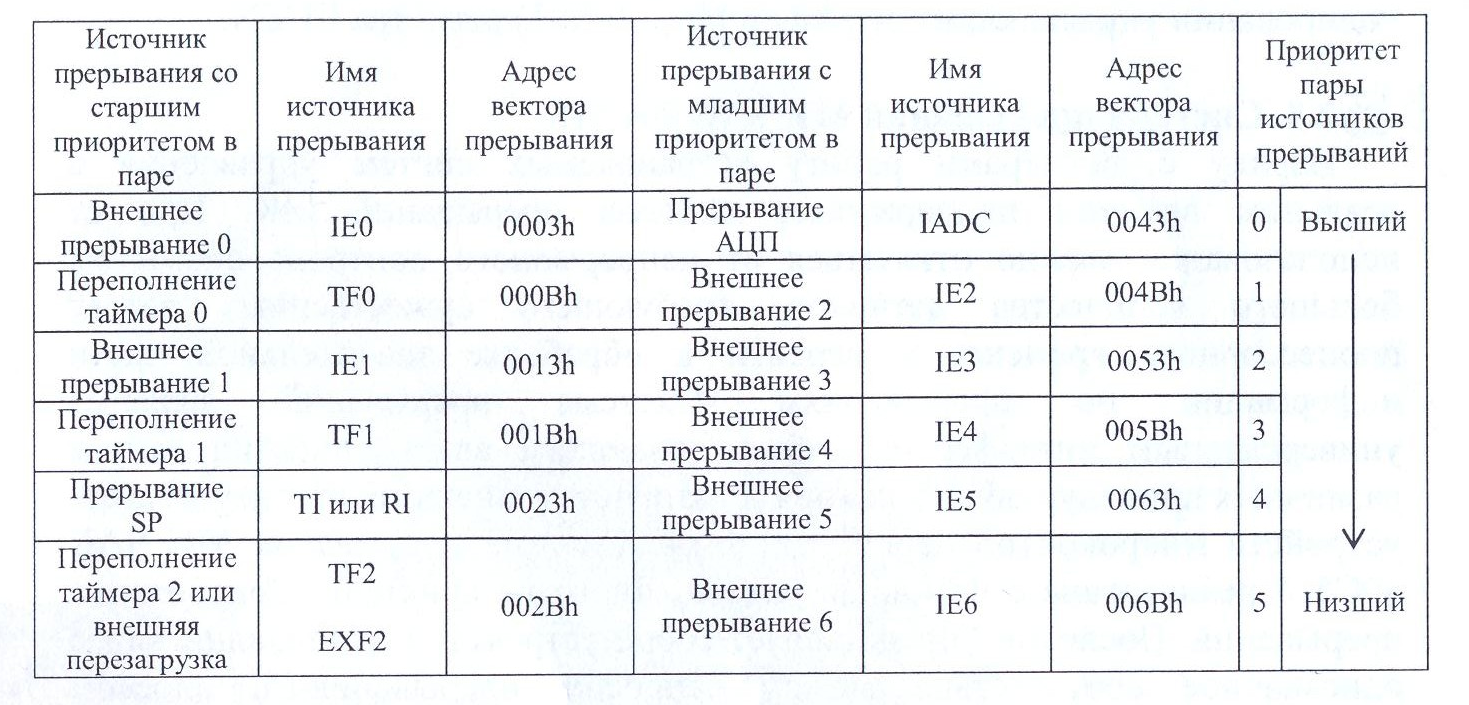


Рис. 7. Пары источников прерываний SAB80C515

Каждой паре источников прерывания программно можно определить один из 4х «глобальных» уровней обслуживания (приоритета). В пределах одного глобального уровня приоритет запроса определяется зарезервированным местом в последовательно опрашиваемой цепочке поступивших запросов. Внутри каждой пары источник прерывания, указанный слева, имеет более высокий приоритет по сравнению со вторым источником пары. Приоритет любого запроса может быть повышен путём перевода этого запроса на другой глобальный уровень с более высоким уровнем приоритетов. Приоритеты пар источников прерываний в пределах одного глобального уровня фиксированы: приоритет первой пары – наивысший, а приоритет шестой пары – самый низкий.

Задание глобального уровня приоритеты каждой пары источников прерывания осуществляется путём установки или сброса пары бит IP0.x и IP1.x (x = 0 – 5) в регистрах управления приоритетами IP0 и IP1.

**3.4. Структура информационных связей МК с подключенными внешними устройствами.**

Вариант №1.

Клавиатура:

* Вход клавиатуры – P1.7 – P1.4.
* Выход клавиатуры – P5.7 – P5.4.

ЖКИ:

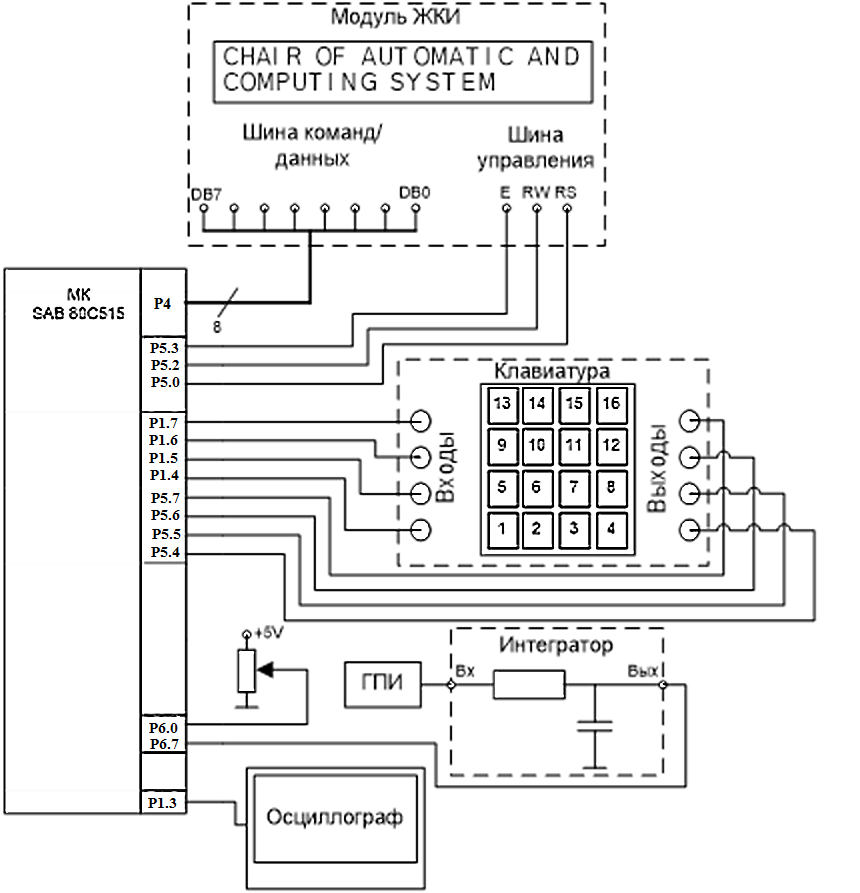
* Шина ЖКИ – P4.
* RS – P5.0.
* R/W – P5.2.
* E – P5.3.

Аналоговые сигналы:

* Потенциометр – P6.0.
* Интегратор – P6.7.

Другие порты:

* Линия формирования меандра – P1.3.



*Рис. 8. Структура информационных связей МК с подключенными внешними устройствами.*

1. **Лабораторные задания**

**Программа 1. Генерация меандра с заданной частотой**

Была разработана программа, генерирующая на заданном выводе порта МК меандр с частотой 100 Гц.

Для разработки программы генерации меандра с частотой, зависящей от номера нажатой клавиши, были определены двухбайтные константы перезагрузки таймера для каждой нажатой клавиши. Эти константы определяют частоту генерируемого меандра.

Расчет констант выполнялся следующим образом. Сперва была составлена таблица (Табл. 1.), в которой каждому возможному номеру нажатой клавиши сопоставлена частота, определяемая выражением №\*100 Гц. Таким образом был получен столбец «Частота». Если клавиша не нажата, то в результате опроса клавиатуры будет сформирован код 0. В этом случае частота будет эквивалентна нажатию клавиши 1, то есть 100 Гц.

*Табл. 1. Расчет констант для перезагрузки счетчика*

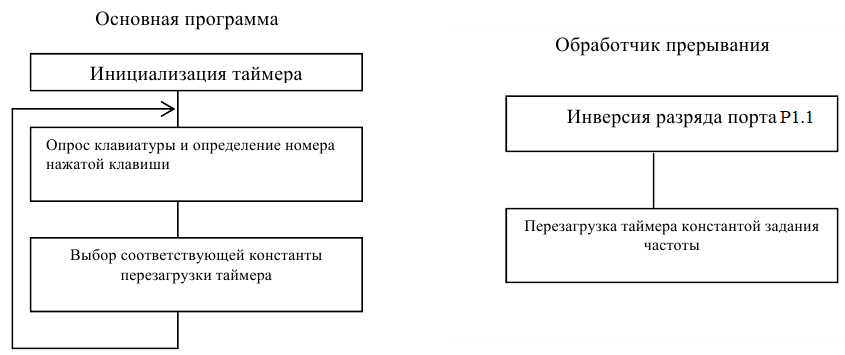
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Номер  нажатой клавиши | Частота, Гц | Константа Nзагр до корректировки | Измеренная частота до корректировки, Гц | Константа Nзагр после корректировки | Измеренная частота после корректировки, Гц |
| 0 | 100 | EC77 | 99 | ECA8 | 99 |
| 1 | 100 | EC77 | 99 | ECA8 | 99 |
| 2 | 200 | F63B | 199 | F647 | 199 |
| 3 | 300 | F97C | 297 | F98C | 300 |
| 4 | 400 | FB1D | 396 | FB29 | 400 |
| 5 | 500 | FC17 | 493 | FC24 | 500 |
| 6 | 600 | FCBD | 591 | FCC9 | 597 |
| 7 | 700 | FD34 | 687 | FD41 | 697 |
| 8 | 800 | FD8E | 784 | FD9A | 803 |
| 9 | 900 | FDD3 | 879 | FDE0 | 896 |
| 10 | 1000 | FE0B | 976 | FE16 | 997 |
| 11 | 1100 | FE38 | 1070 | FE44 | 1097 |
| 12 | 1200 | FE5E | 1165 | FE6A | 1195 |
| 13 | 1300 | FE7E | 1258 | FE8A | 1304 |
| 14 | 1400 | FE99 | 1355 | FEA4 | 1402 |
| 15 | 1500 | FEB1 | 1448 | FEBC | 1493 |
| 16 | 1600 | FEC6 | 1537 | FED2 | 1591 |

Столбец «Период» определяет длительность тика T для каждого из значений частоты в соответствии с выражением , где – период импульсов тактовой частоты, N – расчетное значение длительности. Т.к. в режиме 1 на вход программируемого делителя поступают импульсы с частотой и , то , что упрощает расчеты, ведь T=N.

Столбец «Константа» содержит значения, которыми нужно задать делитель таймера, чтобы тот мог генерировать соответствующий меандр. Поскольку счетчик таймера работает на сложение, для формирования «тика» T требуемой длительности в делитель таймера необходимо загружать константу . Коды указаны в шестнадцатеричной системе.

После загрузки программы и тестирования на стенде было установлено, что теоретические коэффициенты загрузки таймера, рассчитанные по формулам, не позволяют точно установить частоту. Это происходит из-за того, что обработчик прерывания занимает определенное время, тем самым уменьшая частоту генерируемого сигнала. Ввиду этого, необходимо выполнить коррекцию коэффициентов загрузки таймера. Результат коррекции приведен в Табл. 1.

Программа была реализована по следующему алгоритму:



*Рис. 9. Алгоритм программы генерации меандра*

Перед запуском программы была подключена клавиатура и осциллограф, согласно схеме соединений (Рис. 8).

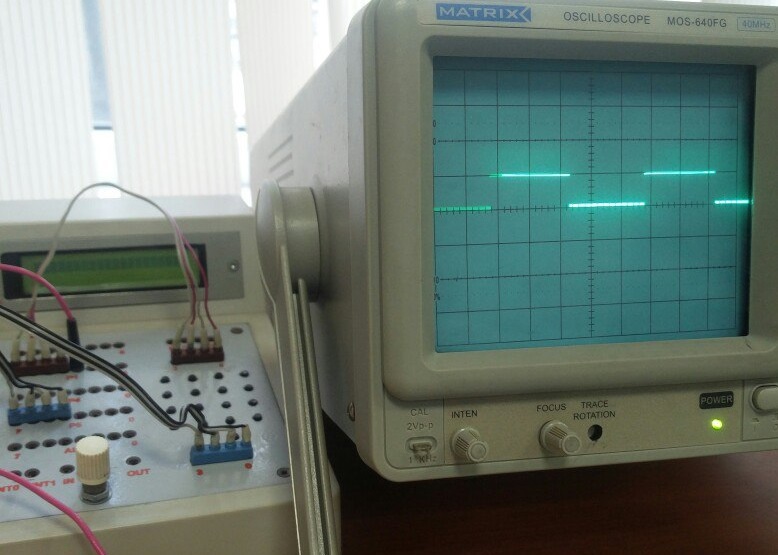
Код основного модуля программы (3.1.asm):

|  |
| --- |
| *; 3.1.asm*  org 8100h  **mov**  r5, #ECh       *; TH*  **mov**  r6, #77h      *; TL*        lcall   init        *; Иниц. таймера и системы прерываний*    **loop**:                   *; рабочий цикл*      lcall memklav  **mov**  dptr, #8300h   *; Коды для TH TL*  **mov**  a, 34h         *; Читаем номер нажатой клавиши*      clr  c              *; Очищаем бит С*      rlc  a              *; Сдвиг влево А (на место A[0] идёт 0 - бит С)*  **push** a              *; Сохраняем А в стеке*      movc  a, @a+dptr    *; В А идёт значение по адресу [А + DPTR]*  **mov**  r5, a          *; Кладём А в ТН*  **pop** a               *; Восстанавливаем А из стека*  **inc**  a              *; Инкрементируем для перехода на TL*      movc  a, @a+dptr    *; В А идёт значение по адресу [А + DPTR]*  **mov**  r6, a          *; Кладём А в ТL*      sjmp  **loop**    init:      anl  TMOD,#11110000b*; Иниц. таймера для работы*      orl  TMOD,#00000001b*; в режиме 16-битного счётчика*  **mov**  TH0, r5        *; Иниц. счётчика Т/С0 для*  **mov**  TL0, r6        *; Формирования тика 5 мс (по-умолчанию)*  **setb**  ea            *; Разрешение всех прерываний*  **setb**  et0           *; Разрешение  прерывания Т/С0*  **setb**  tr0           *; Разрешение счёта*  **ret**  tim0:  **mov**  TH0, r5        *; Иниц. счётчика Т/С0 для*  **mov**  TL0, r6        *; Формирования тика 5 мс (по-умолчанию)*      cpl  P1.3           *; Формирование меандра*      reti        org  800Bh          *; Обработчик прерывания Т/С0*      ljmp  tim0        include ASMS\4081\_3\bk\3\sklav.asm        org  8300h          *; Массив кодов для TH TL*  *; Коды после корректировки*  cod **db** EBh,6Fh,  EBh,6Fh,  F5h,A9h,  F9h,48h,  FAh,ECh,  FBh,F5h,  FCh,91h,  FDh,0Dh,  FDh,73h,  FDh,BCh,  FDh,F6h,  FEh,40h,  FEh,5Dh,  FEh,82h,  FEh,97h,  FEh,B0h,   FEh,C4h  *; Коды до корректировки*  *;cod db ECh,77h,  ECh,77h,  F6h,3Bh,  F9h,7Ch,  FBh,1Dh,  FCh,17h,  FCh,BEh,  FDh,34h,  FDh,8Eh,  FDh,D3h,  FEh,0Bh,  FEh,38h,  FEh,5Eh,  FEh,7Eh,  FEh,9Ah,  FEh,B2h,  FEh,C6h* |

Код подключаемого модуля клавиатуры (sklav.asm):

|  |
| --- |
| *; sklav.asm*  org 8600h  memklav:  **mov** 20h, #0h *; 0 for clear C*  **mov** R1, #33h    *; Адрес первой ячейки памяти для просмотра*  **mov** R3, #3h *; счетчик(по строкам и столбцам)*  **mov** 35h, #0h    *; Счётчик нажатых клавиш*  **mov** 37h, #0h    *; Код символа*  **mov** 38h, #0h    *; номер строки*  **mov** 39h, #0h    *; номер столбца*      lcall klav    *; Сначала - проверка на ноль (ничего не нажато)*  zero\_chk:  **mov** C, 0h       *; clear C*  **mov** A, @R1  *; Читаем данные из памяти*  *;mov 56h, R1*      subb A, #f0h    *; Отнимаем 0Fh - если будет ноль, то ничего не нажато.*  *; Иначе считаем, что было какое-нибудь нажатие.*  **jz** skip\_cntr    *; A==0 - пропускаем счётчик нажатий*  **inc** 35h     *; Не ноль - инкремент счётчика нажатий*  **mov** A,@R1  **mov** 37h,A       *; Сохраняем код нажатой клавиши.*  **mov** 38h,R3  *; Сохранили номер строки нажатой клавиши*    skip\_cntr:  **dec** R1      *; Берём следующий элемент из памяти*  *; Пока не достигли конца массива для проверки -*  **dec** R3      *; увеличиваем номер строки*  **mov** C, 0h       *; clear C*      cjne R1, #2Fh, zero\_chk *; - продолжаем цикл*  *; Вышли из цикла проверки отсутствия нажатий*      **mov** A, 35h  *; Грузим в А счётчик нажатий*  **jz** wr\_0     *; 0 нажатий - пишем ноль*  **mov** C, 0h       *; clear C*      cjne A, #01h, wr\_FF     *; больше 1 нажатия - пишем FF*    **mov** dptr, #cdMask   *; начало массива кодов*  **mov** R3,#0h*; ; обнулили счетчик*    find\_column:  **inc** R3*;     ; счетчик номера столбца*  **mov** 39h,R3  *; сохраняем номер столбца*  **mov** A,R3*;*  **mov** C, 0h       *; clear C*      subb A,#5h  **jz** wr\_FF *; Т.к. клавишу точно нажали(или несколько)*  *; ее код обазятельно должен найтись в массиве*  *; иначе - было нажато несколько клавиш, и код не совпал*      movx A, @dptr   *; записали элемент*  **inc** dptr        *; сразу inc индекс в массиве*  **mov** C, 0h       *; clear C*      cjne A, 37h, find\_column *; если число не равно найденному,*  *; продолжим поиск*  get\_num:  *; номер строки\*4+номер столбца*  **mov** A, 38h  **mov** C, 0h       *; clear C*      rl A      rl A            *; два сдвига числа =\*4*  *;add A, 39h     ; получили число*  **add** A, #5h      subb A, 39h  **mov** 34h, A  *; запись числа*      sjmp ext  wr\_0:   **mov** 34h, #0h      sjmp ext  wr\_FF:  **mov** 34h, #FFh      sjmp ext    *; Существующие коды клавиш - характерны для столбца.*  cdMask: **db** E0h, D0h, B0h, 70h    ext:    **ret**    p5: **equ** f8h    klav:   **mov** r0, #30h    *; задаем адрес карты памяти*      orl p5, #f0h    *; настраиваем порт на ввод*  **mov** a, #7fh *; загружаем код бегущего нуля*    mb: **mov** r2, a        rlc a  **mov** p1.7, c      rlc a  **mov** p1.6, c      rlc a  **mov** p1.5, c      rlc a  **mov** p1.4, c    **mov** a, p5   *; считываем данные с клавиатуры*      anl a,#f0h  **mov** @r0, a  *; и запоминаем их*  **inc** r0      *; увеличиваем адрес для записи*  **mov** a, r2      rr a        *; осуществляем сдвиг*      cjne a, #f7h, mb*; выполняем цикл*  **ret** |

Результат выполнения программы на лабораторном стенде:



*Рис. 10. Сформированный меандр*

Проверим степень влияния времени обработки прерывания. Рассчитаем время, которое затрачивается на одно прерывание (метка tim0 в основном файле 3.1.asm):

|  |
| --- |
| **(** . . . **)**  tim0:  **mov**  TH0, r5        *; Иниц. счётчика Т/С0 для*  **mov**  TL0, r6        *; Формирования тика 5 мс (по-умолчанию)*  ~~cpl  P1.3~~           *; Формирование меандра*  ~~reti~~        org  800Bh          *; Обработчик прерывания Т/С0*      ljmp  tim0  **(** . . . **)** |

*Табл. 2. Система команд микроконтроллера*

|  |  |  |  |
| --- | --- | --- | --- |
| Мнемокод | Б | Ц | Функция |
| MOV direct, Rn | 2 | 2 | (direct) := (Rn) |
| LJMP addr16 | 3 | 2 | (PC):= addr15-0 |

Таким образом вызов одной подпрограммы обработчика прерываний занимает . Однако, обработка запроса прерывания состоит не только из программы прерывания. Также есть задержка фиксации прерывания (С1), анализ флага прерывания (С2), длинный вызов по адресу вектора прерывания (С3, С4). Тогда общее время на прерывание составляет Вызов подпрограммы действительно занимает некоторое время, что сказывается на результатах, поэтому введение коррекции неизбежно.

**Программа 2. Формирование ШИМ-сигнала**

Была разработана программа формирования ШИМ-сигнала с управляемой скважностью и заданной частотой 100 Гц.

Для разработки программы формирования ШИМ-сигнала с управляемой скважностью было необходимо рассчитать значение делителя частоты и определить масштабирующий коэффициент .

Частота ШИМ-сигнала определялась выражением  Гц, где VAR – номер варианта. В моему случае VAR=1, так что  Гц. Настройка таймера T/C2 для работы в режиме автогенератора осуществлялась следующим образом: , где – частота тактовых импульсов (равняется 12 МГц), – расчетное значение делителя. Для суммирующего счетчика таймера T/C2 константа перезагрузки регистра CRC вычисляется по формуле: .

Кодовое (цифровое) управление длительностью ШИМ-сигнала реализовано с помощью АЦП. Код управления скважностью , формируемый АЦП, размещался в ячейке с именем pwm\_var. Поскольку разрядность типовых АЦП обычно равна 8 битам, для корректной работы ШИМ-генератора на базе 16-разрядного счетчика содержимое pwm\_var необходимо масштабировать путем преобразования 8-разрядного значения АЦП (pwm\_var) в 16-разрядный код управления скважностью

здесь i - управляющий код, формируемый на выходе АЦП (i = 0 — 255),

– масштабирующий множитель.

Масштабирующий множитель вычисляется по формуле

где – минимально допустимое значение на выходе АЦП, т.е. при нуле (в ШИМ-генераторе ),

– максимальное значение кода на выходе АЦП, т.е. .

Тогда . Величина является двухбайтным числом, содержащим целую (3h) и дробную (ECh) части. Целая часть является старшим байтом, а дробная часть - младшим байтом 16-разрядного кода управления скважностью.

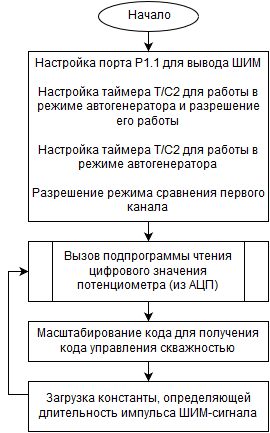
Значение , определяющее длительность формируемого ШИМ-сигнала, является результатом умножения константы управления скважностью pvm\_var, формируемой на выходе АЦП, на масштабирующий коэффициент . При грубом масштабировании исходное значение pvm\_var умножается на целую часть коэффициента (). Однако в этом случае возможна большая погрешность масштабирования. Поэтому в большинстве случаев вычисление выполняют с использованием как , так и . Общий алгоритм вычисления можно представить следующей схемой.

1. Значение pvm\_var умножается на . Результат – 16-битовое число является базовым значением вычисляемого параметра .

2. Значение pvm\_var умножается на . Старший байт 16-разрядного результата является добавкой п(оправкой), которую необходимо учитывать при вычислении .

3. 16-битовое число 1-го произведения суммируется со старшим байтом 2-го произведения. Результат 16-разрядного сложения является значением . Инверсное значение загружается в регистр задания скважности ССх (х=1-3).

Программа была реализована по следующему алгоритму:



*Рис. 11. Алгоритм программы формирования ШИМ-сигнала*

Перед запуском программы был подключен потенциометр и осциллограф, согласно схеме соединений (Рис. 8).

Код основного модуля программы (3.2.asm):

|  |
| --- |
| *; 3.2.asm*  org 8400h  T2CON:  **equ** 0C8h    pwm:      lcall U1\_read *; Закомментирован в следующем пункте 3*      lcall calc      lcall init  *;lcall 128h ; Раскомментирован в следующего пункта 3*      ljmp pwm  **ret**    calc:  **mov** a, 40h  **mov** b, #03h        *; целая часть*  **mul** ab  **mov** r2, b        *; старший байт*  **mov** r3, a        *; младший байт*    **mov** a, 40h  **mov** b, #ECh      *; дробная часть*  **mul** ab  **mov** a, b         *; помещаем в АКК ст. байт*      clr c            *; сброс переноса*  **add** a, r3        *; добавляем к 1му произведению*  **mov** r3, a        *;  ст. байт 2го произведения*  **mov** a, r2      addc a, #0h      *; учет переноса*  **mov** r2, a    **mov** a, r2        *; инвертирование константы*      cpl a  **mov** r2, a  **mov** a, r3      cpl a  **mov** r3, a    **ret**    init:      orl p1, #00001000b *; настройка порта P1.3 для вывода ШИМ*  **mov** T2CON, #0 *; остановка таймера T/C2*    *; Задаем частоту*  **mov** CBh, #FCh *; Старший байт*  **mov** CAh, #17h *; Младший байт*    **mov** C1h, #00001000b *; разрешение работы 1-го канала*  **mov** C3h, r2  **mov** C2h, r3  **mov** T2CON, #00010001b *; задание режима генератора*  **ret**      **include**ASMS\43501\_3\bk\3\p32\myadc.asm *; Закомментирован в следующем пункте 3* |

Код подключаемого модуля цифровой обработки сигнала потенциометра (myadc.asm):

|  |
| --- |
| *; myadc.asm*  org 8100h    ADCON:      **equ** D8h  *ADDA*T:      **equ** D9h  DAPR:    **equ** Dah  U1:      **equ** 40h  U2:      **equ** 41h    U1\_read:    **mov** A,#08h *;выбор нулевого канала*      anl ADCON, #E0h *;инициализация АЦП*      orl ADCON, A  *;mov DAPR, #D4h ;задаем диапазон*  **mov** DAPR, #00h *;задаем диапазон*  **jb** D8h.4,**$**    **mov** r7, #15 *;длина паузы*  m1:     djnz r7, m1 *;пауза*  **mov** U1, ADDAT    U2\_read:    **mov** A,#0fh *; выбор седьмого канала*      anl ADCON, #E0h *; инициализация АЦП*      orl ADCON, A  *;mov DAPR, #D4h ; задаём диапазон*  **mov** DAPR, #62h *;задаем диапазон*  **jb** D8h.4, **$**    **mov** r7, #15 *; длина паузы*  m2:     djnz r7, m2 *; пауза*  **mov** U2, ADDAT    *;lcall 128h*      ljmp U1\_read    **ret** |

Работа ШИМ-генератора контролировалась с помощью осциллографа. Результат выполнения программы представлен в виде зависимости измеренных значений в Табл. 3 и на Рис. 12. Для измерения указанных параметров также использовался осциллограф.

*Табл. 3. Зависимость от*

|  |  |
| --- | --- |
| Напряжение потенциометра, В | Длительность импульса, мс |
| 1,5 | 3,4 |
| 1,9 | 4,4 |
| 2,2 | 5,6 |
| 2,7 | 6,2 |
| 3,2 | 7,2 |
| 3,7 | 8,4 |
| 4,2 | 9,8 |
| 5 | 10,7 |



*Рис. 12. Зависимость от*

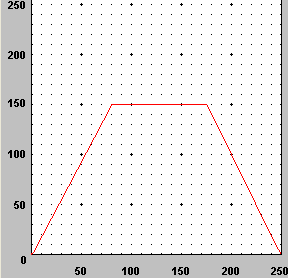
В результате получения функциональной зависимости, было установлено, что импульс действительно изменяется до 10 миллисекунд, что свидетельствует о правильной работе ШИМ-генератора с частотой 100 Гц.

**Программа 3. Модификация программы ШИМ-сигнала**

Была модифицирована программа формирования ШИМ-сигнала, для управления скважностью ШИМ-сигнала были использованы цифровые коды управляющего воздействия, формируемого инструментальной ЭВМ при работе с вкладкой «Окна управления».

Для этого в циклический участок программы была добавлена команда lcall 128h – вызов однократного сеанса связи, а в окне «Окна управления» был установлен адрес ячейки памяти, в которую передается управляющий код. Так как теперь вход ШИМ-генератора подключен не к потенциометру, нет смысла подключать модуль myadc.asm и вызывать lcall U1\_read.

На вход формирователя ШИМ-сигнала был подан трапециевидный сигнал:



*Рис. 13. Трапециевидный сигнал на входе формирователя ШИМ-сигнала*

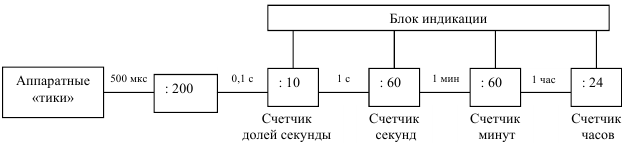
После подачи сигнала из «Окон управления» мы наблюдали плавное изменение ШИМ-сигнала на осциллографе.

**Программа 4. Электронные часы**

Была разработана программа «Электронные часы» с отображением на ЖКИ текущего времени с точностью 0,1 с.

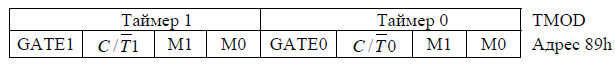
В качестве счетных импульсов временных «тиков» были использованы прерывания таймера, следующие с частотой 2 кГц (каждые 500 мкс). Программный счетчик «тиков» был реализован в фоновой циклической программе.

Так как прерывания возникают каждые 500 мкс, то для того, чтобы получить 0.1 сек необходимо, чтобы прошло 200 тиков, для того, чтобы получить 1 сек – 2000 тиков, и так далее. Функциональная схема алгоритма работы программы «Электронные часы»:



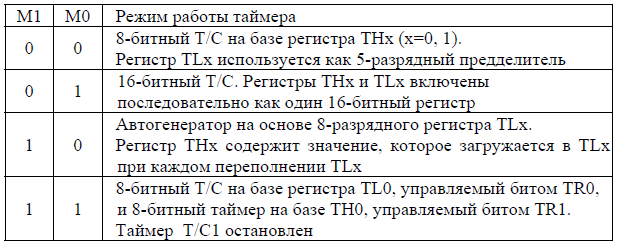
*Рис. 14. Функциональная схема электронных часов*

Настройка таймера была произведена следующим образом: в первую очередь был настроен регистр TMOD, который содержит набор программно управляемых бит (по четыре бита для каждого таймера), используемых для задания режима T/C (Рис. 15).



*Рис. 15. Регистр TMOD*

GATE0 – бит управления блокировкой таймера; C/T0 – бит, определяющий функцию (таймер или счетчик); M1, M0 – биты, задающие режим работы.



*Рис. 16. Назначение битов режима работы TMOD*

Настройка таймера, в программе:

|  |
| --- |
| anl TMOD,#11110000b  orl TMOD,#00000001b |

При использовании маски #11110000b, биты таймера 1 остались нетронутыми, а биты таймера 0 обнулились. После этого установили в единицу только нулевой бит регистра TMOD, что соответствует настройке: GATE0 = 0 (таймер не блокирован), C/T0 = 0 (таймер), M1 = 0, M0 = 1 (шестнадцатибитный таймер).

Также дополнительно, по заданию преподавателя, была добавлена временная остановка часов при нажатии на кнопку клавиатуры.

Перед запуском программ были подключены ЖКИ и клавиатура, согласно схеме соединений (Рис. 8).

Код основного модуля программы (3.4.asm):

|  |
| --- |
| *; 3.4.asm*  org 8400h    timertics:  **equ** 70h  dsec:   **equ** 71h  sec:    **equ** 72h  min:    **equ** 73h  hour:   **equ** 74h    strdata:    **equ** 8700h        lcall timerinit      lcall init\_clk  infloop:      lcall memklav    **mov**  a, 34h      cjne a, #01h, cont *; Остановка таймера при нажатии на кнопку 1*      ljmp infloop    cont:   lcall upd\_time      lcall set\_clk      lcall lcd      ljmp infloop  **ret**    init\_clk:  **mov** dsec, #00h *; Обнуляем часы*  **mov** sec, #00h  **mov** min, #00h  **mov** hour, #00h  **ret**    upd\_time:  **mov** a, timertics      clr c      subb a, #200    *; 500 мкс \* 200 = 0.1 сек*  **jc** stop\_upd  **mov** timertics, a    *; tics -= 200*  **inc** dsec    **mov** a, dsec      cjne a, #10, stop\_upd       *; 1 секунда*  **mov** dsec, #00h  **inc** sec    **mov** a, sec      cjne a, #60, stop\_upd       *; 1 минута*  **mov** sec, #00h  **inc** min    **mov** a, min      cjne a, #60, stop\_upd       *; 1 час*  **mov** min, #00h  **inc** hour    **mov** a, hour      cjne a, #24, stop\_upd       *; 1 день*  **mov** hour, #00h  stop\_upd:   **ret**    set\_clk:  **mov** dptr, #strdata  *;*  **mov** dpl, #20    *; индикация 1-2 символов (часы)*  **mov** r5, hour      lcall hex2dec  **mov** a, r6      lcall put\_dig   *; Старший разряд*  **inc** dpl  **mov** a, r7      lcall put\_dig   *; Младший разряд*    **inc** dpl  **inc** dpl     *; индикация 4-5 символов (минуты)*  **mov** r5, min      lcall hex2dec  **mov** a, r6      lcall put\_dig   *; Старший разряд*  **inc** dpl  **mov** a, r7      lcall put\_dig   *; Младший разряд*    **inc** dpl  **inc** dpl     *; индикация 7-8 символов (секунды)*  **mov** r5, sec      lcall hex2dec  **mov** a, r6      lcall put\_dig   *; Старший разряд*  **inc** dpl  **mov** a, r7      lcall put\_dig   *; Младший разряд*    **inc** dpl  **inc** dpl     *; индикация 10 символа (децисекунды)*  **mov** r5, dsec      lcall hex2dec  **mov** a, r7      lcall put\_dig   *; Младший разряд*  **ret**    hex2dec:  **mov** a, r5  **mov** b, #10  **div** ab  **mov** r6, a  **mov** r7, b    **ret**    put\_dig:  **add** a, #30h *; Перевод в ASCII*      movx @dptr, a  **mov** b, a  **mov** a, dpl  **add** a, #40h  **mov** r1, a  **mov** @r1, b  **ret**        include C:\SHELL51\ASMS\43501\_3\bk\3\p34\timer.asm      include C:\SHELL51\ASMS\43501\_3\bk\3\p34\disp.asm      include C:\SHELL51\ASMS\43501\_3\bk\3\sklav.asm        org strdata  line1:  **db** '\_\_\_\_\_\_\_TIMER:\_\_\_\_\_\_\_'  line2:  **db** 'hh:mm:ss:d\_\_\_\_\_\_\_\_\_\_' *;* |

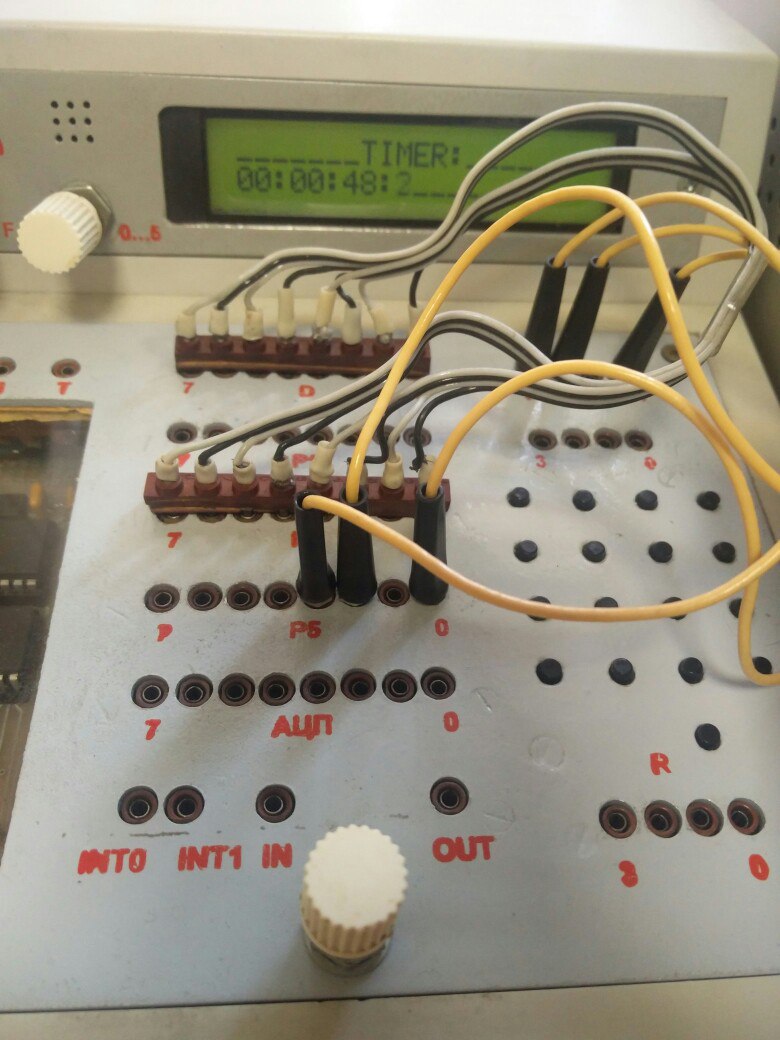
Код подключаемого модуля индикации ЖКИ (disp.asm):

|  |
| --- |
| *; disp.asm*  lcd:      P4: **equ** E8h      P5: **equ** F8h  indic:      clr P5.0  **mov** r4,#38h *; Установка 8-битного режима обмена с ЖКИ с выводом обеих строк*      lcall ind\_wr  **mov** r4,#0Ch *; Отображение экрана без курсоров*      lcall ind\_wr  **mov** r4,#80h *; Загрузка в счетчик АС адреса нулевой ячейки 1-й строки памяти ЖКИ*      lcall ind\_wr    *; Подготовка к вводу данных (R/S=1)*    str1\_set:   **mov** dptr,#strdata   *;Адрес внешней памяти, где хранится выводимые строки*  **setb** P5.0    wr\_str1:    movx a,@dptr  **mov** r4,a      lcall ind\_wr    *; Запись данных в ЖКИ*  **inc** dptr  **mov** a,dpl      cjne a,#14h,wr\_str1 *; проверка окончания 1-й строки*    str2\_set:      clr P5.0  **mov** r4,#C0h *; Команда для адресации 1-й ячейки второй строки*      lcall ind\_wr  **setb** P5.0    wr\_str2:    movx a,@dptr  **mov** r4,a      lcall ind\_wr  **inc** dptr  **mov** a,dpl      cjne a,#026h,wr\_str2  **ret**    ind\_wr:     **mov** P4, r4  *; Загрузка в порт P4 записываемой в ЖКИ информации*  **setb** P5.3   *; Установка сигнала E*      clr P5.2        *; R/W=0 (запись)*      lcall delay      clr P5.3   *; Сброс сигнала E*      lcall delay  **setb** P5.3   *; Установка сигнала E*  **ret**    delay:  **mov** r3, #7  m1: djnz r3, m1  **ret** |

Код подключаемого модуля таймера (timer.asm):

|  |
| --- |
| *; timer.asm*  timerinit:      anl TMOD,#11110000b *; Инициализация таймера*      orl TMOD,#00000001b    **mov** TH0, #FEh   *; Инициализация счетчика 500 мкс*  **mov** TL0, #0Bh    **setb** ea     *; разрешение прерываний*  **setb** et0    *; разрешение внутр. прер. T/C0*  **setb** tr0  **ret**       timer\_ir:  **mov** TH0, #FEh   *; Инициализация счетчика 500 мкс*  **mov** TL0, #0Bh  **inc** timertics   *; Инкрементируем счетчик прерываний*      reti        org 800bh      ljmp timer\_ir |

Код подключаемого модуля клавиатуры (sklav.asm) абсолютно идентичен подключаемому модулю клавиатуры из Программы 1 (генерация меандра с заданной частотой).



*Рис. 17. Электронные часы*

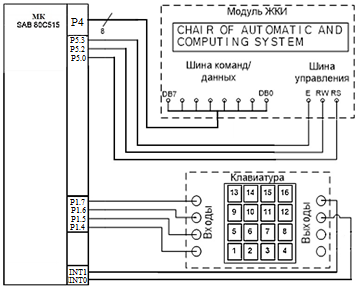
С помощью секундомера на телефоне была проверена точность электронных часов. Были засечены пять минут времени, во время которых одновременно работала программа и секундомер на телефоне. Секундомер выдал результат 05:00:134, а программа 04:59:7. Задержка в 0.434 объясняется скорее скоростью реакции человека, останавливающего секундомер и программу, чем неправильными результатами программы.

**Программа 5. Электронный секундомер**

Была разработана программа «Электронный секундомер», определяющая интервал времени между внешними прерываниями int0 и int1, генерируемыми при нажатии двух клавиш разных столбцов клавиатуры стенда.

Для определения интервала времени между двумя внешними прерываниями int0 и int1, генерируемых при нажатии двух клавиш разных столбцов блока клавиатуры, необходимо было модифицировать предыдущую программу и дополнить ее двумя обработчиками прерываний.

Схема подключения для программы «Электронного секундомера»:



*Рис. 18. Схема подключения для «Электронного секундомера»*

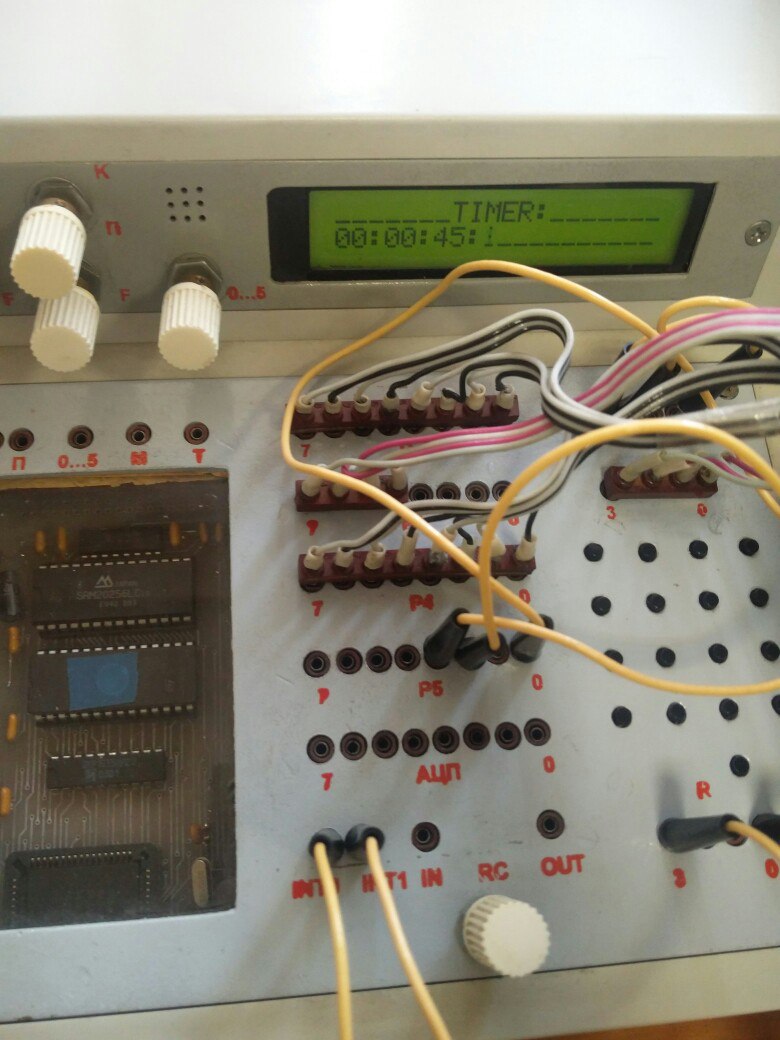
Предпоследний порт был подключен к выводу int0 и служил сигналом для запуска секундомера по нажатию на любую из клавиш третьего столбца клавиатуры. Последний порт был подключен к выводу int1 – для окончания счета при нажатии клавиши из четвертого столбца стенда.

При поступлении запроса int0 его обработчик осуществляет запуск таймера, обнуляет счетчик времени, запрещает собственные прерывания int0 и разрешает прерывания int1. Обработчик прерывания int1 должен остановить таймер, блокировать собственные прерывания int1 и разрешить прерывания int0. Значение интервала времени между внешними прерываниями int0 и int1 отображаются на экране ЖКИ.

Код основного модуля программы (3.5.asm):

|  |
| --- |
| *; 3.5.asm*  org 8400h        timertics:  **equ** 70h      dsec:   **equ** 71h      sec:    **equ** 72h      min:    **equ** 73h      hour:   **equ** 74h      strdata:    **equ** 8700h        lcall timerinit  **mov** p1,#0h      clr ex1  **setb** ex0  reset:  lcall timerinit      lcall init\_clk  infloop:  **jb** 10h, reset      lcall upd\_time      lcall set\_clk      lcall lcd      ljmp infloop  **ret**    init\_clk:  **mov** dsec, #00h  **mov** sec, #00h  **mov** min, #00h  **mov** hour, #00h      clr 10h  **ret**    upd\_time:  **mov** a, timertics      clr c      subb a, #200    *; 500 мкс \* 200 = 0.1 сек*  **jc** stop\_upd  **mov** timertics, a    *; tics -= 200*  **inc** dsec    **mov** a, dsec      cjne a, #10, stop\_upd       *; 1 секунда*  **mov** dsec, #00h  **inc** sec    **mov** a, sec      cjne a, #60, stop\_upd       *; 1 минута*  **mov** sec, #00h  **inc** min    **mov** a, min      cjne a, #60, stop\_upd       *; 1 час*  **mov** min, #00h  **inc** hour    **mov** a, hour      cjne a, #24, stop\_upd       *; 1 день*  **mov** hour, #00h  stop\_upd:   **ret**    set\_clk:  **mov** dptr, #strdata  *;*  **mov** dpl, #20    *; индикация 1-2 символов (часы)*  **mov** r5, hour      lcall hex2dec  **mov** a, r6      lcall put\_dig   *; Старший разряд*  **inc** dpl  **mov** a, r7      lcall put\_dig   *; Младший разряд*    **inc** dpl  **inc** dpl     *; индикация 4-5 символов (минуты)*  **mov** r5, min      lcall hex2dec  **mov** a, r6      lcall put\_dig   *; Старший разряд*  **inc** dpl  **mov** a, r7      lcall put\_dig   *; Младший разряд*    **inc** dpl  **inc** dpl     *; индикация 7-8 символов (секунды)*  **mov** r5, sec      lcall hex2dec  **mov** a, r6      lcall put\_dig   *; Старший разряд*  **inc** dpl  **mov** a, r7      lcall put\_dig   *; Младший разряд*    **inc** dpl  **inc** dpl     *; индикация 10 символа (децисекунды)*  **mov** r5, dsec      lcall hex2dec  **mov** a, r7      lcall put\_dig   *; Младший разряд*  **ret**    hex2dec:  **mov** a, r5  **mov** b, #10  **div** ab  **mov** r6, a  **mov** r7, b  **ret**    put\_dig:  **add** a, #30h *; Перевод в ASCII*      movx @dptr, a  **mov** b, a  **mov** a, dpl  **add** a, #40h  **mov** r1, a  **mov** @r1, b  **ret**        include C:\SHELL51\ASMS\43501\_3\bk\3\p35\disp.asm    timerinit:      anl TMOD,#11110000b      orl TMOD,#00000001b *; Инициализация таймера*    **mov** TH0, #FEh  **mov** TL0, #0Bh    *; Инициализация счетчика 500 мкс*    **setb** ea  **setb** et0  **setb** tr0  **ret**     timer\_ir:  **mov** TH0, #FEh   *; Инициализация счетчика 500 мкс*  **mov** TL0, #0B  **inc** timertics   *; Инкрементируем счетчик прерываний*      reti    int0:      clr ex0  **setb** ex1      lcall timerinit      reti    int1:      clr ex1  **setb** ex0  **setb** 10h      reti        org 8003h   *;внешнее прерывание 0*      ljmp int0        org 800bh   *; прерывание таймера*      ljmp timer\_ir        org 8013h   *;внешнее прерывание 1*      ljmp int1        org strdata  line1:  **db** '\_\_\_\_\_\_\_TIMER:\_\_\_\_\_\_\_'  line2:  **db** 'hh:mm:ss:d\_\_\_\_\_\_\_\_\_\_\_\_' *;* |

Код подключаемого модуля индикации ЖКИ (disp.asm) абсолютно идентичен подключаемому модулю клавиатуры из Программы 4 (электронные часы).



*Рис. 19. Электронный секундомер*

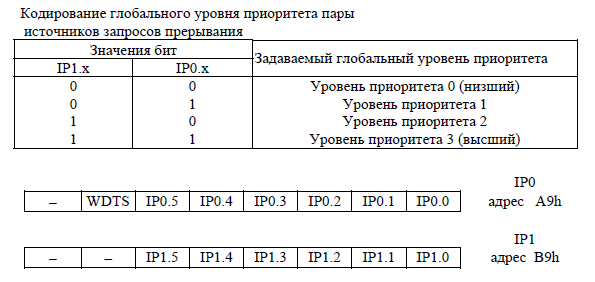
В результате, при нажатии на кнопку из первого столбца запускался счет времени, что можно было наблюдать на экране стенда. Повторное нажатие не приводило к перезапуску таймера, т.к. после выполнения обработчика прерывания для int0 это прерывание блокировалось. Остановка секундомера с помощью клавиш из четвертого ряда приводила к остановке счета. После остановки можно было вновь запустить секундомер.

**Программа 6. Модификация электронных часов**

Была исследована работа электронных часов при наличии нескольких источников прерываний. Для этого была произведена модификация программы «электронные часы», с дополнением в неё обработчика прерывания от приёмопередатчика последовательного порта.

Модификация включает в себя добавление установления приоритетов переполнения таймера 0 и прерывания от SP и добавления в цикл процедуры lcall 128h внутри обработчика прерывания последовательного порта.

С помощью установки пары бит регистров управления приоритетами IP0.x и IP1.x (x=1,4) (Рис. 20.) производилась настройка «глобального» уровня обслуживания для источников прерывания: переполнение таймера 0 (TF0) и прерывания SP (RI или TI).



*Рис. 20. Форматы регистров IP0 и IP1*

Перед запуском программы был подключен ЖКИ, согласно схеме соединений (Рис. 8).

Код основного модуля программы (3.6.asm):

|  |
| --- |
| *; 3.6.asm*  org 8400h    timertics:  **equ** 70h  dsec:   **equ** 71h  sec:    **equ** 72h  min:    **equ** 73h  hour:   **equ** 74h        strdata:    **equ** 8700h        lcall timerinit      lcall init\_clk  infloop:      lcall upd\_time      lcall set\_clk      lcall lcd      ljmp infloop  **ret**  init\_clk:  **mov** dsec, #00h  **mov** sec, #00h  **mov** min, #00h  **mov** hour, #00h      clr 10h  **ret**    upd\_time:  **mov** a, timertics      clr c      subb a, #200    *; 500 мкс \* 200 = 0.1 сек*  **jc** stop\_upd  **mov** timertics, a    *; tics -= 200*  **inc** dsec    **mov** a, dsec      cjne a, #10, stop\_upd       *; 1 секунда*  **mov** dsec, #00h  **inc** sec    **mov** a, sec      cjne a, #60, stop\_upd       *; 1 минута*  **mov** sec, #00h  **inc** min    **mov** a, min      cjne a, #60, stop\_upd       *; 1 час*  **mov** min, #00h  **inc** hour    **mov** a, hour      cjne a, #24, stop\_upd       *; 1 день*  **mov** hour, #00h  stop\_upd:   **ret**    set\_clk:  **mov** dptr, #strdata  *;*  **mov** dpl, #20    *; индикация 1-2 символов (часы)*  **mov** r5, hour      lcall hex2dec  **mov** a, r6      lcall put\_dig   *; Старший разряд*  **inc** dpl  **mov** a, r7      lcall put\_dig   *; Младший разряд*    **inc** dpl  **inc** dpl     *; индикация 4-5 символов (минуты)*  **mov** r5, min      lcall hex2dec  **mov** a, r6      lcall put\_dig   *; Старший разряд*  **inc** dpl  **mov** a, r7      lcall put\_dig   *; Младший разряд*    **inc** dpl  **inc** dpl     *; индикация 7-8 символов (секунды)*  **mov** r5, sec      lcall hex2dec  **mov** a, r6      lcall put\_dig   *; Старший разряд*  **inc** dpl  **mov** a, r7      lcall put\_dig   *; Младший разряд*    **inc** dpl  **inc** dpl     *; индикация 10 символа (децисекунды)*  **mov** r5, dsec      lcall hex2dec  **mov** a, r7      lcall put\_dig   *; Младший разряд*  **ret**    hex2dec:  **mov** a, r5  **mov** b, #10  **div** ab  **mov** r6, a  **mov** r7, b  **ret**    put\_dig:  **add** a, #30h *; Перевод в ASCII*      movx @dptr, a  **mov** b, a  **mov** a, dpl  **add** a, #40h  **mov** r1, a  **mov** @r1, b  **ret**    usart:  **jnb** ri, skip  **push**    a  **push**    0  **push**    psw      lcall   128h       *; Вызов однократного сеанса связи*      clr     ri  **pop** psw  **pop** 0  **pop** a  skip:   reti        include C:\SHELL51\ASMS\43501\_3\bk\3\p36\disp.asm      include C:\SHELL51\ASMS\43501\_3\bk\3\p36\timer.asm        org 8023h      ljmp usart     1. org strdata 2. line1:  **db** '\_\_\_\_\_\_\_TIMER:\_\_\_\_\_\_\_' 3. line2:  **db** 'hh:mm:ss:d\_\_\_\_\_\_\_\_\_\_' *;* |

Код подключаемого модуля таймера (timer.asm):

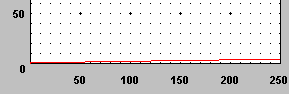
|  |
| --- |
| *; timer.asm*  timerinit:      anl TMOD,#11110000b *; Инициализация таймера*      orl TMOD,#00000001b    **mov** TH0, #FEh   *; Инициализация счетчика 500 мкс*  **mov** TL0, #0Bh    **setb** ea     *; разрешение прерываний*  **setb** et0    *; разрешение внутр. прер. T/C0*  **setb** tr0    **mov** A9h, #00010000b  *; Установка приоритетов для таймера (регистры IP0 и IP1)*  **mov** B9h, #00010000b  **setb** **es**  **ret**       timer\_ir:  **mov** TH0, #FEh   *; Инициализация счетчика 500 мкс*  **mov** TL0, #0Bh  **inc** timertics   *; Инкрементируем счетчик прерываний*      reti        org 800bh      ljmp timer\_ir |

В «Окнах управления» было проведено три эксперимента:

1. Уровень приоритета прерываний от таймера ниже уровня приоритета прерываний от последовательного порта.

|  |
| --- |
| **mov** A9h, #00010000b *; IP0*  **mov** B9h, #00010000b *; IP1* |

В этом случае прерывания от таймера не обрабатываются в момент выполнения обработчика от последовательного порта, сами прерывания от таймера могут прерваться довольно длительной процедурой обработчика прерывания от последовательного порта. Это приводит к существенному растягиванию «тика», счет времени отстает заметно.

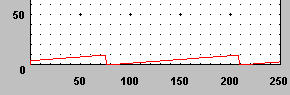


*Рис. 21. Длина тика при низком приоритете таймера*

1. Уровень приоритета прерываний от таймера равен уровню приоритета прерываний от последовательного порта.

|  |
| --- |
| **mov** A9h, #00010010b *; IP0*  **mov** B9h, #00010010b *; IP1* |

При равных приоритетах прерывания от таймера не обрабатываются в момент выполнения обработчика прерываний от последовательного порта. Поэтому обработка части прерываний от таймера откладывается до завершения обработчика прерываний от последовательного порта, что приводит к растягиванию «тика». Из-за этого скорость счета уменьшается, наблюдаем отставание времени.

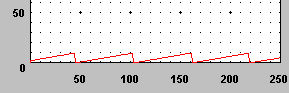


*Рис. 22. Длина тика при равных приоритетах таймера и последовательного порта*

1. Уровень приоритета прерываний от таймера выше уровня приоритета прерываний от последовательного порта

|  |
| --- |
| **mov** A9h, #00000010b *; IP0*  **mov** B9h, #00000010b *; IP1* |

В этом случае прерывания от таймера вызываются сразу после возникновения, не зависимо от того, что именно – фоновая программа или обработчик прерывания от последовательного порта — обрабатываются в момент возникновения прерывания. Отставания счета времени не наблюдалось.



*Рис. 23. Длина тика при высоком приоритете таймера*

**Программа 7. Многозадачная операционная система**

Была разработана программа простейшей многозадачной операционной системы с разделением времени.

Была реализована многозадачность, в которой был использован принцип разделения времени: каждой задаче по очереди предоставляется свой временной промежуток, называемый квантом. Квант задается таймером, длина кванта – константой перезагрузки. Обработчику прерываний необходимо кроме перезагрузки константы таймера осуществить операцию переключения контекста, то есть сохранение набора всех регистров, используемых программой, запись их в память, вычисление адреса следующей задачи, восстановление ее контекста из памяти.

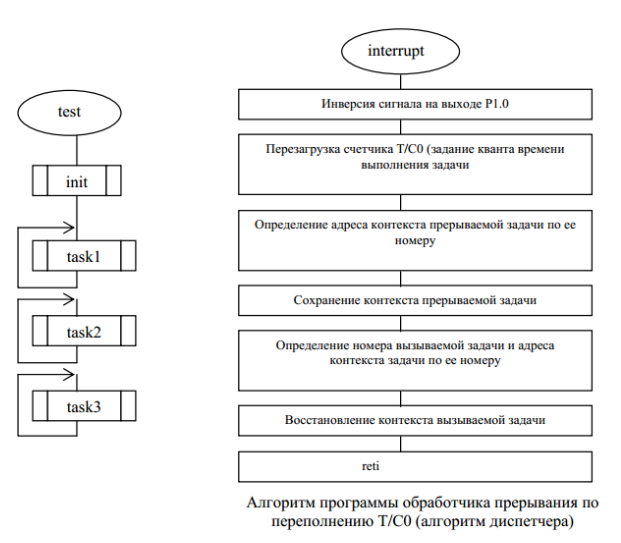


*Рис. 24. Контекст задачи*

Адрес дескриптора задачи вычисляется при помощи номера прерываемой задачи. Дескрипторы хранятся во внешней памяти, для них в программе зарезервированы области данных. Номер задачи вычисляется по следующему алгоритму. Из номера текущей задачи вычитается 2. Если при этом результат отрицательный, то значение переменной, отвечающей за номер задачи, увеличивается на единицу. Иначе ей присваивается ноль. Адреса дескрипторов отличаются только младшим байтом, чтобы проще было вычислять номер задачи.

Перед началом работы программы вызывается подпрограмма, инициализирующая таймер, устанавливающая номер вызываемой задачи в 0, инициализируются дескрипторы задач.

Каждая задача зациклена на выполнение только самой себя. При смене контекста происходит так же смена регистра PC, то есть счетчика команд. Все программы размещаются в разных областях внешней памяти, поэтому никак между собой не взаимодействуют, за исключением чтения и записи в область внутренней памяти.



*Рис. 25. Схема алгоритма обработчика прерывания*

Перед запуском программ были подключены ЖКИ и клавиатура, согласно схеме соединений (Рис. 8).

Код основного модуля программы (3.7.asm):

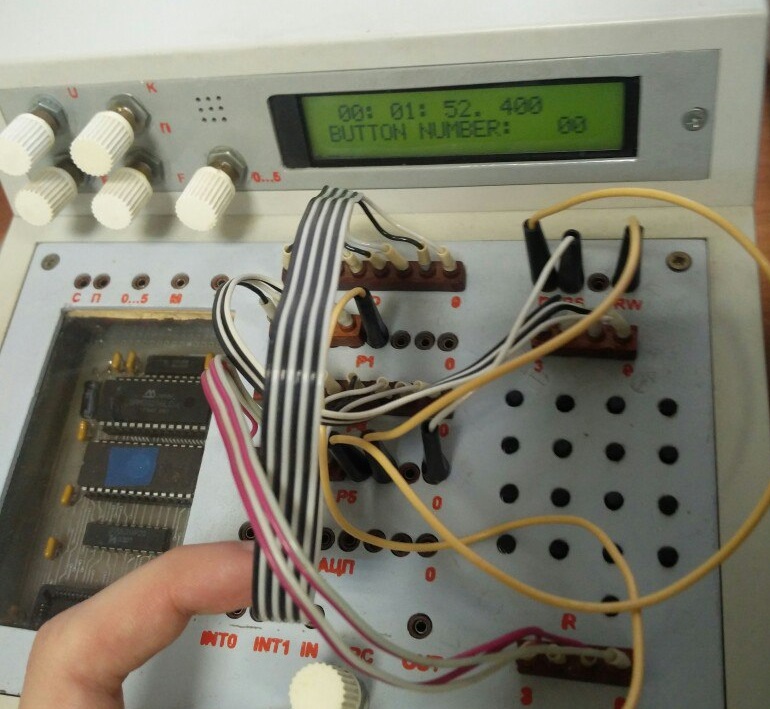
|  |
| --- |
| *; 3.7.asm*  org 8100h    num\_task: **equ** 50h    **mov** **SP**, #7h      lcall init    *; Задача1 - определение номера нажатой клавиши*  *; и преобразование в ASCII код*  prog1:      cpl P1.1      lcall memklav      lcall decim      sjmp prog1    *; Задача 2 - индикация номера нажатой клавиши*  prog2:      cpl P1.2      lcall indic      sjmp prog2    *; Задача 3 - электронные часы*  prog3:      cpl P1.3      lcall clock      sjmp prog3    *; дескриптор задачи1*  prog1\_d:    **db** 11h, 1,  0, 0, 0, 0, 0, 0, 00, 00, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0    *; дескриптор задачи2*  prog2\_d:    **db** 11h, 1,  0, 0, 0, 0, 0, 0, 00, 00, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0    *; дескриптор задачи3*  prog3\_d:    **db** 11h, 1,  0, 0, 0, 0, 0, 0, 00, 00, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0    *; инициализация дескрипторов и таймера*  init:  **mov** num\_task, #0  *; обнуление декод. значения клавиши*  **mov** 34h, #0    **mov** dptr, #prog2  **mov** r0, DPL  **mov** r1, DPH  **mov** dptr, #prog2\_d      lcall init\_rout    **mov** dptr, #prog3  **mov** r0, DPL  **mov** r1, DPH  **mov** dptr, #prog3\_d      lcall init\_rout    *;anl TMOD, #00000000b*  *;orl TMOD, #00000001b*  **mov** TMOD, #11110001b  **mov** TH0, #F6h  **mov** TL0, #3Bh    *;сброс регистров значений таймера*  **mov** ms, #0h  **mov** sd10, #0h  **mov** sec, #0h  **mov** min, #0h  **mov** hours, #0h    **setb** ea *;разрешение всех прерываний*  **setb** et0    *;разрешение прерываний от таймера 0*  **setb** tr0    *;разрешение счета таймера 0*  **ret**    init\_rout:  **mov** a, DPL  **add** a, #8  **mov** DPL, a  **mov** a, r0      movx @dptr, a  **inc** dptr  **mov** a, r1      movx @dptr, a  **ret**    tim0:  **mov** TH0, #F6h   *; 2,5 ms F6 47*  **mov** TL0, #47h      cpl p1.0    *; программа-диспетчер*  dispatcher:    *; сохранение SFR*  **push** dph  **push** dpl  **push** psw  **push** b  **push** a  **push** 0  **push** 1      lcall def\_context\_addr    *; сохранение контекста*  **mov** r0, **sp**  *;количество сохраняемых параметров*  **inc** r0  **mov** r1, #0h  prpm1:  **mov** a, @r1      movx @dptr, a  **inc** r1  **inc** dptr      djnz r0, prpm1        lcall def\_ntx\_tsk      lcall def\_context\_addr        movx a, @dptr  **mov** b, a  **mov** r1, #0h  prpm2:      movx a, @dptr  **mov** @r1, a  **inc** r1  **inc** dptr      djnz r0, prpm2    *; восстановление SFR*  **dec** b  **mov** **SP**, b  **pop** 1  **pop** 0  **pop** a  **pop** b  **pop** PSW  **pop** DPL  **pop** DPH      reti    *; определение DPTR*  def\_ntx\_tsk:  **inc** num\_task  **mov** a, num\_task      cjne a, #3h, exit\_def\_ntf  **mov** num\_task, #0h  exit\_def\_ntf:  **ret**    def\_context\_addr:  **mov** a, num\_task      rr a      rr a      rr a  **mov** r0, a  **mov** dptr, #prog1\_d  **mov** a, DPL  **add** a, r0  **mov** DPL, a  **jnc** exit\_def  **inc** DPH  exit\_def:  **ret**        org 800bh      ljmp tim0        include ASMS\43501\_3\bk\3\p37\clock.asm      include ASMS\43501\_3\bk\3\p37\iklav.asm |

Код подключаемого модуля индикации и клавиатуры (iklav.asm):

|  |
| --- |
| *; iklav.asm*  org 8700h    P4: **equ** E8h  P5: **equ** F8h    indic:  clr P5.0        *; Подготовка к вводу КОМАНД: RS = 0*  **mov** r4, #38h    *; 8-битовый режим обмена с выводом обеих строк*      lcall ind\_wr    *; Запись команды в ЖКИ*  **mov** r4, #0Ch    *; Активизация всех знакомест дисплея без курсора*      lcall ind\_wr  **mov** r4, #80h    *; Адрес нулевой ячейки 1-ой строки*      lcall ind\_wr    **mov** dptr, #FFD0h  **setb** P5.0   *; Подготовка к вводу ДАННЫХ: RS = 1*    *; Выводим 1-ую строку*  wr\_str1:    movx a, @dptr *; Читаем символ из внешней памяти*  **mov** r4, a      lcall ind\_wr  *; Запись данных в ЖКИ*  **inc** dptr      *; Формируем сл. адрес видеобуфера*  **mov** a, dpl  *; Мл. часть dptr*      cjne a, #E4h, wr\_str1*; Проверка окончания вывода символов 1 строки*      clr P5.0  **mov** r4, #C0h      lcall ind\_wr  **setb** P5.0    *; Выводим 2-ую строку*  wr\_str2:    movx a, @dptr *; Читаем символ из внешней памяти*  **mov** r4, a     *; Запись данных в ЖКИ*      lcall ind\_wr  *; Формируем сл. адрес видеобуфера*  **inc** dptr  **mov** a, dpl      cjne a, #0F8h, wr\_str2*; Проверка окончания вывода символов 2 строки*  **ret**    ind\_wr:     **mov** P4, r4 *; Грузим в порт Р4 передаваемую посылку*  **setb** p5.3   *; Установка сигнала Е*      clr p5.2    *; Сигнал R/W=0 (запись)*      lcall delay      clr p5.3    *; Сброс сигнала Е*      lcall delay  **setb** p5.3  **ret**    delay:  **mov** r3, #7  m2: djnz r3, m2  **ret**    memklav:  **mov** 20h, #0h *; 0 for clear C*  **mov** R1, #33h    *; Адрес первой ячейки памяти для просмотра*  **mov** R3, #3h *; счетчик(по строкам и столбцам)*  **mov** 35h, #0h    *; Счётчик нажатых клавиш*  **mov** 37h, #0h    *; Код символа*  **mov** 38h, #0h    *; номер строки*  **mov** 39h, #0h    *; номер столбца*      lcall klav    *; Сначала - проверка на ноль (ничего не нажато)*  zero\_chk:  **mov** C, 0h       *; clear C*  **mov** A, @R1  *; Читаем данные из памяти*  *;mov 56h, R1*      subb A, #f0h    *; Отнимаем 0Fh - если будет ноль, то ничего не нажато.*  *; Иначе считаем, что было какое-нибудь нажатие.*  **jz** skip\_cntr    *; A==0 - пропускаем счётчик нажатий*  **inc** 35h     *; Не ноль - инкремент счётчика нажатий*  **mov** A,@R1  **mov** 37h,A       *; Сохраняем код нажатой клавиши.*  **mov** 38h,R3  *; Сохранили номер строки нажатой клавиши*    skip\_cntr:  **dec** R1      *; Берём следующий элемент из памяти*  *; Пока не достигли конца массива для проверки -*  **dec** R3      *; увеличиваем номер строки*  **mov** C, 0h       *; clear C*      cjne R1, #2Fh, zero\_chk *; - продолжаем цикл*  *; Вышли из цикла проверки отсутствия нажатий*    **mov** A, 35h  *; Грузим в А счётчик нажатий*  **jz** wr\_0     *; 0 нажатий - пишем ноль*  **mov** C, 0h       *; clear C*      cjne A, #01h, wr\_FF     *; больше 1 нажатия - пишем FF*    **mov** dptr, #cdMask   *; начало массива кодов*  **mov** R3,#0h*; ; обнулили счетчик*    find\_column:  **inc** R3*;     ; счетчик номера столбца*  **mov** 39h,R3  *; сохраняем номер столбца*  **mov** A,R3*;*  **mov** C, 0h       *; clear C*      subb A,#5h  **jz** wr\_FF *; Т.к. клавишу точно нажали(или несколько)*  *; ее код обазятельно должен найтись в массиве*  *; иначе - было нажато несколько клавиш, и код не совпал*      movx A, @dptr   *; записали элемент*  **inc** dptr        *; сразу inc индекс в массиве*  **mov** C, 0h       *; clear C*      cjne A, 37h, find\_column *; если число не равно найденному,*  *; продолжим поиск*  get\_num:  *; номер строки\*4+номер столбца*  **mov** A, 38h  **mov** C, 0h       *; clear C*      rl A      rl A            *; два сдвига числа =\*4*  *;add A, 39h     ; получили число*  **add** A, #5h      subb A, 39h  **mov** 34h, A  *; запись числа*      sjmp ext  wr\_0:   **mov** 34h, #0h      sjmp ext  wr\_FF:  **mov** 34h, #FFh      sjmp ext    *; Существующие коды клавиш - характерны для столбца.*  cdMask: **db** E0h, D0h, B0h, 70h    ext:    **ret**  p5: **equ** f8h    klav:   **mov** r0, #30h    *; задаем адрес карты памяти*      orl p5, #f0h    *; настраиваем порт на ввод*  **mov** a, #7fh *; загружаем код бегущего нуля*    mb: **mov** r2, a        rlc a  **mov** p1.7, c      rlc a  **mov** p1.6, c      rlc a  **mov** p1.5, c      rlc a  **mov** p1.4, c  **mov** a, p5   *; считываем данные с клавиатуры*      anl a,#f0h  **mov** @r0, a  *; и запоминаем их*  **inc** r0      *; увеличиваем адрес для записи*  **mov** a, r2      rr a        *; осуществляем сдвиг*      cjne a, #f7h, mb*; выполняем цикл*  **ret**    *;Приведение полученной цифры к десятичному формату*  decim:  **mov** a,34h      cjne a,#ffh, wrff  **mov** a,#46h  **mov** dptr, #str2 + 17      movx @dptr,a  **inc** dptr      movx @dptr,a  **ret**    wrff:   **mov** dptr, #str2 + 17  **mov** a, 34h  **mov** b, #10  **div** ab  **add** a, #30h      movx @dptr, a  **inc** dptr  **mov** a, b  **add** a, #30h      movx @dptr, a  **ret**    *; видеобуффер*          org FFD0h  str1:   **db**  20h, 20h, 20h, 3Ah, 20h, 20h, 20h, 3Ah, 20h, 20h, 20h, 2eh, 20h, 20h, 30h, 30h, 20h, 20h, 20h, 20h  str2:   **db**  'BUTTON NUMBER:      ' |

Код подключаемого модуля таймера (clock.asm):

|  |
| --- |
| *; clock.asm*  org 8300h    ms: **equ** 3Fh *; тики*  sd10:   **equ** 40h *; 100 мс*  sec:    **equ** 41h *;с*  min:    **equ** 42h *;м*  hours:  **equ** 43h *;ч*    *; инициализация*  initt:  *;orl TMOD, #00010000b ;для работы в режиме 16-битного счетчика*        anl TMOD, #1Fh  **mov** TH1, #FEh   *;инициализация счетчика T/C1*  **mov** TL1, #0Bh   *;формирования "тика" 5 мс*    *; наивысший приоритет для T/C1*  **mov** A9h, #08h  **mov** B9h, #00h    *;setb ea    ;разрешение всех прерываний*  **setb** et1    *;разрешение прерываний*  **setb** tr1    *;разрешение счета*  **ret**  clock:    tim1:   **mov** TH1, #FEh    *;2KHz = 500mks*  **mov** TL1, #17h    *; 17*    **inc** ms       *;инкремент тиков*  **mov** r5, ms      clr c  **mov** a, r5      subb a, #200  **jc** end\_tim  *;если количество тиков равно 200*  **mov** ms, #0h      lcall inc\_dec\_sec    end\_tim:    reti    *; инкрементируем мс, сек и мин*  inc\_dec\_sec:  **inc** sd10    *; инкремент 0,1 сек*  **mov** r5, sd10      cjne r5, #64h, end  *; проверка сек == 0,1*  **inc** sec     *; инкремент секунд*  **mov** r5, sec  **mov** sd10, #0h      cjne r5, #3Ch, end  *; проверка сек == 60*  **inc** min     *; инкремент минуты*  **mov** r5, min  **mov** sec, #0h      cjne r5, #3Ch, end  *; проверка мин == 60*  **inc** hours       *; инкремент часы*  **mov** min, #0h    end:    lcall to\_int  **ret**        org 801bh      ljmp tim1        org 8500h    to\_int:    *; ƒл¤ десятых долей секунд*  **mov** a, 40h  **mov** dptr, #FFDEh      lcall overal  **dec** dpl      lcall overal    *; ƒл¤ секунд*  **mov** a, 41h  **mov** dptr, #FFDAh      lcall overal  **dec** dpl      lcall overal    *; ƒл¤ минут*  **mov** a, 42h  **mov** dptr, #FFD6h      lcall overal  **dec** dpl      lcall overal    *; ƒл¤ часов*  **mov** a, 43h  **mov** dptr, #FFD2h      lcall overal  **dec** dpl      lcall overal  **ret**    overal: **mov** b, #10d *;основание системы счисления*  **div** ab  **mov** r1, a  **mov** a, b  **add** a, #30h *;ASCII символа*      movx @dptr, a   *;символ*  **mov** a, r1  **ret** |



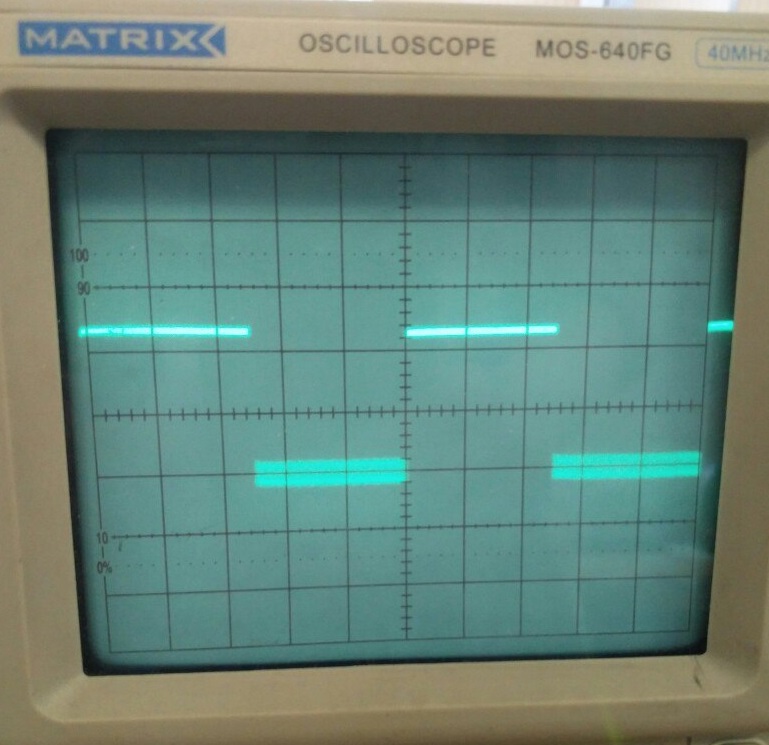
*Рис. 26. Многозадачная операционная система*

Каждая из задач 1-3 инвертирует соответствующий ей бит P1.1-P1.3. Воспользуемся этим для контроля времени выполнения каждой задачи. Для того, чтобы провести чистый эксперимент, закомментируем код задач, оставив только инверсию P1.1-P1.3:

|  |
| --- |
| *( . . . )*  *; Задача1 - определение номера нажатой клавиши и преобразование в ASCII код*  prog1:  **cpl P1.1**  *; lcall memklav*  *; lcall decim*      sjmp prog1    *; Задача 2 - индикация номера нажатой клавиши*  prog2:  **cpl P1.2**  *; lcall indic*      sjmp prog2    *; Задача 3 - электронные часы*  prog3:  **cpl P1.3**  *; lcall clock*      sjmp prog3  *( . . . )* |

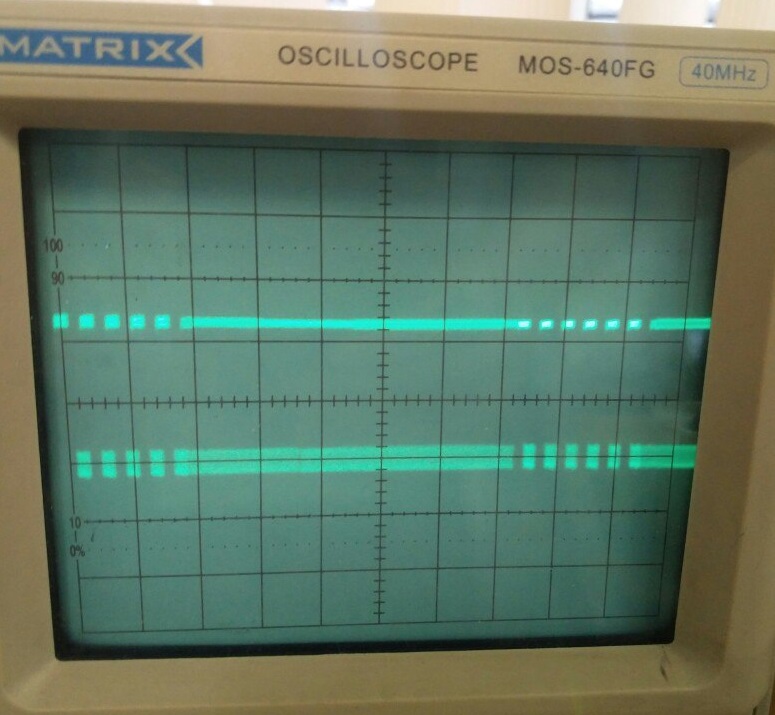
Была проконтролирована последовательность переключения выполняемых задач и правильность сохранения/восстановления контекстов задач при их переключении.

В первую очередь, мы убедились в правильности работы таймера:

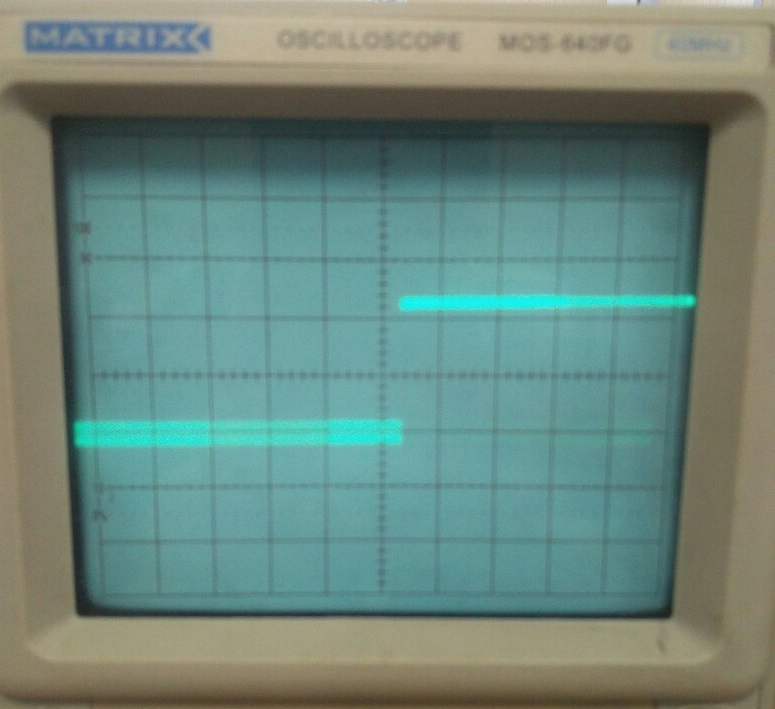


*Рис. 27. Задача 0. Таймер ОС*

Длительность импульса равна 2.5 мс, как и ожидалось. Теперь проверим, что каждая из задач не превышает выделенного ей кванта времени в 2.5 мс.



*Рис. 28. Задача 1. Опрос клавиатуры*



*Рис. 29. Задача 2. Индикация*



*Рис. 30. Задача 3. Секундомер*

В ходе экспериментов, мы убедились, что все задачи работают правильно. Операционная система правильно выводит время и номер нажатой клавиши. Также все три задачи правильно переключаются, о чем свидетельствуют данные с осциллографа.

1. **Вывод**

Система таймеров MKSAB80C515 имеет в своем составе три счетчика которые могут быть запрограммированы на работу в различных режимах. В данной работе система таймеров успешно была использована для формирования меандра требуемой частоты (то есть программной реализации делителя частоты), формирования ШИМ-сигнала, реализации электронных часов, квантования работы многозадачной операционной системы.

При обработке прерываний необходимо помнить о том, что на обработчик прерываний также тратится некоторое время процессора. В большей части программ это не критично, однако для программ, требующих точности, необходимо вводить коррекцию.

Команда lcall 128h позволяет отлаживать работу микроконтроллерных систем, назначение которых формирование выходного сигнала, в зависимости от входного. Этот метод очень полезен, не только для задания собственных сигналов, но и для поиска ошибок в работе программы.

Экспериментально было подтверждено, что точность счета таймеров достаточна для реализации электронных часов и секундомеров. Эксперимент заключался в том, что одновременно запускается секундомер на компьютере и на микроконтроллере. Через какое-то время они одновременно останавливаются и анализируется результат. На интервале 5 минут секундомер показал значение 05:00:134, а микроконтроллер 04:59:7 и погрешность составила 0.13%, что скорее объясняется скорее скоростью реакции человека, останавливающего секундомер и программу, чем неправильными результатами программы.

Очень важную роль играет система приоритетов прерываний. Она позволяет настроить 4 уровня приоритета для шести пар прерываний. В крупных вычислительных системах управления объектами может возникать большое число запросов на прерывание, и они должны обрабатываться с разными приоритетами. Так, например, для таймера, который должен обрабатываться в режиме реального времени, должен быть выбран максимальный приоритет, поскольку низкий уровень приоритета может не позволить ему реализовывать функции отсчёта временных промежутков.