Санкт-Петербургский государственный политехнический университет

Факультет технической кибернетики

Кафедра компьютерных систем и программных технологий

**Отчёт по лабораторной работе №3**

«Изучение таймеров и системы прерываний

по дисциплине "Микропроцессорные системы»

Вариант №5

Работу выполнил студент

Дорофеев Юрий Владимирович гр. 4081/12 \_\_\_\_\_\_\_\_\_\_\_\_\_\_

Работу принял преподаватель

Павловский Евгений Григорьевич \_\_\_\_\_\_\_\_\_\_\_\_\_\_

г. Санкт-Петербург

2012

**Содержание**

[1. Цель работы 3](#_Toc344422455)

[2. Теоретические сведения 3](#_Toc344422456)

[2.1. Блок таймеров/счетчиков 3](#_Toc344422457)

[2.2. Блок быстрого ввода-вывода МК SAB 80С515 6](#_Toc344422458)

[2.3. Система прерываний МК SAB80C515 9](#_Toc344422459)

[3. Программа работы 11](#_Toc344422460)

[4. Выполнение программ 12](#_Toc344422461)

[4.1. Формирование меандра 13](#_Toc344422462)

[4.2 Формирование ШИМ-сигнала 17](#_Toc344422463)

[4.3 Формирование ШИМ-сигнала с управлением с помощью инструментальной ЭВМ ………………………………………………………………………………………….20](#_Toc344422464)

[4.4 Электронные часы 22](#_Toc344422465)

[4.5 Электронный секундомер 27](#_Toc344422466)

[4.6 Модифицированная программа «Электронные часы» 30](#_Toc344422467)

[4.7 Многозадачная операционная система с разделением времени 34](#_Toc344422468)

[5. Выводы 40](#_Toc344422469)

1. Цель работы

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

1. Теоретические сведения
   1. Блок таймеров/счетчиков

Таймеры/счетчики предназначены для подсчета внешних событий, для формирования программно управляемых временных задержек и выполнения времязадающих функций в МК.

МК SAB 80С515 содержит в своем составе три программируемых 16-разрядных счетчика-таймера – Т/С0, Т/С1 и Т/С2. Таймеры Т/Сx реализуются на основе 16-разрядных суммирующих счетчиков со схемами управления. Программно-доступными регистрами являются регистры THx, TLx.

Собственно таймер/счетчик T/C реализуется с помощью 16-разрядных регистров блока SFR − TH0 и TL0 (T/C0), TH1 и TL1 (T/C12), TH2 и TL2 (T/L2). Таймеры/счетчики не идентичны.

Таймеры Т/С0 и Т/С1 могут использоваться в качестве программно управляемого таймера, генератора программируемой частоты, счетчика внешних событий. Таймер Т/С1 также используется для синхронизации работы приемопередатчика последовательного порта SP.

Таймер Т/С2 может функционировать как управляемый таймер или использоваться в качестве счетчика внешних событий. Но основная функция таймера 2 связано с реализацией на его основе функций быстрого ввода и функции быстрого вывода.

При работе в режиме таймера содержимое T/C увеличивается в каждом машинном цикле, т.е. через каждые 12 периодов тактовой частоты fCR со скоростью счета fCR/12. В этом режиме таймер можно использовать в качестве счетчика машинных циклов. При работе в качестве счетчика событий содержимое T/C инкрементируется при переходе сигнала из 1 в 0 на соответствующем внешнем выводе таймера/счетчика T0, T1 или T2. В этом режиме уровень сигнала на внешнем выходе TX опрашивается в фазе S5P2 каждого машинного цикла. При обнаружении высокого уровня сигнала в одном цикле и низкого уровня в следующем фиксируется наличие (поступление) внешнего события. Новое значение счетчика формируется в фазе S3P1 цикла, следующего за тем, в котором был обнаружен переход из 1 в 0. Так как на распознавание перехода требуется два машинных цикла, то максимальная частота счета входных событий равна fCR/24.

Для задания режимов работы таймера/счетчика T/Cx, где х = 0 или 1, и управлением T/Cx используются регистры блока SFR-регистр режимов TMOD и регистр управления TCON. Форматы регистров TMOD и TCON приведены на рис. 1. и рис. 2.

Регистр TMOD (адрес 89h) содержит набор программно управляемых бит (по четыре бита для каждого таймера), используемых для задания режима T/C.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Таймер 1** | | | | **Таймер 0** | | | | TMOD |
| GATE1 |  | M1 | M0 | GATE1 |  | M1 | M0 | адрес 89h |

GATE - бит управления блокировкой T/C

 − бит, определяющий функцию T/C: таймер () или счетчик ()

M1, M0 − биты, определяющие режим работы T/C.

|  |  |  |
| --- | --- | --- |
| M1 | M0 | Режим работы таймера |
| 0 | 0 | 8-битный Т/С на базе регистра THx (x=0, 1); регистр TLx используется как 5-разрядный предделитель |
| 0 | 1 | 16-битный Т/С. Регистры THx и TLx включены последовательно как один 16-битный регистр |
| 1 | 0 | автогенератор на основе 8-разрядного регистра TLx; регистр THx содержит значение, которое загружается в TLx каждый раз при переполнении TLx |
| 1 | 1 | 8-битный Т/С на базе регистра TL0, управляемый битами управления таймера 0, и 8-битный таймер на базе TH0, управляемый битами управления таймера 1; Т/С1 остановлен |

*Рис. 1. Формат и назначение отдельных бит регистра TMOD*

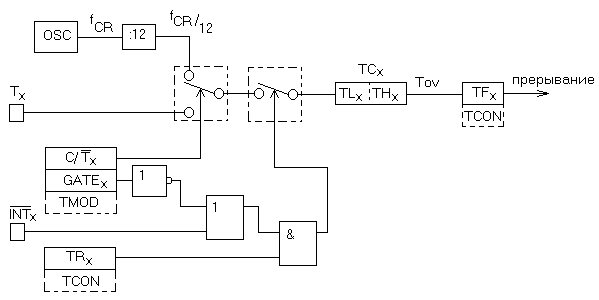
Регистр TCON для каждого таймера T/Cx (x = 0, 1) содержит бит управления разрешением работы TRx, с помощью которого при необходимости осуществляют пуск или останов таймера, и флаг переполнения TFx. Установка бита разрешения счёта TRx является необходимым, но недостаточным условием разрешения счёта. Счёт разрешен при TRx = 1, если отсутствует блокировка работы T/Cx (бит GATEx в регистре TMOD сброшен или отсутствует запрос прерывания на внешнем входе ). При невыполнении хотя бы одного из приведенных условий работа T/Cx блокируется, и счёт запрещается. Четыре младших бита регистра TCON к работе таймера отношения не имеют.

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

*Рис. 2. Формат регистра TCON*

Режимы работы 0, 1, 2 таймеров T/C0 и T/C1 одинаковы для обоих таймеров. В этих режимах Т/С0 и Т/С1 полностью независимы друг от друга. В режиме 3 работа Т/С0 и Т/С1 различна. Более того, перевод Т/С0 в режим 3 влияет на работу Т/С1 (установка режима 3 в Т/С0 блокирует (останавливает) работу Т/С1). Работа Т/С1 в режиме 3 не влияет на Т/С0.

*Режим 0*

В режиме 0 таймеры Т/С0 и Т/С1 являются 13-битными таймерами/счетчиками, состоящими из 5-разрядного предделителя на 32, реализованного на базе регистра TLx и 8-разрядного программируемого счетчика-делителя на основе регистра THx. Логику работы T/Cx в режиме 0 поясняет рис. 3.

*Рис. 3. Логика работы T/Cx в режимах 0 и 1*

В режиме таймера (соответствующий бит  в регистре TMOD сброшен) счетные импульсы fCR/12 через делитель на 32 поступают на вход программируемого 8-битного делителя. При fCR = 12 МГц счетчик-делитель THx увеличивает свое значение через каждые 32 мкс. Путем программной установки THx могут быть реализованы различные временные задержки в диапазоне от 32 мкс до 8192 мкс. Временные задержки, большие 8 мс, могут быть получены накопление переполнения TFx в каком-либо рабочем регистре под управление программы.

В режиме счетчика событий T/Cx увеличивает свое значение на 1, когда сигнал на входе Tx переходит из 1 в 0.

Работой T/Cx управляет схема управления, которая разрешает счет при установке бита TRx, если либо бит GATEx = 0, либо на входе  поступает сигнал высокого уровня. Установка бита GATEx позволяет сигналу на входе  управлять таймером и легко изменять ширину импульсов.

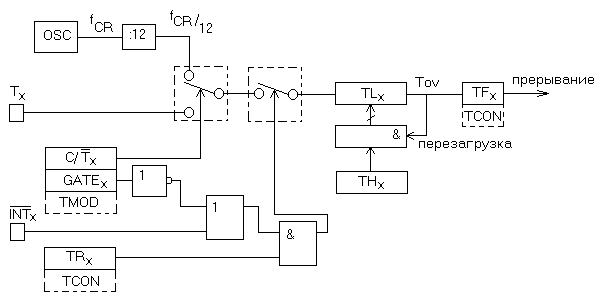
*Режим 1*

Режим 1 аналогичен режиму 0. Отличие состоит в том, что в этом режиме в качестве счетчика-делителя используется программируемый 16-битный счетчик, реализованный на основе последовательно включенных THx и TLx (рис.).

Путем программной установки THx и TLx могут быть реализованы временные задержки в диапазоне от 1 мкс до 65535 мкс.

*Режим 2*

В режиме 2 Т/Сх работает как автогенератор, реализованный на основе 8-разрядного регистра TLx. Логику работы ТСх в режиме 2 поясняет рис. 4.



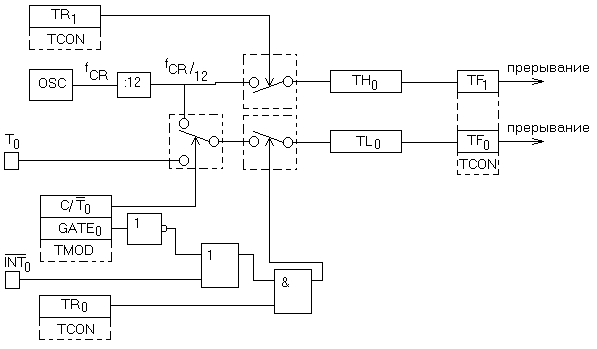
*Рис. 4. Логика работы Т/Сх в режиме 2*

В режиме 2 переполнение регистра TLx не только устанавливает флаг TFx, но и автоматически перезагружает регистр TLx значением из регистра THx, который предварительно инициирован программным путем. Перезагрузка TLx не изменяет содержимое регистра THx.

*Режим 3*

В режиме 3 Т/С0 и Т/С1 работают по-разному.

Таймер Т/С0 в режиме 3 функционирует как два независимых 8-битных счетчика TH0 и TL0, при этом счетчик TH0 может выполнять только функцию таймера. Его управление осуществляет управляющий бит TR1, а переполнение TH0 фиксируется во флаге TF1. Логику работы T/C0 в режиме 3 поясняет рис. 5.

*Рис. 5. Логика работы Т/С0 в режиме 3*

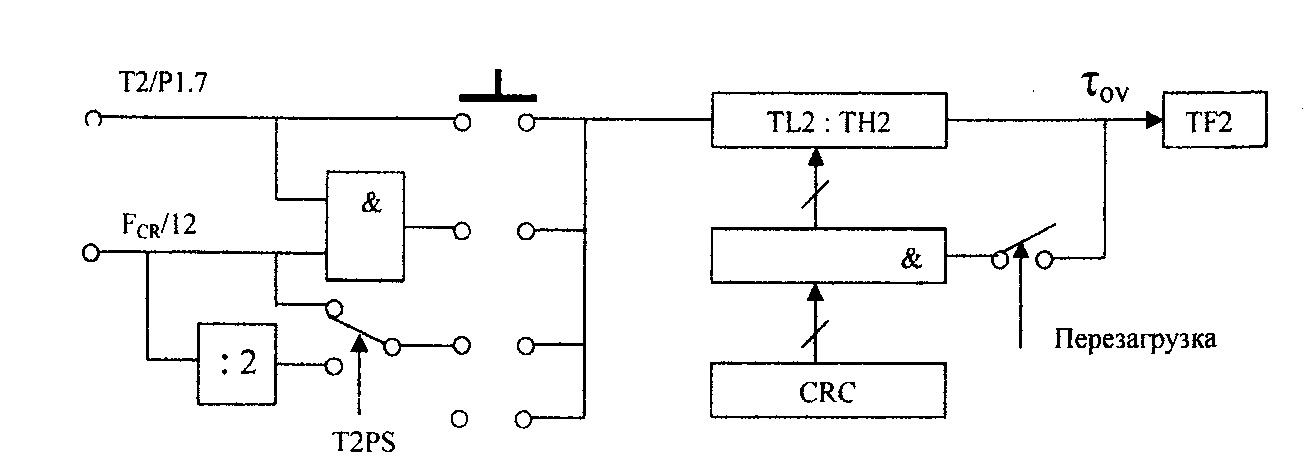
В режиме 3 на базе регистра TL0 реализуется обычный 8-разрядный счётчик-таймер, для управления которым используются биты управления TR0, GATE и .

В случае, когда T/C0 установлен в режим 3, Т/С1 может быть использован в любом применении, не требующем прерывания. В частности, при установленном режиме 2 Т/С1 может использоваться в качестве генератора частоты передачи последовательного порта SP. Можно считать, что при переключении Т/С0 в режим 3, МК 80C515 имеет в своем составе три таймера.

При переключении в режим 3 Т/С1 сохраняет неизменным свое содержимое, т.е. достигается тот же эффект, что и при сбросе управляющего бита TR1.

* 1. Блок быстрого ввода-вывода МК SAB 80С515

16-разрядный таймер-счетчик Т/С2 (рис. 6), наряду с реализацией функций счета внешней 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. 6. Функциональная схема таймера Т/С2*

Режим работы таймера Т/С2 задается программно с помощью управляющих бит регистра T2CON блока SFR (рис. 7).

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| CFh | CEh | CDh | CCh | CBh | CAh | C9h | C8h | T2CON |
| T2PS | I3FR | I2FR | T2R1 | T2R0 | T2CM | T2I1 | T2I0 | адрес C8h |

*Рис. 7. Формат регистра T2CON*

Биты T2I1, T2I0 задают режим работы таймера Т/С2 и определяют источник входных счетных импульсов (табл. 1).

*Таблица 1. Режимы работы таймера Т/С2*

|  |  |  |
| --- | --- | --- |
| **T2I1** | **T2I0** | **Режим работы таймера 2** |
| 0 | 0 | Таймер 2 остановлен. Счетные импульсы на вход не поступают. |
| 0 | 1 | Режим неуправляемого таймера. На вход таймера поступают импульсы внутреннего источника с частотой fCR/12 (T2PS = 0) или fCR/24 (T2PS = 1) |
| 1 | 0 | Режим управляемого таймера. На вход таймера поступают счетные импульсы с частотой fCR/12 (T2PS = 0) или fCR/24 (T2PS = 1) |
| 1 | 1 | Режим счетчика внешних событий, поступающих на вход T2/P1.7 |

Биты T2R1, T2R0 назначают режим перезагрузки регистров TH2:TL2 таймера T/C2 (табл. 2).

*Таблица 2. Режимы перезагрузки таймера Т/С2*

|  |  |  |
| --- | --- | --- |
| **T2R1** | **T2R0** | **Режим перезагрузки** |
| 0 | х | Перезагрузка таймера 2 запрещена |
| 1 | 0 | Режим 0: автоматическая перезагрузка импульсами переполнения таймера 2 |
| 1 | 1 | Режим 1: перезагрузка таймера 2 по сигналу перехода из 1 в 0 на внешнем входе T2EX/P1.5 |

Бит T2CM назначает один из двух режимов (0 или 1) сравнения блока HSIO.

Бит T2PS - бит разрешения работы делителя на 2 частоты внутреннего источника счетных импульсов таймера 2.

Структурная схема HSIO имеет вид, показанный на рис. 8.



*Рис. 8. Структурная схема блока HSIO*

В качестве входов/выходов блока HSIO используются многофункциональные выходы порта P1:

P1.0/ INT3/CRC - вход внешнего прерывания 3 / выход 0 HSO / вход 0 HSI

P1.1/INT4/CC1 - вход внешнего прерывания 4 / выход 1 HSO / вход 1 HSI

P1.2/INT5/CC2 - вход внешнего прерывания 5 / выход 2 HSO / вход 2 HSI

P1.3/INT6/CC3 - вход внешнего прерывания 6 / выход 3 HSO / вход 3 HSI

P1.4/ - вход внешнего прерывания 2

P1.5/T2EX - вход внешней перезагрузки таймера 2

P1.6/CLKOUT - вход системного тактового генератора. Сигнал на указанном входе не влияет на функционирование блока HSIO

P1.7/T2 - внешний сигнал таймера 2 в режиме счетчика внешних событий или вход внешнего управления таймера 2.

Функционирование блока HSIO поддерживает несколько регистров блока SFR: TH2/TL2 (16-разрядный регистр таймера 2), T2CON (регистр управления таймера 2), CCEN (регистр разрешения сравнения/захвата); CCH-1,-2,-3/CCL-1,-2,-3 (16-разрядный регистр сравнения/ захвата 1, 2, 3); CRCH/CRCL (16-разрядный регистр сравнения/перезагрузки/ захвата), IRCON (регистр управления прерываниями).

Назначение режима, в котором используются регистры блока HSIO, кодируется управляющими битами регистра CCEN (по два бита для указания режима использования каждого регистра) (табл. 3).

*Таблица 3. Режимы использования регистра ССх*

|  |  |  |
| --- | --- | --- |
| **bit 2i** | **bit 2i+1** | **Режим использования регистра CCx** |
| 0 | 0 | Захват/сравнение запрещены |
| 0 | 1 | Захват внешнего события по (быстрый ввод) |
| 1 | 0 | Сравнение разрешено (быстрый вывод или ШИМ) |
| 1 | 1 | Режим чтения "на лету" (запись в регистр CCx) |

Устройство HSIO может работать в 2-х режимах: захвата и сравнения.

### *Режим сравнения*

В режиме сравнения 16-разрядное значение, загруженное в любой регистр CC1 - CC3, CRC сравнивается с текущим содержимым регистра TH, TL таймера 2. В момент равенства этих значений генерируется внутренний сигнал сравнения, который может использоваться для формирования выходного события (сигнал заданного вида) на соответствующем выводе P1.0 - P1.3. порта P1. Структура одного разряда канала показана на рис. 9.

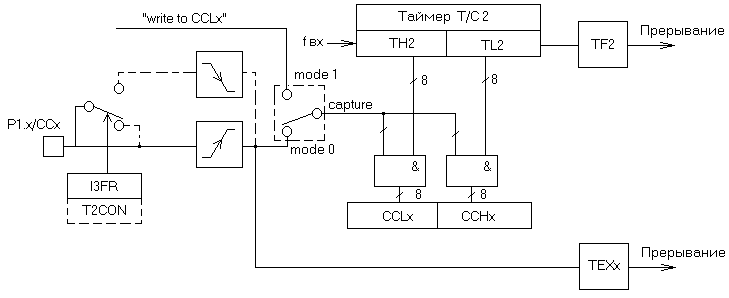


*Рис. 9. Структура одного разряда канала блока HSIO при работе в режиме сравнения*

Режим сравнения 0 является режимом формирования ШИМ-сигналов, а режим сравнения 1 - режимом формирования внешних событий заданного вида.

### *Режим захвата*

Структура одного канала блока HSIO в режиме захвата представлена на рис. 10.



*Рис. 10. Структура одного канала блока HSIO в режиме захвата*

Захват события – это время его наступления. В режиме быстрого ввода в каналах захвата с регистрами CC1 - CC3, захват внешнего события происходит при изменении входного сигнала на выводе P1.1/CC1 - P1.3/CC3 только при переходе из 0 в 1, в канале с регистром CRC (CC0) захват внешнего события может происходить при изменении входного сигнала на выводе P1.0/CC0 как из 0 в 1, так и из 1 в 0.

Режим захвата 0 - на соответствующий вход канала захвата (Р1.0/СС0 - Р1.3/СС3) формируется внутренний сигнал capture, обеспечивающий перепись текущего содержимого таймера 2 в соответствующий регистр CCx (x = 0, 1, 2, 3), а также устанавливается соответствующий флаг прерывания IEXx и если прерывания разрешены, может быть вызвана подпрограмма обслуживания прерывания.

Режим захвата 1 - захват инициируется программно при выполнении команды пересылки какого-либо значения в регистр CCLx. В этом случае в качестве внутреннего сигнала capture при перезаписи содержимого счетчика таймера 2 в регистр CCx используется внутренний сигнал "write to CCLx", формируемый при выполнении указанной команды. Здесь фактически происходит программное чтение содержимого таймера 2 без остановки счета (используя иные методы чтения значений T2H и T2L можно считать ошибочные значения, т.к. таймер изменит свое значение между выполнением 2-х команд). При инициализации захвата флаг IEXx не устанавливается и запрос прерывания не формируется.

* 1. Система прерываний МК SAB80C515

Микроконтроллер SAB 80C515 обладает гибкой системой прерываний, обеспечивающей реакцию на запросы, которые могут поступать от внешних или внутренних периферийных устройств.

Система прерываний микроконтроллера SAB 80C515 является многовекторной и многоуровневой. Она имеет 6 внутренних и 8 внешних источников запросов, которые с помощью логических схем микроконтроллера преобразуются в 12 векторов прерывания, однозначно задающих адреса программ обслуживания прерывания. Очередность обслуживания прерывания определяется механизмом 4-уровневой системы приоритетного прерывания.

Все источники прерывания МК объединены попарно, образуя 6 пар.

К каждой паре можно источников прерывания можно программно можно определить один из 4-х уровней обслуживания (приоритета). В пределах одного уровня обслуживания приоритет обслуживания определяется зарезервированным местом в последовательно опрашиваемой цепочке поступивших запросов. Внутри каждой пары источник прерывания, указанный слева, имеет более высокий приоритет. В пределах одного уровня приоритеты пар фиксированы. Первой пары – наивысший. Шестой – самый низкий.

*Таблица 4. Пары источников прерываний и приоритеты прерываний внутри уровня.*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 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 |
| 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 | T1 или R1 | 0023h | Внешнее прерывание 5 | IE5 | 0063h | 4 |
| Переполнение таймера 2 или внешняя перезагрузка | TF2  EXF2 | 002Bh | Внешнее прерывание 6 | IE6 | 006Bh | 5 |

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

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| IP0 | Адрес А9h | - | WDTS | IP0.5 | IP0.4 | IP0.3 | IP0.2 | IP0.1 | IP0.0 |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| IP1 | Адрес B9h | - | - | IP1.5 | IP1.4 | IP1.3 | IP1.2 | IP1.1 | IP1.0 |

*Таблица 5. Форматы регистров IP0 и IP1.*

*Внешние прерывания*

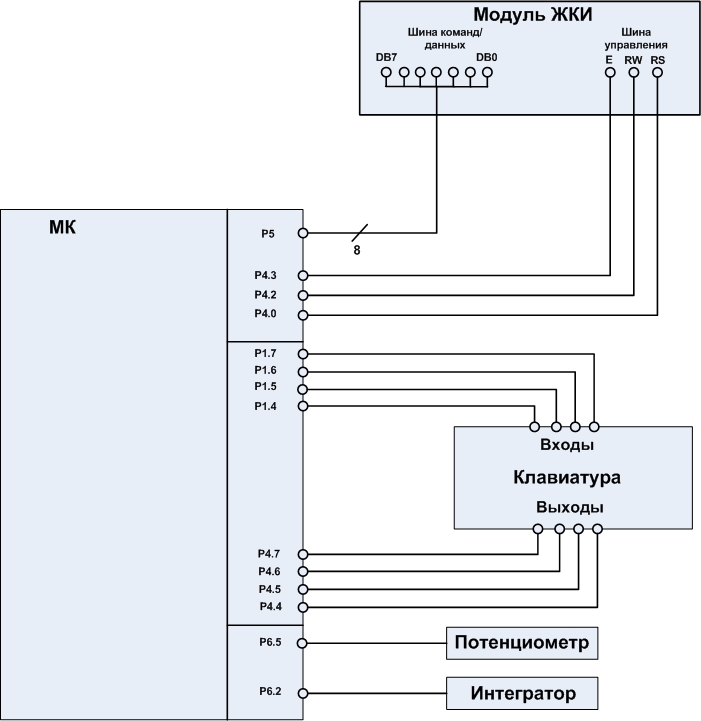
Внешние прерывания могут активизироваться при поступлении сигнала заданного вида (уровня или перехода от одного уровня к другому) на соответствующий вход запроса прерывания INTx. При идентификации (обнаружении) прерывания устанавливается флаг запроса прерывания и, если прерывания разрешены, вызывает программу обработки прерывания.

Все выводы внешних прерываний опрашиваются один раз в каждом машинном цикле. Если внешнее прерывание вызывается уровнем, для его распознавания необходимо, чтобы внешний источник удерживал активный уровень, по крайней мере, один машинный цикл. Флаг, фиксирующий запрос прерывания по уровню, просто повторяет (не запоминает) входной сигнал запроса, поэтому для обработки такого прерывания необходимо, чтобы запрос удерживался до вызова программы обработки прерывания, при этом для предотвращения повторного вызова программы обработки прерывания сигнал запроса должен сбрасываться до завершения программы обработки прерывания. В общем случае флаг запроса прерывания по уровню установлен при активном входе запроса прерывания и сброшен при деактивизации входа запроса.

Если внешнее прерывание активизируется перепадом, для его идентификации требуется, по крайней мере, два машинных цикла, при этом соответствующий флаг прерывания будет установлен (зафиксирован в триггере флага), если две последовательные проверки входа внешнего прерывания покажут в одном машинном цикле сигнал исходного уровня, а в следующем − сигнал уровня перехода. Сброс флага прерывания по переходу выполняется аппаратно (автоматически) при вызове программ обработки прерывания.

1. Программа работы
2. Изучить работу таймеров-счетчиков микроконтроллера и принципы организации системы прерываний.
3. Разработать программу, генерирующую на выбранном разряде порта МК меандр с частотой Nклав\*100 Гц. Работу с линией порта проводить в обработчике прерываний, генерируемых таймером Т/С0. Для управления генерируемой частотой использовать клавиатуру.
4. Разработать программу формирования ШИМ-сигнала с частотой 1000 Гц и регулируемой скважностью от АЦП.
5. Модифицировать программу п.3 так, чтобы управляющий код задавался через «Окна управления».
6. Реализовать на микроконтроллере счетчик времени с отображением на ЖКИ текущего времени с точностью 0,1 с. В качестве опорного программного генератора использовать прерывания таймера.
7. Разработать программу, определяющую интервал времени между прерываниями INT0 и INT1, генерируемых при нажатии двух клавиш разных столбцов клавиатуры стенда.
8. Модифицировать программу п.5 так, чтобы с помощью «окон управления» выводились доли секунды. Использовать разные приоритеты запроса прерываний.
9. Разработать программу управления простейшей многозадачной системой с разделением времени.
10. 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

Разработать и выполнить программу, генерирующую на заданном разряде порта МК импульсный сигнал прямоугольной формы (меандр) с частотой, определяемой номером Nклав нажатой клавиши блока клавиатуры (F = N \* 100 Гц).

Меандр получается в результате инверсии наблюдаемого разряда порта 1. Поэтому для формирования меандра заданной частоты импульсы переполнения таймера 0 должны следовать с вдвое большей частотой. Сама частота определяется номером нажатой клавиши. Поскольку счётчик таймера работает на сложение, для формирования «тика» требуемой длительности в него необходимо загружать константу Nзагр = (216-1) – N, где N – расчетное значение длительности.

В начале работы программы требуется разрешить все прерывания, прерывание от таймера 0 и его счёт. Кроме того, необходимо заменить стандартный обработчик прерывания таймера 0 на собственный, в котором реализовать инверсию на выходе порта 1.2, определить константу загрузки и направить её в соответствующие регистры таймера 0.

Схема соединений приведена на рис.11. На рис. 12 и рис. 13 изображены схемы основной программы и обработчика прерывания таймера 0.

МК

P1.2

Клавиатура

P5 [7..4]

Входы [3..0]

Выходы [3..0]

P5[3..0]

Осциллограф

Рис.11. Схема соединения МК 80C515 и периферии.

**Схема программы формирования меандра**

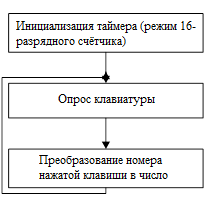
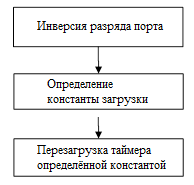
 

Рис. 12. Основная программа. Рис.13 Обработчик прерывания.

***Код программы:***

Файл meandr.asm

org 8300h

TH0: equ 8Ch ; старший байт загрузки таймера 0

TL0: equ 8Ah ; младший байт загрузки таймера 0

TMOD: equ 89h ; регистра управления таймера 0

lcall init ; подпрограмма инициализации таймера 0

loop: ; бесконечный цикл

lcall klav ; подпрограмма опроса клавиатуры

lcall klav1 ; подпрограмма преобразования № клавиши в код

sjmp loop ; зациклить программу

ret

init: ; инициализация таймера 0

anl TMOD, #11110000b ; сбросить настройки

orl TMOD, #00000001b ; установить в режим 16-разрядного таймера

mov TH0, #ECh ; в начале программы частота 100Гц

mov TL0, #78h

setb ea ; разрешить все прерывания

setb et0 ; разрешить прерывание таймера 0

setb tr0 ; разрешить счёт

ret

org 800Bh

lcall tim0

tim0: ; обработчик прерывания таймера 0

cpl P1.2 ; инверсия 2го разряда 1го порта

mov a, 34h ; определение константы загрузки таймера

rlc a

dec a

mov r0, a

mov a, @r0

mov TL0, a

dec r0

mov a, @r0

mov TH0, a

reti

org A0h

freqn0: db ECh ;100 Hz

db 77h

freqn1: db F6h ;200 Hz

db 3Ch

freqn2: db F9h ;300 Hz

db 7Dh

freqn3: db FBh ;400 Hz

db 1Eh

freqn4: db FCh ;500 Hz

db 18h

freqn5: db FCh ;600 Hz

db BEh

freqn6: db FCh ;700 Hz

db 35h

freqn7: db FDh ;+00 Hz

db 8Fh

freqn8: db FDh ;900 Hz

db D4h

freqn9: db FEh ;1000 Hz

db 0Ch

freqnA: db FEh ;1100 Hz

db 39h

freqnB: db FEh ;1200 Hz

db 5Fh

freqnC: db FEh ;1300 Hz

db 7Fh

freqnD: db FEh ;1400 Hz

db 9Ah

freqnE: db FEh ;1500 Hz

db B2h

freqnF: db FEh ;1600 Hz

db C7h

include C:\SHELL51\asms\4081\_3\code\klav1.asm

include C:\SHELL51\asms\4081\_3\code\klav.asm

Файл klav.asm

klav:

p4: equ 0E8h ;Работаем с портом 4

mov r0,#30h ;Ук-ль на начало карты (30h – адрес карты)

orl p4,#f0h ;Настройка порта р4 на ввод

mov a,#f7h ;Загрузка бегущего нуля

;Цикл сканирования

m1: mov p4,a ;Записываем в p4 сканирующий код

mov r2,a ;Сохраняем состояние текущей строки

mov a,p4 ;Чтение состояния клавиатуры

anl a,#f0h

mov @r0,a ;Заполняем ячейку карты памяти

inc r0 ;Переходим к следующему адресу

mov a,r2

rr a ;Сдвигаем сканирующий ноль

cjne a,#7fh,m1

ret

Файл klav1.asm

klav1:

mov r0, #2Fh ; Адрес карты памяти

mov r1, #34h ; Адрес результата

mov a, #0 ; Код нажатой клавиши в аккумуляторе

clr 0 ; Нажата ли хотя бы одна кнопка

clr 1 ; Нажато более одной кнопки

start:

inc r0

lcall check ; Анализ ячейки памяти

jb 1, finish ; Нажато более одной кнопки - конец

cjne r0, #33h, start

jb 0, finish ; Была ли нажата хотя бы одна клавиша?

mov @r1, #00h

finish: ret

check:

mov 02h, @r0 ; Запись в r2 текущего значения из карты памяти

cjne r2, #F0h, n0 ; Была ли нажата клавиша?

add a, #4 ; Если нет, то переходим к следующему столбцу

ret

n0: ; Проверка нажатия первой кнопки

inc a

cjne r2, #70h, n1 ; Первая кнопка нажата?

jb 0, n4 ; Если до этого уже была нажата клавиша, то

mov @r1, a ; устанавливаем bit 1 и выходим

setb 0

ret

n1: ; Проверка нажатия второй кнопки

inc a

cjne r2, #B0h, n2

jb 0, n4

mov @r1, a

setb 0

ret

n2: ; Проверка нажатия третьей кнопки

inc a

cjne r2, #D0h, n3

jb 0, n4

mov @r1, a

setb 0

ret

n3: ; Проверка нажатия четвертой кнопки

inc a

cjne r2, #E0h, n4

jb 0, n4

mov @r1, a

setb 0

ret

n4: ; Переход сюда, если выяснилось, что было нажато более одной кнопки

mov @r1, #FFh

setb 1

ret

***Результаты выполнения:***

Проверка работоспособности программы заключалась в запуске программы и проверке вывода на экран осциллографа меандров различной частоты.

Результат испытаний программы приведён в табл. 6.

Таблица 2. Расчётные и реальные частоты меандра

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № нажатой клавиши | Расчётное значение частоты | Nзагр | Hex | Значение частоты, измеренное осциллографом | Погрешность |
| 0 | 100 | 60535 | EC77 | 100 | 0 |
| 1 | 200 | 63035 | F63B | 193 | 7 |
| 2 | 300 | 63870 | F982 | 291 | 9 |
| 3 | 400 | 64349 | FB5F | 389 | 11 |
| 4 | 500 | 64567 | FC37 | 500 | 0 |
| 5 | 600 | 64734 | FCDE | 588 | 12 |
| 6 | 700 | 64853 | FD55 | 694 | 6 |
| 7 | 800 | 64938 | FDAA | 806 | 6 |
| 8 | 900 | 64987 | FDE0 | 893 | 7 |
| 9 | 1000 | 65046 | FE16 | 1000 | 0 |
| A | 1100 | 65088 | FE40 | 1099 | 1 |
| B | 1200 | 65123 | FE63 | 1190 | 10 |
| C | 1300 | 65156 | FE84 | 1292 | 8 |
| D | 1400 | 65184 | FEA0 | 1396 | 4 |
| E | 1500 | 65206 | FEB6 | 1489 | 11 |
| F | 1600 | 65226 | FEAC | 1592 | 8 |

Наличие встроенных таймеров позволяет с минимальным участием микропроцессорного устройства контроллера генерировать импульсы заданной длительности с достаточно высокой точностью. Повысить точность формирования импульсов требуемой частоты можно, проведя коррекцию констант загрузки таймера, связанную с числом машинных циклов, затрачиваемых на обработку прерывания от него. Такая коррекция проводилась в случаях большого отклонения полученной частоты от требуемой.

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 Fpwm определяется константой перезагрузки Nзагр.

Так как скважность имеет переменные значения, то и Nзагр будет иметь переменные значения Nзагрi. Каждое такое значение вычисляется по формуле.

Nзагрi = Nзагр0 + Nti, где Nti – масштабированное значение управляющего кода.

Вычислим коэффициент масштабирования Км.

Kmint = 3 – старший байт, который определяет целую часть

Kmfract = 9 – младший байт, который определяет дробную часть

Теперь мы можем определить промасшатбированное значение управляющего кода:

1. Nti = i \* Kм, где i – число дискрет управляющего кода (i = 0-255).

***Пояснения к программе:***

Скважность ШИМ-сигналов в данной работе регулируется при помощи ручки потенциометра.

Сначала необходимо получить значение с выхода АЦП.

Затем рассчитывается Nti – промасштабированное значение управляющего кода.

По формуле (1) происходит расчёт Nti. 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 ; Nзагр0

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 Nзагр­I 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 Т/С2 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

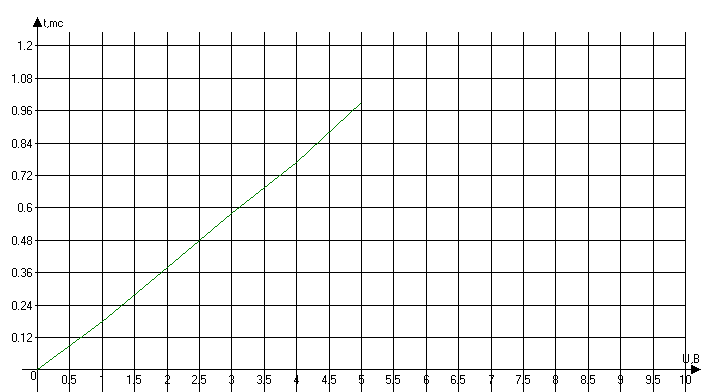
ret

***Результаты выполнения:***

Таблица 3.Результат выполнения программы

|  |  |
| --- | --- |
| Uпот, В | τpwm, 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.



*Рис.12. Зависимость t = f(U)*

На экране осциллографа были отображены сигналы, с изменяющейся в зависимости от подаваемого на вход АЦП напряжения скважностью (ШИМ-сигналы).

Блок быстрого ввода-вывода HSIO позволяет формировать требуемые выходные сигналы без постоянного участия микроконтроллера. Ему не требуется прерывать текущую задачу, а потому реакция на события осуществляется более быстро. Кроме того, блок HSIO осуществляет захват события - безопасное чтение регистровой пары TH/TL без остановки таймера. Как захват, так и сравнение выполняются внутренними 16-разрядными схемами HSIO, а потому увеличивается точность формирования сигнала захвата/сравнения (не тратится время на вызов обработчика прерывания и возврат из него).

4.3 Формирование ШИМ-сигнала с управлением с помощью инструментальной ЭВМ

Модифицировать программу предыдущего задания, используя для управления скважностью ШИМ-сигнала цифровые коды управляющего воздействия, формируемого инструментальной ЭВМ при работе с вкладкой «Окна управления» (Опция «вход»).

Для выполнения данного задания необходимо в циклической процедуре формирования ШИМ-сигнала должна размещаться команда lcall 128h, которая реализует переход на подпрограмму однократного сеанса связи с инструментальной ЭВМ. Также необходимо не выполнять теперь аналого-цифровое преобразование, так как входной сигнал задаётся из инструментальной ЭВМ.

Текст программы не изменился, за исключением добавления в неё команды перехода на подпрограмму однократного сеанса связи с инструментальной ЭВМ (lcall 128h) и отключения процедуры аналого-цифрового преобразования.

Файл pwm.asm (жирным выделены изменения)

…

lcall pwm ; процедура формирования ШИМ-сигнала

loop: ; бесконечный цикл

**; lcall adc ; процедура аналого-цифрового преобразования**

mov a, U ; цифровое значение напряжения -> аккумулятор

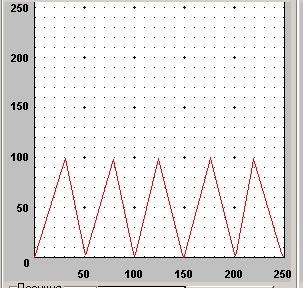
lcall count ; процедура преобр-я напр-я в длит-ть сигнала

**lcall 128h ;однократный сеанс связи с инструментальной ЭВМ**

sjmp loop ; зациклить программу

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.

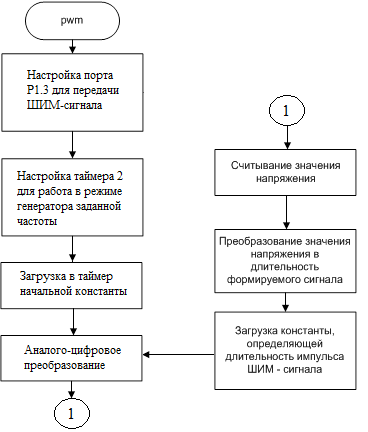
При достижении счётчика секунд до значения 60 происходит его сброс и инкремент счётчика минут.

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 TС0

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 p1.7

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

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, генерируемыми при нажатии двух клавиш разных столбцов клавиатуры стенда.

***Описание программы:***

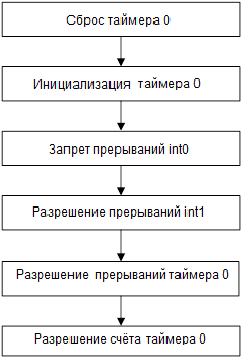
Алгоритм работы электронного секундомера основан на работе электронных часов.

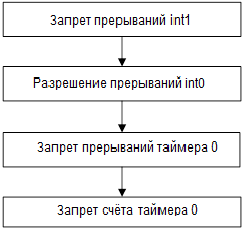
Добавлены обработчики прерываний int0 и int1.

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 ;милисекунды

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:***

При нажатии любой кнопки первого столбца формируется прерывание int0, о чём свидетельствует запуск секундомера (на ЖКИ начинается счёт времени, начиная с 00:00:00.0). Повторные нажатия на кнопки этого столбца не приводят ни к какой реакции. Нажатие на любую кнопку четвёртого столбца приводит к формированию прерывания int1, о чём свидетельствует остановка счёта секундомера. Повторные нажатия на кнопки этого столбца не приводят ни к какой реакции. Если опять нажать на любую кнопку первого столбца, то секундомер продолжит счёт с времени, на котором он остановился по нажатию кнопки четвёртого столбца.

На примере решения данной задачи были рассмотрены формирование и обработка внешних прерываний. Одному устройству соответствует свой вход прерывания, чего оказывается вполне достаточно для решения поставленной задачи.

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 теперь прерывания от таймера Т/С0, 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 ;милисекунды

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 TС0

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

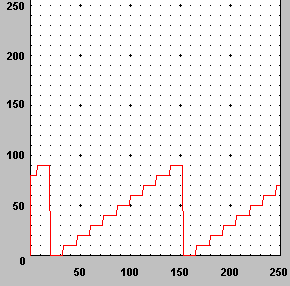
include asms\4081\_4\Dorofeev\5.3\mclock\to\_int.asm

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

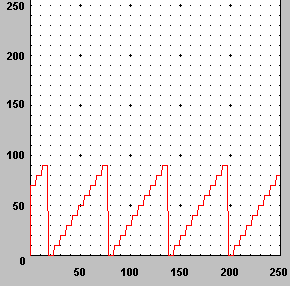
Результат работы программы:

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.

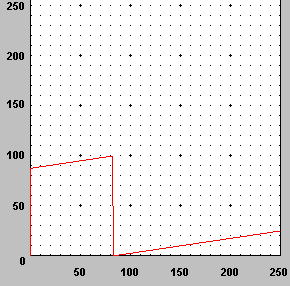
***Результаты:***



*Рис.17. Равные приоритеты*



*Рис. 18. Приоритет таймера больше*



*Рис. 18. 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:***

Cуть 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 TC/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 истечению 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 reti.

***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 | … | … | PCL | PCH | 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

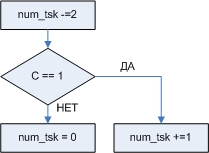
inc DPTR

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.

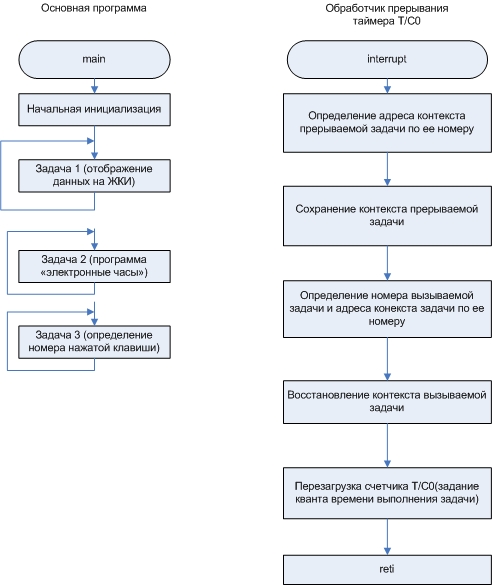
***Выбор очередной задачи на исполнение:***

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, #В8h ; 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\_dptr

mov r1, #0h

prpm1:

mov a, @r1

movx @dptr, a

inc r1

inc dptr

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\_dptr

mov r1, #0h

prpm2:

movx a, @dptr

mov @r1, a

inc r1

inc dptr

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\_dptr: 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:**

Работу разработанной операционной системы можно косвенно наблюдать двумя способами. Первый – работают все три задачи так, словно они работают одновременно: номер нажатой клавиши преобразуется в число и отображается на ЖКИ. Второй – с помощью осциллографа: попеременно формируется меандр на выходах порта P1.1, P1.2, P1.3.

Разработанная ОС не изолирует задачи друг от друга: все задачи исполняются в одном адресном пространстве и регистровый контекст состоит лишь из регистров: SP, ACC, PSW, DPTR, R0 и R1. Все остальные регистры являются разделяемыми. Изолировать программы друг от друга принципиально невозможно, поскольку МК не предоставляет механизмов защиты памяти.

1. Выводы

В данной лабораторной работе были изучены таймеры-счётчики и система прерываний.

Таймеры-счетчики предназначены для подсчёта внешних событий, для формирования программно управляемых временных задержек и выполнения времязадающих функций в МК. В работе с их использованием были решены задачи программирования часов и секундомера. Однако при работе с обычными таймерами существует проблема, так как не во всех режимах они перезагружаются автоматически. Следовательно, их необходимо перезагружать в собственном обработчике прерывания по переполнению, на что тратится время. Этот фактор необходимо учитывать при расчёте числа перезагрузки таймера.

Кроме обычных таймеров-счётчиков в МК есть так называемый блок быстрого ввода-вывода (HSIO). Он позволяет формировать требуемые выходные сигналы без постоянного участия микроконтроллера. Ему не требуется прерывать текущую задачу, а потому реакция на события осуществляется более быстро. Для формирования ШИМ-сигнала блок HSIO используется в режиме сравнения.

Команда lcall 128h позволяет отлаживать работу микроконтроллерных систем, назначение которых формирование выходного сигнала в зависимости от входного. Процедура отладки системы несложна и описана в соответствующем пункте работы. Используя опцию «формирование входа» на инструментальной ЭВМ была отлажена программа формирования ШИМ-сигнала, при отключённом объекте управления.

Кроме сказанного выше, блок HSIO осуществляет безопасное чтение регистровой пары TH/TL без остановки таймера (захват события). Как захват, так и сравнение выполняются внутренними 16-разрядными схемами HSIO. Это обеспечивает точность работы - перезагрузка таймера происходит за 1 машинный цикл без участия процессора.

На примере решения задачи «электронный секундомер» были рассмотрены формирование и обработка внешних прерываний (int0/int1). Когда нужно просто зафиксировать событие, удобно использовать одну или несколько линий запроса таких прерываний. К этим линиям могут подключаться устройства, как напрямую, так и через специальный контроллер.

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.

Самым сложным оказалось задание, связанное с разработкой простейшей многозадачной операционной системы. После детальной отладки и дополнительного изучения материала проблемы, связанные с пониманием механизмов сохранения и восстановления контекстов, были решены и программа стала правильно работать.

Микроконтроллер имеет большой ряд достоинств, однако, есть и заметный недостаток - невозможно изолировать друг от друга программы, поскольку он не предоставляет никаких механизмов защиты памяти. Так и простейшая ОС, реализованная в последнем задании лабораторной работы, не изолирует задачи друг от друга: все они исполняются в одном адресном пространстве и регистровый контекст состоит лишь из регистров: SP, ACC, PSW, DPTR, R0 и R1. Все остальные регистры являются разделяемыми.

Однако, стоит заметить, что наличие встроенных механизмов защиты памяти в изучаемом МК не является необходимым, так как он предназначен для решения совершенно иных задач, нежели псевдопараллелизация выполнения заданий во времени.