

Санкт-Петербургский государственный политехнический университет  
Факультет технической кибернетики  
Кафедра компьютерных систем и программных технологий

## **Отчёт по лабораторной работе**

«Изучение таймеров и системы прерываний  
по дисциплине "Микропроцессорные системы»

Работу выполнил студент  
Дорофеев Юрий Владимирович гр. 4081/12 \_\_\_\_\_

Работу принял преподаватель  
Павловский Евгений Григорьевич \_\_\_\_\_

г. Санкт-Петербург  
2012

## Содержание

1. Цель работы .....	3
2. Теоретические сведения .....	3
2.1. Блок таймеров/счетчиков .....	3
2.1. Система прерываний МК SAB80C515.....	5
3. Программа работы .....	7
4. Выполнение программ.....	8
4.1. Формирование меандра .....	9
4.2 Формирование ШИМ-сигнала .....	12
4.3 Формирование ШИМ-сигнала с управлением с помощью инструментальной ЭВМ .....	16
4.4 Электронные часы .....	18
4.5 Электронный секундомер .....	23
4.6 Модифицированная программа «Электронные часы».....	26
4.7 Многозадачная операционная система с разделением времени .....	30
5. Выводы .....	36

## 1. Цель работы

- Приобретение практических навыков программирования таймеров;
- Изучение принципов программного деления частоты;
- Знакомство с организацией и использованием многоуровневой системы прерываний.

## 2. Теоретические сведения

### 2.1. Блок таймеров/счетчиков

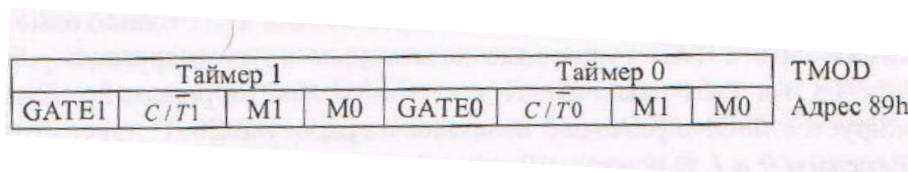
Микроконтроллер МК SAB 80C515 содержит в своём составе три программируемых 16-разрядных таймера-счетчика – T/C0, T/C1 и T/C2. Таймеры T/Cx (x = 0, 1, 2) реализуются на основе 16-разрядных суммирующих счётчиков со схемами управления. Все три таймера могут работать либо в режиме таймера, либо в режиме счётчика событий. Программно доступными регистрами таймеров T/C0, T/C1 и T/C2 микроконтроллера являются регистры блока SFR:

- TH0 и TL0 для таймера 0
- TH1 и TL1 для таймера 1
- TH2 и TL2 для таймера 2

Таймеры T/C0 и T/C1 имеют одинаковую внутреннюю структуру и могут использоваться в качестве программно 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 счетчика внешних событий. Хотя основное его назначение связано с реализацией на его основе функций быстрого ввода/вывода.

Для задания режимов работы таймера T/Cx (x = 0, 1) и его управления используются два регистра блока SFR – регистр режимов TMOD и регистр управления TCON. Форматы регистров приведены на рис. 1 и рис. 3. Регистр TMOD содержит набор программно управляемых бит (по 4 бита для каждого таймера), используемых для задания режима T/C.



Таймер 1				Таймер 0				TMOD Адрес 89h
GATE1	C/T1	M1	M0	GATE0	C/T0	M1	M0	

Рис. 1. Формат регистра TMOD

- GATE – бит управления блокировкой таймера;
- C/T – бит, определяющий функцию: таймер или счетчик;
- M1, M0 – биты, задающие режим работы;

M1	M0	Режим работы таймера
0	0	8-битный Т/С на базе регистра ТНх (х=0, 1). Регистр ТLх используется как 5-разрядный предделитель
0	1	16-битный Т/С. Регистры ТНх и ТLх включены последовательно как один 16-битный регистр
1	0	Автогенератор на основе 8-разрядного регистра ТLх. Регистр ТНх содержит значение, которое загружается в ТLх при каждом переполнении ТLх
1	1	8-битный Т/С на базе регистра ТL0, управляемый битом ТR0, и 8-битный таймер на базе ТН0, управляемый битом ТR1. Таймер Т/С1 остановлен

Рис. 2. Формат и назначение отдельных бит регистра TMOD

8Fh	8Eh	8Dh	8Ch	8Bh	8Ah	89h	88h	TCON
TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0	Адрес 88h

Рис. 3. Формат регистра TCON

Регистр TCON для каждого таймера содержит бит управления разрешением работы TRx (x = 0, 1), 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.

Таймер/счетчик Т/С2 наряду с реализацией функций счета 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. Режим работы таймера задается программно с помощью управляющих бит регистра T2CON блока SFR.

0CFh	0Ceh	0CDh	0CCh	0CBh	0CAh	0C9h	0C8h	T2CON
T2PS	I3FR	I2FR	T2R1	T2R0	T2CM	T2I1	T2I0	Адрес 0CBh

Рис. 4. Формат регистра T2CON

Биты T2I1 и T2I0 задают режим работы таймера и определяют источник входных счетных импульсов.

T2I1	T2I0	Режим работы таймера Т/С2
0	0	Таймер 2 остановлен. Счетные импульсы на вход не поступают.
0	1	Режим неуправляемого таймера. На вход таймера поступают импульсы с частотой $f_{CR}/12$ (T2PS = 0) или $f_{CR}/24$ (T2PS = 1)
1	0	Режим управляемого таймера. На вход таймера поступают счетные импульсы с частотой $f_{CR}/12$ (T2PS = 0) или $f_{CR}/24$ (T2PS = 1)
1	1	Режим счетчика внешних событий, поступающих на вход Т2/P1.7

Рис. 5. Назначение бит T2I1 и T2I0

Биты T2R1 и T2R0 назначают режим перезагрузки регистров ТН2:ТL2 таймера. Режимы перезагрузки могут использоваться для вызова подпрограмм обслуживания, которые должны выполняться с определенной периодичностью.

T2R1	T2R0	Режим перезагрузки
0	X	Перезагрузка таймера 2 запрещена
1	0	Режим 0: автоматическая перезагрузка импульсами переполнения таймера T0V2
1	1	Режим 1: перезагрузка T/C2 по сигналу перехода из 1 в 0 на входе T2EX/P1.5

Рис. 6. Назначение бит T2R1 и T2R0

Бит T2PS – бит разрешения работы делителя на 2 частоты внутреннего источника счетных импульсов таймера 2. Бит T2CM, I2FR, I3FR не управляют работой таймера T/C2. Эти биты назначают один из двух режимов сравнения бока HSIO (бит T2CM) или определяют вид входных сигналов внешних прерываний (биты I2FR и I3FR).

## 2.1. Система прерываний МК SAB80C515

Любой таймер/счетчик тесно и неотделимо связан с системой прерываний. Система прерываний – это наиболее простое средство обеспечения выполнения программ в мультизадачном режиме. Другой способ использования прерываний – это реализация задач синхронизации. Это позволяет выполнять более быстродействующие процедуры управления без «оглядки» на остальные, которые по прерыванию будут приостановлены.

Система прерываний является многовекторной и многоуровневой. Она объединяет 6 внутренних и 8 внешних источников запросов, которые с помощью логических схем микроконтроллера 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. Очередность обслуживания прерывания определяется механизмом 4-уровневой системы приоритетного прерывания.

Все источники прерываний МК объединены попарно, образуя 6 пар (рис. 7). Внутри каждой пары источник прерывания, указанный слева имеет более высокий приоритет по сравнению со вторым источником пары.

Источник прерывания со старшим приоритетом в паре	Имя источника прерывания	Адрес вектора прерывания	Источник прерывания с младшим приоритетом в паре	Имя источника прерывания	Адрес вектора прерывания	Приоритет пары источников прерываний
Внешнее прерывание 0	IE0	0003h	Прерывание АЦП	IADC	0043h	0
Переполнение таймера 0	TF0	000Bh	Внешнее прерывание 2	IE2	004Bh	1
Внешнее прерывание 1	IE1	0013h	Внешнее прерывание 3	IE3	0053h	2
Переполнение таймера 1	TF1	001Bh	Внешнее прерывание 4	IE4	005Bh	3
Прерывание SP	TI или RI	0023h	Внешнее прерывание 5	IE5	0063h	4
Переполнение таймера 2 или внешняя перезагрузка	TF2 EXF2	002Bh	Внешнее прерывание 6	IE6	006Bh	5

Рис. 7. Пары источников прерываний и приоритеты прерываний внутри уровня

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

Всего различных приоритетов 4 (от 0 до 3), чем больше значение приоритета, тем выше и сама приоритетность исполнения. Любая задача может быть прервана только задачей с более высоким уровнем приоритета, и только процедуры с приоритетом 3 не могут быть прерваны ни одной другой. Уровни приоритетов справедливы для процедур обработчиков прерываний, обычные же программы не имеют приоритета и являются самыми «бесправными», то есть могут быть прерваны любым обработчиком.

Значения бит		Задаваемый глобальный уровень приоритета
IP1.x	IP0.x	
0	0	Уровень приоритета 0 (низший)
0	1	Уровень приоритета 1
1	0	Уровень приоритета 2
1	1	Уровень приоритета 3 (высший)

*Рис. 8. Кодирование глобального уровня приоритета пары источников запросов прерывания*

Поскольку обработчики прерываний приостанавливают исполнение текущей программы, то для избегания ошибок исполнения прерванной программы, а также запаздывания в управлении, необходимо предусматривать сохранение (а по выходу восстановление) системных регистров программы при выходе обработчика. Также он не должен быть длинным, с точки зрения времени исполнения, чтобы не организовывать большого числа вложенных прерываний и растягивания времени ожидания прерванной программы.

### 3. Программа работы

1. Изучить работу таймеров-счетчиков микроконтроллера и принципы организации системы прерываний.
2. Разработать программу, генерирующую на выбранном разряде порта МК меандр с частотой  $N_{\text{клав}} * 100$  Гц. Работу с линией порта проводить в обработчике прерываний, генерируемых таймером T/C0. Для управления генерируемой частотой использовать клавиатуру.
3. Разработать программу формирования ШИМ-сигнала с частотой 1000 Гц и регулируемой скважностью от АЦП.
4. Модифицировать программу п.3 так, чтобы управляющий код задавался через «Окна управления».
5. Реализовать на микроконтроллере счетчик времени с отображением на ЖКИ текущего времени с точностью 0,1 с. В качестве опорного программного генератора использовать прерывания таймера.
6. Разработать программу, определяющую интервал времени между прерываниями INT0 и INT1, генерируемых при нажатии двух клавиш разных столбцов клавиатуры стенда.
7. Модифицировать программу п.5 так, чтобы с помощью «окон управления» выводились доли секунды. Использовать разные приоритеты запроса прерываний.
8. Разработать программу управления простейшей многозадачной системой с разделением времени.

## 4. Выполнение программ

Для выполнения работ данного цикла разработана структура информационных связей МК с подключаемыми внешними устройствами (клавиатурой, модулем ЖКИ, датчикам аналоговых сигналов):

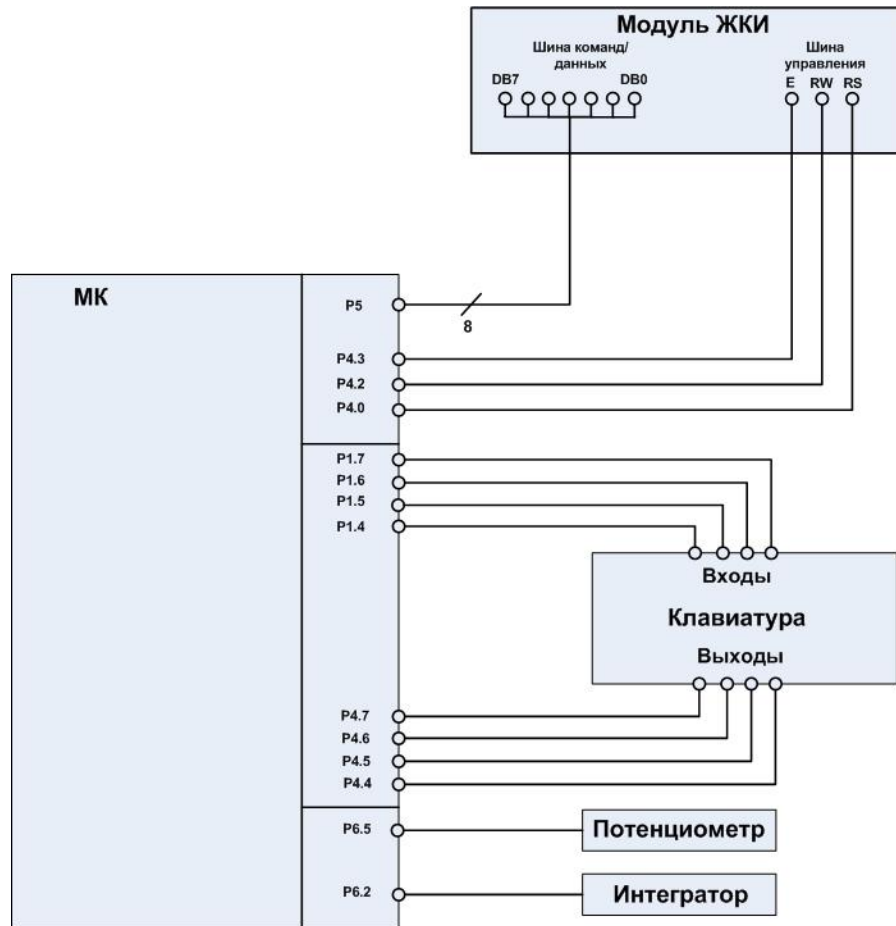


Рис. 9. Структура информационных связей МК с подключаемыми внешними устройствами



#### 4.1. Формирование меандра

По заданию требуется разработать программу, генерирующую на заданном разряде порта МК импульсный сигнал прямоугольной формы (меандр) с частотой, определяемой номером  $N_{\text{клав}}$  нажатой клавиши блока клавиатуры ( $F = N_{\text{клав}} * 100\text{Гц}$ ).

##### *Описание программы:*

В ходе выполнения данной программы происходит постоянный опрос клавиатуры на номер нажатой клавиши. Номер нажатой клавиши определяет значение частоты, с которой будет формироваться меандр.

Расчётные значения частоты занесены во внешнюю память. Их значения представлены в таблице 1.

*Таблица 1. Расчётные значения частоты*

F, Гц	T, мс	Время импульса	Время паузы	Константа для импульса	Константа для паузы
100	10	4	6	F0 60	E8 90
200	5	2	3	F8 30	F4 48
300	3	1	2	FC 17	F8 2F
400	2,5	1	1,5	FC 17	FA 6C
500	2	1	1,2	FC E0	FB 10
600	1,7	0,68	1	FD 58	FC 18
700	1,4	0,56	0,84	FD D0	FC D8
800	1,2	0,48	0,72	FE 20	FD 30
900	1,1	0,44	0,66	FE 48	FD 6C
1000	1	0,4	0,6	FE 70	FD A8
1100	0,9	0,36	0,54	FE 98	FD E4
1200	0,8	0,32	0,48	FE C0	FE 20
1300	0,75	0,3	0,45	FE D4	FE 3E
1400	0,7	0,28	0,42	FE E8	FE 5C
1500	0,65	0,26	0,39	FE FC	FE 7A
1600	0,6	0,24	0,36	FF 10	FE 98
50	20	8	12	E0 C0	D1 20

Одновременно с опросом клавиатуры запущен встроенный в микроконтроллер таймер T/C0. В каждый тик таймера происходит формирование импульса согласно установленным значениям частоты. Для формирования импульса и паузы используется бит 0, который инвертируется в обработчике прерывания таймера.

**Алгоритм:**

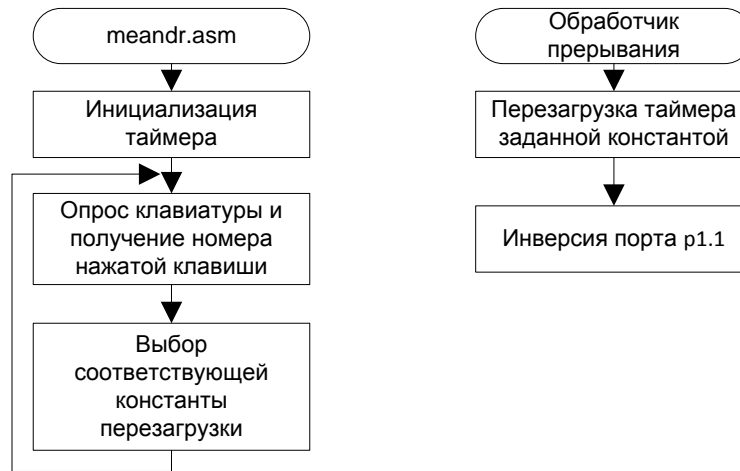


Рис. 10. Алгоритм программы «Формирование меандра»

**Код программы:**

```
org 8100h

lcall init

; опрос клавиатуры
loop: lcall klav1
      mov a,34h
      cjne a,#FFh,cont
      sjmp loop

; выбор значения частоты, соответствующей номер нажатой клавиши
cont: mov dptr, #table
      mov b,#4h
      mul ab
      mov b,a
      movc a,@a+dptr
      mov r3,a
      inc b
      mov a,b
      movc a,@a+dptr
      mov r4,a
      inc b
      mov a,b
      movc a,@a+dptr
      mov r5,a
      inc b
      mov a,b
      movc a,@a+dptr
      mov r6,a
      sjmp loop

;получаем значение частоты в регистрах r3 - r6
init: anl TMOD, #11110000b
      orl TMOD, #00000001b
      mov r3,#F0h
      mov r4,#60h
      mov r5,#e8h
      mov r6,#90h
      setb ea
      setb et0
      setb tr0
      ret
```

```

        include asms\4081_4\Dorofeev\5.3\2.asm
        include asms\4081_4\Dorofeev\5.3\klav1.asm
; 0-бит - бит, отвечающий за смену импульса и паузы
tim0:  jb 0, l1
        mov TH0, r3
        mov TL0, r4
        jmp l2
l1:     mov TH0, r5
        mov TL0, r6
l2:     cpl 0
        cpl p1.1
        reti

table:
        db      f0h, 60h, e8h, 90h;      100Hz
        db      f8h, 30h, f4h, 48h;      200Hz
        db      fch, 17h, f8h, 2fh;      300Hz
        db      fch, 17h, fah, 6ch;      400Hz
        db      fch, e0h, fbh, 50h;      500Hz
        db      fdh, 58h, fch, 18h;      600Hz
        db      fdh, d0h, fch, d8h;      700Hz
        db      feh, 20h, fdh, 30h;      800Hz
        db      feh, 48h, fdh, 6ch;      900Hz
        db      feh, 70h, fdh, a8h;      1000Hz
        db      feh, 98h, fdh, e4h;      1100Hz
        db      feh, c0h, feh, 20h;      1200Hz
        db      feh, d4h, feh, 3eh;      1300Hz
        db      feh, e8h, feh, 5ch;      1400Hz
        db      feh, fch, feh, 7ah;      1500Hz
        db      ffh, 10h, feh, 98h;      1600Hz
        db      e0h, c0h, d1h, 20h;      50Hz

org 800bh
ljmp tim0

```

### **Результаты выполнения:**

В результате выполнения программы получаем экспериментальные значения частоты. Расчётные и экспериментальные значения частоты в зависимости от номера нажатой клавиши представлены в таблице 2.

*Таблица 2.Расчётные и экспериментальные значения частоты*

№ нажатой клавиши	Расчетное значение частоты	Значение частоты, измеренное осциллографом
1	100	98,1
2	200	196,2
3	300	306,9
4	400	408,7
5	500	489,2
6	600	586,3
7	900	719,2
8	100	795,5
9	200	903,8
10	300	1007,6
11	400	1094,2

12	500	1198,3
13	600	1303,3
14	900	1401,8
15	600	1508,1
16	900	1596,6

## 4.2 Формирование ШИМ-сигнала

По заданию необходимо сформировать ШИМ-сигнал заданной частоты и регулируемой скважности. Частота определяется номером рабочего места и равна 400 Гц.

Требуемая частота ШИМ-сигнала  $F_{pwm}$  определяется константой перезагрузки  $N_{загр}$ .

$$N_{загр} = 65536 - \frac{106}{F_{pwm} * 2}$$

Так как скважность имеет переменные значения, то и  $N_{загр}$  будет иметь переменные значения  $N_{загр_i}$ . Каждое такое значение вычисляется по формуле.

$N_{загр_i} = N_{загр0} + N_{ti}$ , где  $N_{ti}$  – масштабированное значение управляющего кода.

Вычислим коэффициент масштабирования  $K_m$ .

$$K_m = \frac{N_{загр\_max} - N_{загр0}}{255} = \frac{FFFF - FC17}{255}$$

$K_{mint} = 3$  – старший байт, который определяет целую часть

$K_{mfract} = 9$  – младший байт, который определяет дробную часть

Теперь мы можем определить промасштабированное значение управляющего кода:

(1)  $N_{ti} = i * K_m$ , где  $i$  – число дискрет управляющего кода ( $i = 0-255$ ).

### Пояснения к программе:

Скважность ШИМ-сигналов в данной работе регулируется значением на потенциометре, которое перед этим проходит аналого-цифровое преобразование.

Сначала необходимо получить значение с выхода АЦП.

Затем рассчитывается  $N_{ti}$  – промасштабированное значение управляющего кода.

По формуле (1) происходит расчёт  $N_{ti}$ . В качестве  $i$  подставляется значение, формируемое на выходе АЦП. Полученное значение загружается в регистр T2CON.

### Алгоритм:

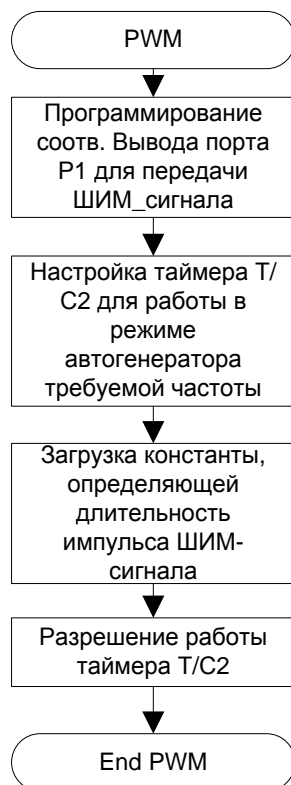


Рис. 11. Алгоритм программы «Формирование ШИМ-сигнала»

### Код программы:

```
org 8400h

m1:  lcall adc_p ; определение значения на выходе потенциометра с помощью АЦП
      lcall pwm  ; вызов подпрограммы формирования ШИМ-сигнала
      sjmp m1
      ret

include asms\4081_4\Dorofeev\5.3\adc_p.asm
include asms\4081_4\Dorofeev\5.3\pwm.asm

; файл adc_p.asm
adc_p:
ADCON: equ D8h
ADDAT:  equ D9h
DAPR:   equ Dah
U:      equ 40h

      mov a, #00000011
      anl ADCON, #E0h
      orl ADCON, a
      mov DAPR, #0h
      mov r7, #15
r1:    djnz R7, r1
mov U, ADDAT ; запись цифрового эквивалента аналогового сигнала на
              ; выходе потенциометра в ячейку по адресу 40h
      ret
```

```

; содержимое файла pwm.asm
pwm:
T2CON:      equ    C8h

; определение Nti (промасштабированный управляющий код)
mov r2, #FCh          ; Nзарп0
mov r3, #17h

mov a, 40h            ; запись в аккумулятор цифрового кода с выхода АЦП
mov b, #3h
mul ab                ; умножение цифрового кода с выхода АЦП на Kmint (старший байт
Км)
mov 41h, b
mov 42h, a
mov a, 40h
mov b, #EAh
mul ab                ; умножение цифрового кода с выхода АЦП на Kmfract
(младший байт Км)
mov a, b
add a, 42h
mov 42h, a
mov a, 41h
addc a, #0h
mov 41h, a

; определение NзарпI и его сохранение в ячейках 41h, 42h
mov a, r3
add a, 42h
mov 42h, a
mov a, 41h
addc a, r2
mov 41h, a

; настройка таймера T/C2 и разрешение его работы
orl P1, #00000100b
mov T2CON, #0
mov CBh, r2
mov CAh, r3
mov C1h, #00100000b
mov C5h, 41h
mov C4h, 42h
mov T2CON, #00010001b
ret

```

### Результаты выполнения:

Таблица 3. Результат выполнения программы

U <sub>пот</sub> , В	τ <sub>pwm</sub> , мс
0	0
1	0,18
2	0,38
3	0,58
4	0,77
5	0,99

График зависимости  $t=f(U)$  представлен на рис.12.

График зависимости  $t=f(U)$

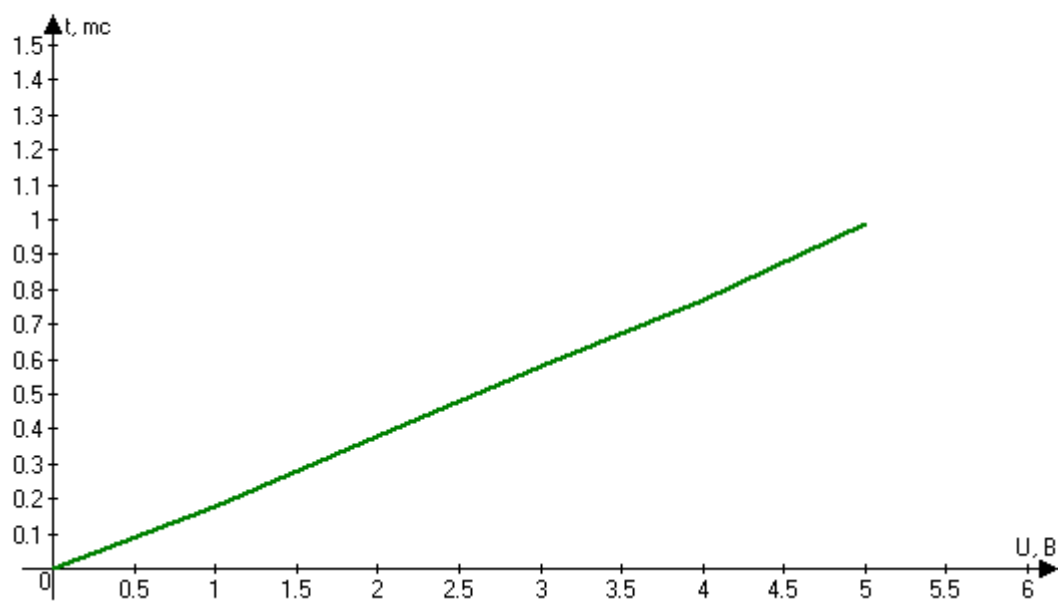


Рис.12. Зависимость  $t=f(U)$

### 4.3 Формирование ШИМ-сигнала с управлением с помощью инструментальной ЭВМ

По заданию необходимо сформировать ШИМ-сигнал, используя для управления скважностью ШИМ-сигнала цифровые коды, управляющего воздействия, формируемого инструментальной ЭВМ при работе с вкладкой «Окна управления».

#### Описание программы:

С помощью воздействия, формируемого инструментальной ЭВМ, при работе с вкладкой «Окна управления» оболочки Shell-51 осуществляется управление скважностью ШИМ-сигнала. Для этого используется команда `lcall 128h`, реализующая переход на подпрограмму однократного сеанса связи с инструментальной ЭВМ. При выполнении этой команды, ЭВМ выдаёт очередное значение управляющего кода. График управляющего воздействия загружается с помощью кнопки «Открыть» вкладки «Окна управления» оболочки Shell-51. Для модификации программы необходимо изменить основную программу.

#### Код программы:

```
org 8400h

m1:  lcall 128      ; однократный сеанс связи с инструментальной ЭВМ
      lcall pwm     ; вызов подпрограммы формирования ШИМ-сигнала
      sjmp m1
      ret

include asms\4081_4\Dorofeev\5.3\pwm.asm

; pwm.asm
pwm:
T2CON:      equ     C8h

      ; определение  $N_{ti}$  (промасштабированный управляющий код)
      mov r2, #FCh      ;  $N_{zarp0}$ 
      mov r3, #17h

      mov a, 40h        ; запись в аккумулятор цифрового кода с выхода АЦП
      mov b, #3h
      mul ab             ; умножение цифрового кода с выхода АЦП на  $K_{mint}$  (старший байт
Km)
      mov 41h, b
      mov 42h, a
      mov a, 40h
      mov b, #EAh
      mul ab             ; умножение цифрового кода с выхода АЦП на  $K_{mfract}$ 
(младший байт Km)
      mov a, b
      add a, 42h
      mov 42h, a
      mov a, 41h
      addc a, #0h
      mov 41h, a

      ; определение  $N_{zarpI}$  и его сохранение в ячейках 41h, 42h
      mov a, r3
      add a, 42h
```



```

mov 42h, a
mov a, 41h
addc a, r2
mov 41h, a

; настройка таймера T/C2 и разрешение его работы
orl P1, #00000100b
mov T2CON, #0
mov CBh, r2
mov CAh, r3
mov C1h, #00100000b
mov C5h, 41h
mov C4h, 42h
mov T2CON, #00010001b
ret

```

Подадим в качестве управляющего сигнала сигнал треугольной формы (рис.13) с вкладки «окна управления»

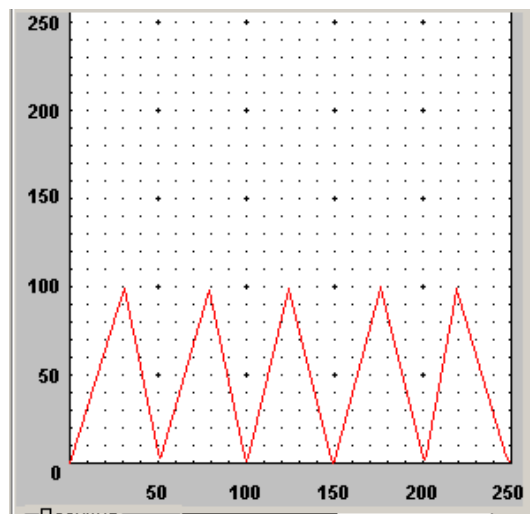


Рис.13. Изменение константы перезагрузки с помощью инструментальной ЭВМ (треугольный сигнал)

В результате на выходе порта P1.2 с помощью осциллографа можно наблюдать циклическое уменьшение и увеличение скважности сигнала.

При подаче трапецеидального сигнала (рис.14.) скважность сигнала, наблюдаемого на выходе порта P1.2 сначала уменьшается, далее через некоторое время снова увеличивается.

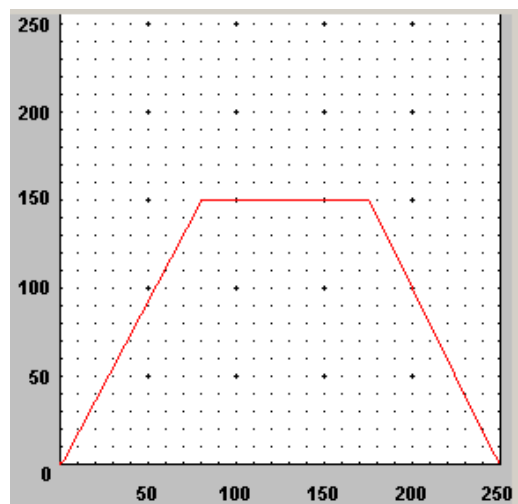


Рис.14. Изменение константы перезагрузки с помощью инструментальной ЭВМ (трапецеидальный сигнал)

## 4.4 Электронные часы

По заданию необходимо разработать программу «Электронные часы» с отображением на экране ЖКИ текущего времени с точностью 0,1 с. В качестве счётных импульсов временных «тиков» используются прерывания таймера, следующие с частотой 2 КГц.

Программный счётчик «тиков» реализован в фоновой циклической программе.

### Описание программы:

В начале программы происходит инициализация часов. Встроенный в микроконтроллер таймер формирует аппаратные тики. Обработчик прерываний таймера вызывается каждые 500 мкс. В нём происходит подсчёт аппаратных «тиков». При достижении 200 «тиков» (0,1 мс) происходит инкремент счётчика долей секунды. При достижении счётчика долей секунд до значения 10 происходит его сброс и инкремент счётчика секунд. При достижении счётчика секунд до значения 60 происходит его сброс и инкремент счётчика минут. При достижении счётчика минут до значения 60 происходит его сброс и инкремент счётчика часов. Значения для электронных часов записываются в ячейки внутренней памяти. Параллельно с подсчётом аппаратных «тиков» происходит перевод значений для часов в ASCII-коды и вывод на экран ЖКИ.

### Алгоритм:

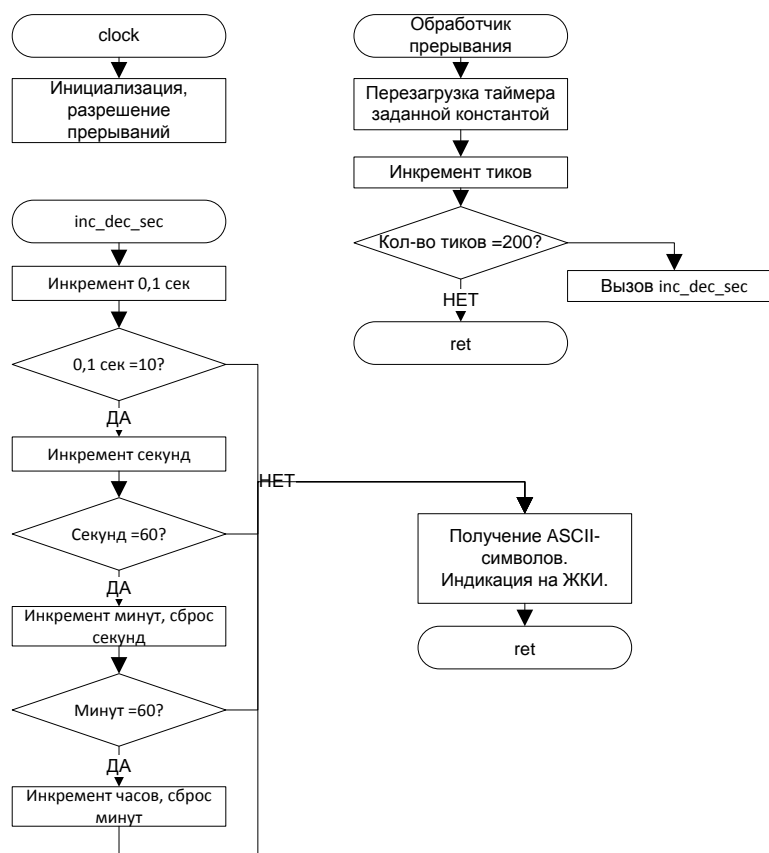


Рис. 15. Алгоритм программы «Электронные часы»

### Код программы:

```
org 8400h

ms:    equ 3Fh      ;тики
sd10:  equ 40h      ;секунды / 10
sec:   equ 41h      ;секунды
min:   equ 42h      ;минуты
hours: equ 43h      ;часы

    lcall init

loop: sjmp loop

; Инициализация
init: anl TMOD, #11110000b ;инициализация таймера
      orl TMOD, #00000001b ;для работы в режиме 16-битного счётчика

      mov TH0, #ECh          ;инициализация счётчика T/C0 для
      mov TL0, #77h          ;формирования "тика" 5 мс

      ; Инициализация часов
      mov ms, #0h
      mov sd10, #0h
      mov sec, #0h
      mov min, #0h
      mov hours, #0h

      setb ea                ;разрешение всех прерываний
      setb et0               ;разрешение прерывания TC0
      setb tr0               ;разрешение счёта
      ret

tim0:   mov TH0, #FEh        ;2KHz = 500mks
        mov TL0, #0Bh
        cpl p1.2
        inc ms                ;инкремент тиков
        mov r5, ms
        clr c
        mov a, r5
        subb #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, #Ah, 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
```

```

lcall indic2
ret

org 800bh
ljmp tim0

include asms\4081_4\Dorofeev\5.3\clock\to_int.asm
include asms\4081_4\Dorofeev\5.3\clock\indic2.asm

```

### ***to\_int.asm***

```

;программа to_int.asm
to_int:
B:     equ F0h
       mov a, 40h

; Для десятых долей секунд
       mov B, #10d ;основание системы счисления
       div ab
       mov r1, a
       mov a, b
       add a, #30h ;ASCII третьего символа
       mov dptr, #FFDDh
       movx @dptr, a      ;третий символ
       mov a, r1

; Для секунд
       mov a, 41h
       mov B, #10d ;основание системы счисления
       div ab
       mov r1, a
       mov a, b
       add a, #30h ;ASCII третьего символа
       mov dptr, #FFDAh
       movx @dptr, a      ;третий символ
       mov a, r1

       mov B, #10d ;основание системы счисления
       div ab
       mov r1, a
       mov a, b
       add a, #30h ;ASCII второго символа
       mov dptr, #FFD9h
       movx @dptr, a      ;второй символ
       mov a, r1

; Для минут
       mov a, 42h
       mov B, #10d ;основание системы счисления
       div ab
       mov r1, a
       mov a, b
       add a, #30h ;ASCII третьего символа
       mov dptr, #FFD6h
       movx @dptr, a      ;третий символ
       mov a, r1

       mov B, #10d ;основание системы счисления
       div ab
       mov r1, a
       mov a, b
       add a, #30h ;ASCII второго символа

```

```

        mov dptr, #FFD5h
        movx @dptr, a      ;второй символ
        mov a, r1

; Для часов
        mov a, 43h
        mov B, #10d ;основание системы счисления
        div ab
        mov r1, a
        mov a, b
        add a, #30h ;ASCII третьего символа
        mov dptr, #FFD2h
        movx @dptr, a      ;третий символ
        mov a, r1

        mov B, #10d ;основание системы счисления
        div ab
        mov r1, a
        mov a, b
        add a, #30h ;ASCII второго символа
        mov dptr, #FFD1h
        movx @dptr, a      ;второй символ
        mov a, r1
        ret

```

### ***indic2.asm***

```

        mov 36h, #25h
p5:     equ 0F8h
w1:     equ 20h      ;определение переменных
w0:     equ 21h

        ;основная программа
indic2:  mov w1, #0
        mov w0, #38h
        lcall ind_wr
        mov w0, #0Ch
        lcall ind_wr
        mov w0, #80h
        lcall ind_wr
        mov w1, #1
        mov dptr, #FFD0h
wr_str1: movx a, @dptr      ;вывод первой строки
        mov w0, a
        lcall ind_wr
        inc dptr
        mov a, dpl
        cjne a, #0E4h, wr_str1
        mov w1, #0
        mov w0, #C0h
        lcall ind_wr
        mov w1, #1
wr_str2: movx a, @dptr      ;вывод второй строки
        mov w0, a
        lcall ind_wr
        inc dptr
        mov a, dpl
        cjne a, #0F8h, wr_str2
        ret

        ;подпрограмма записи информации в ЖКИ
ind_wr:  mov p5, w0
        setb pl.7

```

```

        clr p1.6
        mov a, w1
        mov c, acc.0
        mov p1.4, c
            lcall delay
            clr p1.7
            lcall delay
            setb p1.7
            ret

        ;подпрограмма задержки
delay:    nop
        nop
        nop
        nop
        nop
        nop
        nop
        nop
        nop
        nop
        nop
        nop
        ret

        ;резервирование памяти под видеобуфер
        org FFD0h
str1: db 20h, 20h, 20h, 68h, 20h, 20h, 20h, 6Dh, 20h, 20h, 20h, 73h, 20h,
20h, 6Dh, 73h, 20h, 20h, 20h, 20h
str2: db AFh, BBh, 65h, BAh, BFh, 70h, 6Fh, BDh, BDh, C3h, 65h, 20h, C0h,
61h, 63h, C3h, 20h, 20h, 20h, 20h

```

### ***Результаты программы:***

Стоит отметить, что работа счетчика 500 мкс «тиков» немного отличается от работы остальных счетчиков (десятых долей секунд, секунд и минут). Вместо непосредственного сравнения командой **cjne** выполняется вычитание из содержимого счетчика коэффициента пересчета и последующая проверка флага **C**. Это сделано для исключения возможных ошибок пересчета, которые могут возникать по следующей причине: анализ содержимого счетчика «тиков» осуществляется в фоновой программе и производится не после каждого «тика». Следовательно, не исключена ситуация, когда при непосредственном сравнении содержимое счетчика окажется больше значения коэффициента пересчета и счетчик продолжит считать дальше.

Результат работы программы:

Как и ожидалось, на дисплее ЖКИ отображалось значение минут, секунд и долей секунд.

## 4.5 Электронный секундомер

По заданию необходимо разработать электронный секундомер, который определяет интервал времени между внешними прерываниями Int0 и Int1, генерируемые при нажатии двух клавиш разных столбцов клавиатуры стенда.

### Описание программы:

Алгоритм работы электронного секундомера основан на работе электронных часов. Добавлены обработчики прерываний int0 и int1.

В обработчике Int0 запрещается прерывание int0, разрешается прерывание int1 и разрешается счёт аппаратных «тиков», то есть происходит запуск секундомера.

В обработчике Int1 запрещается прерывание int1, разрешается прерывание int0 и запрещается счёт аппаратных «тиков», то есть происходит останов секундомера.

### Алгоритм:

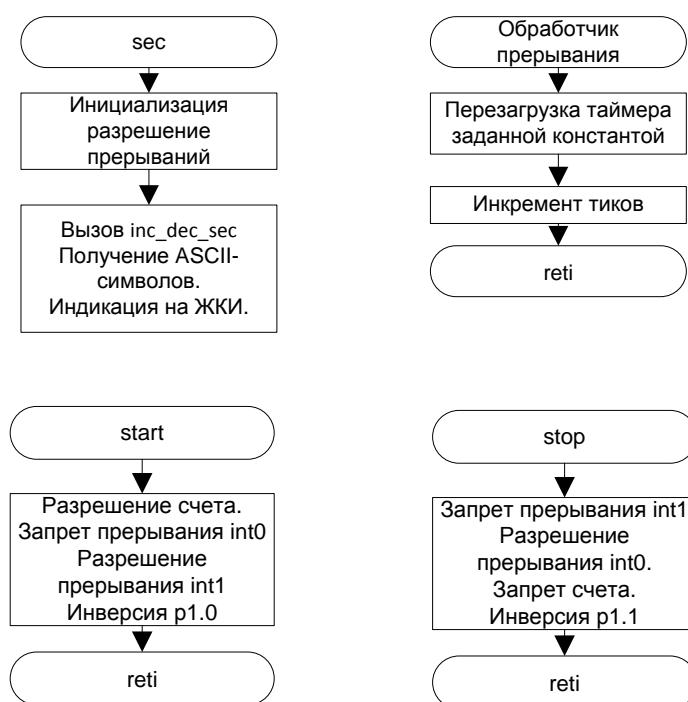


Рис. 16. Алгоритм программы «Электронный секундомер»

### Код программы:

```
org 8400h

ms:   equ 3Fh           ;миллисекунды
sd10: equ 40h           ;секунды / 10
sec:  equ 41h           ;секунды
min:  equ 42h           ;минуты
hours: equ 43h          ;часы
p4:   equ E8h

lcall init
```

```

loop: lcall main
      sjmp loop

; Инициализация
init: anl TMOD, #11110000b ;инициализация таймера
      orl TMOD, #00000001b ;для работы в режиме 16-битного счётчика

      mov TH0, #ECh ;инициализация счётчика T/C0 для
      mov TL0, #77h ;формирования "тика" 5 мс

      mov ms, #0h
      mov sd10, #0h
      mov sec, #0h
      mov min, #0h
      mov hours, #0h

      setb ea ;разрешение всех прерываний

      setb ex0 ;Разр внешн прерываний 0
      clr ex1 ;Разр внешн прерываний 1 (!)
      setb et0
      clr tr0
      mov p4, #0h ;загрузка нуля
      ret

main: cpl p1.3
      lcall inc_dec_sec
      lcall to_int
      lcall indic2
      ret

start: setb tr0
      clr ex0
      setb ex1
      cpl p1.0
      reti

stop: clr ex1
      setb ex0
      cpl p1.1
      clr tr0
      reti

tim0: inc ms ;инкремент тиков
      mov TH0, #FEh ;2KHz = 500mks
      mov TL0, #0Bh
      cpl p1.2

      ;lcall inc_dec_sec
      reti

; Инкрементируем мс, сек и мин
inc_dec_sec:
      mov a, ms
      ;cjne r5, #C8h, end ;если мкс=200*500
      ;mov ms, #0h

      clr c
      subb a, #200
      jc end

      mov ms, #0h

      inc sd10 ; инкремент 0,1 сек

```



```

mov r5, sd10
cjne r5, #Ah, 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
;lcall indic2
ret

include asms\4081_4\Dorofeev\5.3\sec\to_int.asm
include asms\4081_4\Dorofeev\5.3\sec\indic2.asm

org 8013h
ljmp stop

org 800bh
ljmp tim0

org 8003h
ljmp start

```

### ***Результаты программы:***

На экран выводится:

00h 00m 52s 3ms

Секундомер

При запуске программы электронный секундомер установлен в ноль и не запущен.

По нажатию клавиши крайнего левого столбца происходит запуск секундомера.

По нажатию клавиши центрального левого столбца происходит останов секундомера.

При повторном нажатии клавиши крайнего левого столбца происходит запуск секундомера с продолжением счёта.

## 4.6 Модифицированная программа «Электронные часы»

Для выполнения задачи дополним программу «Электронные часы» обработчиком прерывания от приёмопередатчика последовательного порта, который содержит команду вызова однократного сеанса связи с инструментальной ВМ.

Источниками запросов прерываний в модифицированной программе являются теперь прерывания от таймера T/C0, в обработчике которого формируются аппаратные тики часов, и прерывания последовательного порта. При наличии нескольких источников, запросы могут поступать асинхронно, в том числе когда уже обрабатывается один из запросов.

Для удобства наблюдения введем еще один счетчик – счетчик сотых долей секунды и на вкладке «Окна управления» оболочки Shell-51, проконтролируем его содержимое, которое передается на инструментальную ВМ при вызове подпрограммы по адресу 128h.

В качестве источника прерываний последовательного порта используется флаг R1, устанавливаемый при поступлении старт-импульса при запуске программы монитора кнопкой «Пуск» вкладки «Окна управления» оболочки Shell-51.

Исследуем влияние задаваемых уровней приоритетов запросов прерывания на точность формирования реального времени в данной системе.

### *Код программы:*

```
org 8400h

ms:    equ 3Fh ;миллисекунды
sd10:  equ 40h ;секунды / 10
sec:   equ 41h ;секунды
min:   equ 42h ;минуты
hours: equ 43h ;часы

lcall init

loop:  lcall inc_dec_sec
       sjmp loop

; Инициализация
init:  anl TMOD, #11110000b ;инициализация таймера
       orl TMOD, #00000001b ;для работы в режиме 16-битного счётчика

       mov TH0, #ECh ;инициализация счётчика T/C0 для
       mov TL0, #77h ;формирования "тика" 5 мс

       mov ms, #0h
       mov sd10, #0h
       mov sec, #0h
       mov min, #35h
       mov hours, #0Ch

       setb es ; разрешение прерываний приёмопередатчика
       mov A9h, #02h ; задание приоритета прерывания
       mov B9h, #02h

       setb ea ;разрешение всех прерываний
       setb et0 ;разрешение прерывания TC0
       setb tr0 ;разрешение счёта
```

```

        ret

tim0:   mov TH0, #FEh    ;2KHz = 500mks
        mov TL0, #0Bh
        cpl p1.2
        inc ms           ;инкремент тиков
        reti

; Инкрементируем мс, сек и мин
inc_dec_sec:
        mov a, sd10
        mov b, #0Ah
        mul ab
        mov r3, a

        mov r5, ms
        mov a, r5
        clr c
        subb a, #200
        jc end_tim

        mov ms, #0h
        inc sd10         ; инкремент 0,1 сек
        mov r4, sd10
        cjne r4, #Ah, 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
        lcall indic2

end_tim:      ret

usart:   jnb ri, skip
        push psw
        push a
        push 0
        lcall 128h
        pop 0
        pop a
        pop psw
skip:    reti

        org 800bh
        ljmp tim0
        org 8023h
        ljmp usart

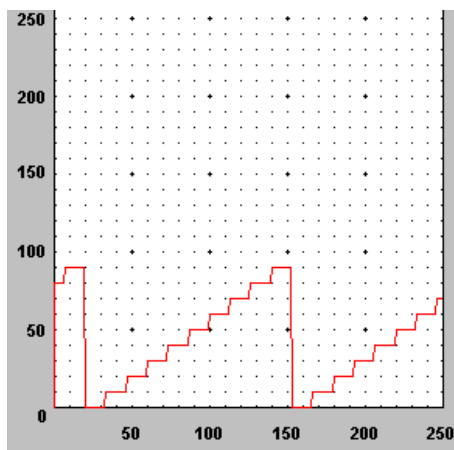
include asms\4081_4\Dorofeev\5.3\mclock\to_int.asm
include asms\4081_4\Dorofeev\5.3\mclock\indic2.asm

```

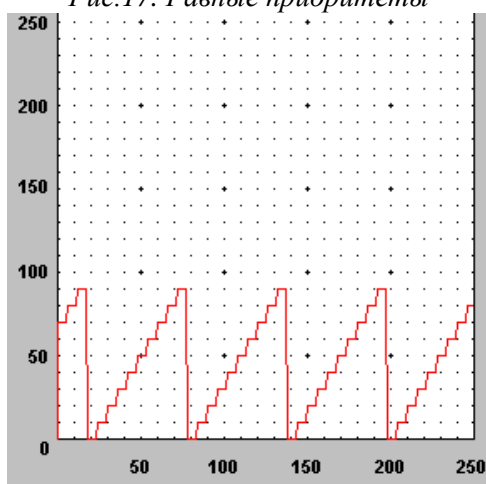
Результат работы программы:

Результаты можем наблюдать во вкладке «окна управления». На рисунках 17-19 изображены последовательно снятые значения счетчика сотых долей секунды для трех случаев. Первый – прерывания от таймера T/C0 и приемопередатчика последовательного порта имеют равный приоритет. Второй – прерывание от таймера имеет более высокий приоритет, чем прерывание от приемопередатчика последовательного порта. Третий – прерывание от таймера имеет более низкий приоритет, чем прерывание от приемопередатчика последовательного порта.

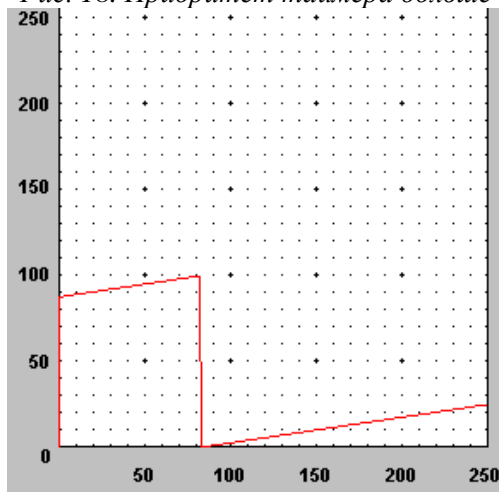
**Результаты:**



*Рис. 17. Равные приоритеты*



*Рис. 18. Приоритет таймера больше*



*Рис. 19. Прерывание от последовательного порта имеет высший приоритет*

Как видно из рисунков, в первом случае (равные приоритеты) счетчик считает немного медленнее, чем ожидалось. Это связано с тем, что при равных приоритетах прерывания от таймера не могут быть обработаны до того, как закончится обработка прерывания от последовательного порта, т.е. накопление «тиков» идет медленнее. Во втором случае часы работают корректно, т.к. прерывания от таймера имеют высший приоритет и обрабатываются сразу после возникновения, соответственно в третьем случае, часы абсолютно неработоспособны, так как прерываниям от приемопередатчика последовательного порта дан высший приоритет.

## 4.7 Многозадачная операционная система с разделением времени

### **Описание системы:**

Система выполняет диспетчеризацию трех взаимодействующих между собой циклически выполняемых программ: определение номера нажатой клавиши, программа «электронные часы», отображение на экране ЖКИ номера нажатой клавиши и текущего времени.

### **Принцип работы системы:**

Суть многозадачности заключается в организации работы «невидящих» друг друга псевдопараллельных процессов («задач»). В системе, поддерживающей многозадачный режим работы, задача выполняется в течение определённого интервала времени, по истечении которого её выполнение приостанавливается и управление передаётся очередной задаче. С целью исключения взаимного влияния процессов друг на друга информация о состоянии некоторых ресурсов процессора на момент переключения сохраняется. Эта информация называется контекстом или дескриптором задачи.

Квант времени выполнения задач задается таймером ТС/0. Чтобы имитировать одновременное, а не поочередное выполнение процессов, квант времени выполнения задач не должен быть слишком большим. В нашем случае квант времени, выделяемой каждой задаче = 5мс. По истечении этого времени происходит формирование импульса переполнения таймера и прерывание работы текущей задачи. При этом адрес очередной команды прерываемой задачи (адрес возврата) запоминается в стеке. Обработчик прерывания по переполнению таймера выполняет очень важную роль – роль диспетчера задач. При этом на него возлагаются следующие функции:

Перезагрузка счётчика таймера расчётной константой (задание кванта времени выполнения задачи).

- Определение адреса дескриптора текущей задачи.
- Сохранение контекста прерываемой задачи.
- Выбор очередной задачи на выполнение (определение номера вызываемой задачи и адреса её дескриптора). См Рис.13.
- Восстановление контекста вызываемой задачи
- Передача управления новой задаче по завершении программы обработчика командой `retl`.

#### **Выбор контекста задачи.**

Формат дескриптора задачи определяется объемом контекста, который зависит от локальных переменных задачи и их размещения в памяти. Учитывая, что запоминание контекста упрощается, если он занимает непрерывную область внутренней памяти, определим контекст как первые 32 байта внутренней памяти МК. Область внутренней памяти, начиная с адреса 20h, является общим ресурсом для всех задач. Она не перезаписывается при смене задач.

Так как задачи не модифицируют начальный указатель стека, а он при включении питания инициализируется значением 07h, то структура начальной области внутренней памяти данных задачи, выделенная по контекст будет выглядеть следующим образом:

Имя	R0	...	R7	...	...	PC <sub>L</sub>	PC <sub>H</sub>	Данные не определены	
Адрес	00		07	Стек задачи					1F

Поскольку указанная область памяти зарезервирована для размещения в ней контекста задачи, перед сохранением контекста во внешней памяти (при переключении задач) в нее необходимо внести также содержимое регистров SFR, входящих в контекст (DPTR,PSW,A,B). Для этого удобно использовать команду *push direct* записывающую содержимое ячейки по соответствующему адресу в стек.

Так как в качестве контекста мы выбрали непрерывную область памяти, то для сохранения и восстановления можно использовать циклические процедуры, исходный код которых следующий:

```
; Циклическая процедура сохранения контекста
. . . . .
mov R0,SP
    inc R0
    mov R1,#0
_save:
    mov A,@R1
    movx @DPTR, A
    inc R1
    inc DPTR
    djnz R0,_save
```

```
; Циклическая процедура восстановления контекста
. . . . .
mov R1,#0
_load:
    movx A,@DPTR
    mov @R1,A
    inc R1
    inc DPTR
    djnz R0,_load
```

Процедура сохранения контекста в своей работе использует регистры R0 и R1, следовательно, чтобы не потерять данные, хранящиеся в них и относящиеся к задаче, перед вызовом процедуры сохранения контекста, их значения также необходимо поместить в стек.

### ***Выбор очередной задачи на исполнение:***

Порядок выполнения задач в системе определяется последовательностью 0, 1, 2, 0, 1, 2 и т.д. В соответствии с этой последовательностью определяется номер задачи num\_tsk, которая будет выполняться следующей. Алгоритм выбора следующей задачи на исполнение очевиден: если номер выполняемой задачи меньше 2, то значение num\_tsk увеличивается на 1, иначе номеру присваивается значение 0. Реализация алгоритма представлена на рис.19.

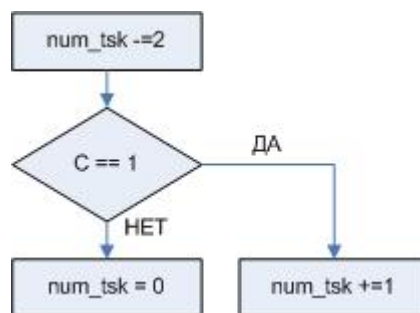


рис.19. Алгоритм выбора номера следующей задачи.

Алгоритм работы программы представлен на рис.20

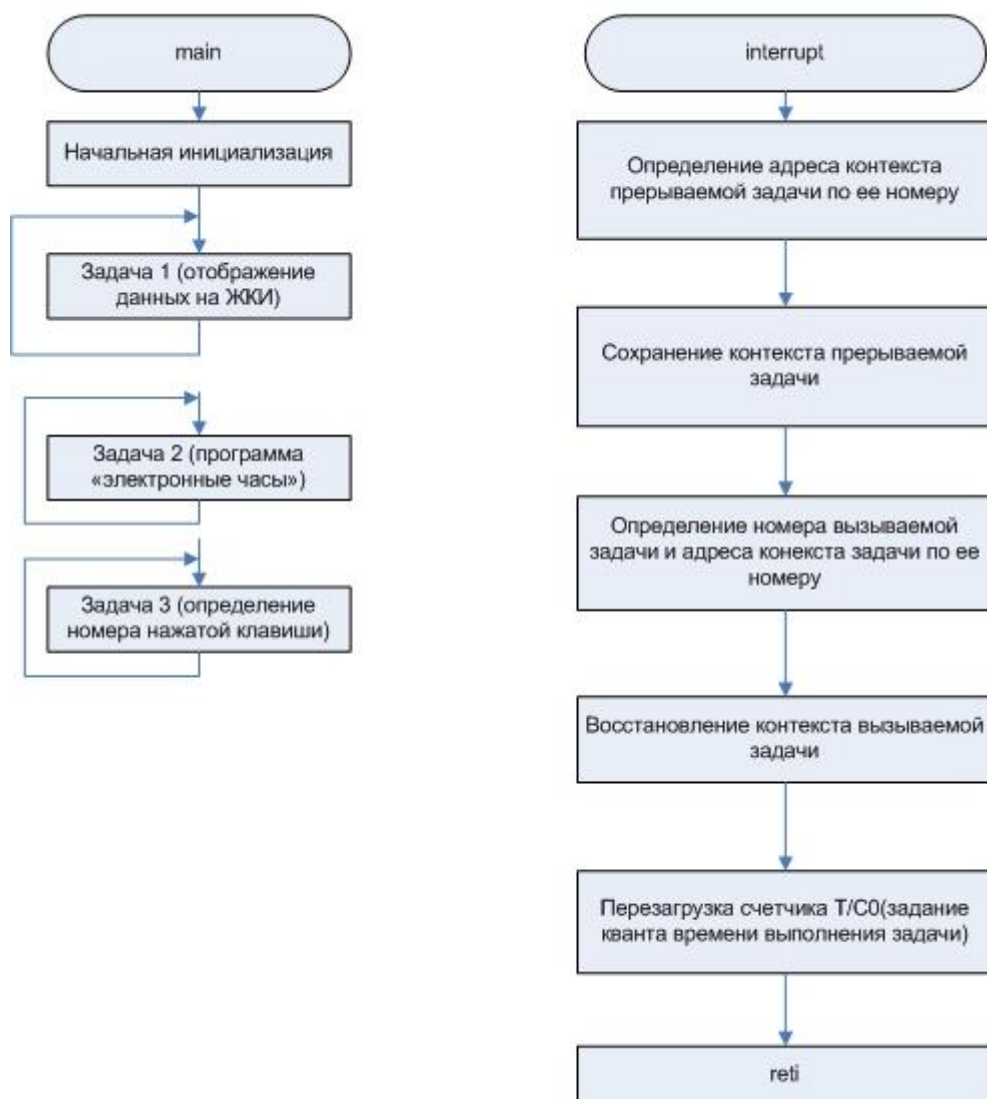


рис.20. Алгоритм работы программы.

### Код программы:

```

org 8400h

ljmp ini
include asms\klav.asm
include asms\define.asm
include asms\toasci.asm
include asms\indic.asm

num_task: equ 20h

ini:
    lcall init
progs:
    ljmp prog1

; инициализация таймера
init:
    anl TMOD, #11110000b
    orl TMOD, #00000001b
  
```



```

        mov TH0, #B8h                ; квант времени одной задачи - 10 мс
        mov TL0, #0Fh

        setb ea                      ;разрешение всех прерываний
        setb et0                    ;разрешение прерываний от таймера 0
        setb tr0                    ;разрешение счёта таймера 0

        ; задаём изначально num_task = 0 - номер задачи
        mov num_task, #0h
        ret

tim0:
        mov TH0, #FEh                ; 2KHz = 500mks
        mov TL0, #0Bh

; программа-диспетчер
dispatcher:

        ; сохранение SFR
        push dph
        push dpl
        push psw
        push b
        push a
        push 0
        push 1

        ; сохранение контекста
        mov r0, sp ;количество сохраняемых параметров
        inc r0
        lcall form_dpctr
        mov r1, #0h
prpm1:
        mov a, @r1
        movx @dpctr, a
        inc r1
        inc dpctr
        djnz r0, prpm1

        ; определение номера следующей задачи
        clr c
        mov a, num_task
        subb a, #2h
        jc inc_num_task
        mov num_task, #0h
        sjmp end_num_task
inc_num_task:    inc num_task
end_num_task:

        ; восстановление контекста
        lcall form_dpctr
        mov r1, #0h
prpm2:
        movx a, @dpctr
        mov @r1, a
        inc r1
        inc dpctr
        djnz r0, prpm2

        ; восстановление SFR
        dec r1
        mov sp, r1
pop 1
pop 0

```

```

    pop a
    pop b
    pop psw
    pop dpl
    pop dph

    reti

    ; определение DPTR
form_dpтр: mov dph, #E0h
    mov a, num_task
    mov b, #20h
    mul ab
    mov dpl, a
    ret

; задача1 - определение номера нажатой клавиши
    org 9000h
prog1:
    cpl P1.1
    lcall klav
    lcall klav1
    sjmp prog1

; задача2 - преобразование номера нажатой клавиши в
; ASCII код
    org 9200h
prog2:
    cpl P1.2
    lcall to_asci
    sjmp prog2

; задача3 - индикация номера нажатой клавиши
    org 9400h
prog3:
    cpl P1.3
    lcall indic
    sjmp prog3

; дескриптор задачи1
    org E000h
prog1_d: db 11h, 1, 0, 0, 0, 0, 0, 0, 00, 90h, 0, 0, 0, 0, 0, 0, 0, 0

; дескриптор задачи2
    org E020h
prog2_d: db 11h, 1, 0, 0, 0, 0, 0, 0, 00, 92h, 0, 0, 0, 0, 0, 0, 0, 0

; дескриптор задачи3
    org E040h
prog3_d: db 11h, 1, 0, 0, 0, 0, 0, 0, 00, 94h, 0, 0, 0, 0, 0, 0, 0, 0

    org 800bh
    ljmp tim0

; видеобуффер
    org FFD0h
str1: db 'Microcontrollers2011'
str2: db 65h, 20h, 20h, 20h, 20h, 42h, 65h, 70h, 61h, 20h, A4h, 79h, 65h,
B3h, 61h, 20h, 20h, 20h, 20h, 20h

```

**Результат работы программы:**

При запуске программы на экране ЖКИ отображается номер нажатой клавиши и текущее значение времени. Кроме того, так как в исходные коды задач были добавлены команды инвертирования соответствующего разряда порта, с помощью осциллографа можно наблюдать цикл работы каждой задачи. Время однократного выполнения каждой задачи представлено в таблице 3.

*Таблица 3*

Номер задачи	Время выполнения
1	900 мкс
2	400 мкс
3	240 мкс

## 5. Выводы

В ходе выполнения данной работы были изучены система таймеров МК SAB80C515, а также его система прерываний. Приведём выводы по каждой из них.

### **Система таймеров МК SAB80C515:**

MKSAB80C515 имеет в своём составе 3 таймера счётчика, которые могут быть запрограммированы на работу в нескольких различных режимах, что позволяет обеспечить решение широкого спектра задач.

В данной работе система таймеров МК применялась для формирования меандра требуемой частоты и ШИМ-сигнала фиксированной частоты и переменной скважности.

Для систем, работающих в реальном времени (например программы «Электронные часы» и «Секундомер»), крайне важной является точность функционирования системных часов, которые в свою очередь, имеют в основе таймер. Загружая таймер необходимой константой, мы можем задать частоту аппаратных «тиков» такой, какой нам необходимо. Логика программы заключается в подсчёте этих тиков с их дальнейшей обработкой.

Для повышения точности работы системы, имеющих в своём составе таймер, необходимо прежде всего учесть время выполнения самого обработчика прерывания от таймера, поскольку время его выполнения удлиняет формируемый «тик». Естественно, что чем меньше длительность выполнения программы обработчика, тем точнее часы. Поэтому «логику» проектируемой системы лучше всего реализовать в какой-либо фоновой программе, которая выполняется в бесконечном цикле. Обработчик, в свою очередь, лишь обновляет необходимые данные в памяти, откуда их обновлённое значение уже получает сама программа, исполняющаяся в фоновом режиме. Иными словами, обработчик и программа, отвечающая за логику проектируемой системы, должны иметь какое-либо разделяемое хранилище данных, в которое обработчик будет писать значения, а основная программа – читать оттуда.

Другим способом повышения работоспособности системы является учёт длины обработчика прерывания от таймера в основной программе. Например, в нашей программе, формирующей меандр требуемой частоты в зависимости от номера нажатой клавиши, можно было бы изменить константы перезагрузки таймера с учётом длины обработчика. Однако такой подход плох тем, что при изменении самого обработчика придётся перезагружать внешнюю память новыми значениями констант, которые предварительно необходимо будет перерасчитать.

### **Система прерываний МК SAB80C515:**

Система прерываний позволяет нашей программе узнать о наступлении какого-либо внутреннего или внешнего события. Приход этого события может быть обработан программой. Для этого нужно переопределить обработчик соответствующего события (прерывания).

Так как событий в системе может быть много и они могут носить совершенно разный характер, то можно сделать вывод, что не все события «равноправны». На одни система должна среагировать быстрее, чем на другие. Поэтому вводится система приоритетов между прерываниями. Соответственно, каждому событию может быть

присвоено какое-либо значение приоритета, в соответствии с которым оно будет обработано системой.

При расстановке приоритетов, необходимо продумать варианты прихода в систему событий и определить желаемую реакцию этой системы. Так, к примеру, пусть в систему может прийти только два события: одно внутреннее (сигнал переполнения таймера, на котором построены часы реального времени системы) и внешнее (к примеру сигнал о приёме данных на одном из портов). Прежде всего, нужно понимать, что если отдать наибольший приоритет событию поступления данных на порт МК, то системные часы «сбьются». Это произойдёт потому, что если прерывание от порта придёт раньше прерывания от таймера, а выполнение обработчика прерывания от порта ещё не завершилось, то прерывание от таймера не сможет прервать выполнение обработчика прерывания от порта, что приведёт к сбою системных часов и нарушению нормального функционирования систем в целом. Если же отдать приоритет таймеру, то система будет работать в соответствии со здравым смыслом. Часы будут работать без сбоев, так как их обработчик будет выполняться всегда своевременно, а обработчику прерывания от порта придётся «подождать», что, однако, будет совершенно неощутимо для пользователя системы.