Министерство общего и профессионального образования
Российской Федерации
Уральский государственный технический университет

ВЗАИМОДЕЙСТВИЕ МИКРОКОНТРОЛЛЕРА INTEL 8051 С ОБЪЕКТАМИ УПРАВЛЕНИЯ

Методические указания к лабораторной работе №4
по курсу “Цифровые устройства и микропроцессоры”
для студентов всех форм обучения специальностей
200700 – Радиотехника;
201500 – Бытовая радиоэлектронная аппаратура

Екатеринбург 2001

УДК 681.322

Составитель В.А.Добряк

Научный редактор доц., канд. техн. наук В.И.Елфимов

ВЗАИМОДЕЙСТВИЕ МИКРОКОНТРОЛЛЕРА INTEL 8051 С ОБЪЕКТАМИ УПРАВЛЕНИЯ: Методические указания к лабораторной работе №4 по курсу “Цифровые устройства и микропроцессоры”/ В.А.Добряк. Екатеринбург: Изд-во УГТУ, 2001. 24 с.

Методические указания предназначены для использования при выполнении лабораторного практикума и могут использоваться при курсовом проектировании. Содержат описание организации взаимодействия микроконтроллеров с объектами управления, контрольные вопросы, примеры программ на языке ассемблера, порядок выполнения домашнего и лабораторного заданий.

Библиогр.: 5 назв. Рис. 10.

Подготовлено кафедрой радиоэлектроники информационных систем.

ОГЛАВЛЕНИЕ

1. ЦЕЛЬ И СОДЕРЖАНИЕ РАБОТЫ

Целью работы является изучение основ организации взаимодействия микроконтроллеров семейства 8051 Intel с объектами управления. Цель состоит также в продолжение начатого в лабораторных работах №1 - №3 [3 - 5] изучения интегрированной среды ProView фирмы Franklin Software Inc., предназначенной для разработки программного обеспечения этого семейства. Работа рассчитана на 4 - 8 часов домашней подготовки и 4 - 8 часов занятий в лаборатории в зависимости от количества выполняемых заданий.

При подготовке к работе изучаются основные приёмы программирования, направленные на организацию работы с различными объектами управления. Далее составляются программы на языке ассемблера, и разрабатывается методика их отладки. Перед началом лабораторной работы проводится коллоквиум. Студенты, выполнившие домашнее задание, а также успешно ответившие на поставленные вопросы, допускаются к лабораторной части работы. При выполнении лабораторного задания осуществляется ввод исходных текстов, трансляция и отладка программ. Затем оформляется и защищается отчёт с указанным ниже содержанием.

2. ЗАДАНИЯ ДЛЯ ДОМАШНЕЙ ПОДГОТОВКИ

2.1. Изучение аппаратных средств

Изучите аппаратные средства микроконтроллера Intel 8051 (К1816ВЕ51), предназначенные для взаимодействия с объектами управления [1 - 3]:

Параллельные порты P0 - P3. Альтернативные функции порта P3.

Регистры таймеров TH0, TL0 и TH1, TL1. Работа в режиме таймера и в режиме счётчика. Регистр режима таймера/счётчика TMOD, формат управляющего слова. Режимы работы таймера/счётчика. Регистр управления/статуса таймера TCON, функциональное назначение разрядов.

Буфер приёмопередатчика SBUF. Режимы работы последовательного порта. Регистр управления приёмопередатчиком SCON, функциональное назначение разрядов. Установка скорости приёма/передачи.

Регистр управления мощностью PCON, функциональное назначение разрядов.

Система прерываний микроконтроллера. Регистр приоритетов прерываний IP, регистр маски прерываний IE.

2.2. Изучение системы команд

Изучите систему команд микроконтроллера с точки зрения поддержки взаимодействия с объектами управления [3, 4]:

Типы операндов и структура информационных связей. Символические имена регистров специальных функций и портов. Адресация битов в регистрах специальных функций, карта адресуемых битов. Команды работы с регистрами специальных функций.

2.3. Изучение типовых процедур взаимодействия микроконтроллера с объектами управления и подготовка программ

Изучите раздел методических указаний “ВЗАИМОДЕЙСТВИЕ МИКРОКОНТРОЛЛЕРА С ОБЪЕКТАМИ УПРАВЛЕНИЯ”.

Для дальнейшей работы в лаборатории от Вас потребуется значительный объем домашней подготовки. Подготовьте к отладке следующие программы:

1. Подпрограмма обработки внешнего прерывания уровня 0 (SUBIN0).
2. Ожидание импульсного сигнала (WAITIMP).
3. Устранение дребезга контактов на основе счетчика (DBNC).
4. Подсчёт числа импульсов между двумя событиями (CNTEVNT).
5. Подсчёт числа импульсов за заданный промежуток времени на основе двух таймеров/счётчиков (CNTTIME).
6. Ожидание заданного кода (WTCODE).
7. Опрос группы двоичных датчиков с передачей управления прикладным программам (GOCODE).
8. Опрос группы импульсных датчиков (KBRD).
9. Формирование временной задержки длительностью 100 мкс программным способом (DELAY).
10. Формирование временной задержки длительностью 1 с программным способом (DLY1).
11. Формирование временной задержки с помощью таймера (TIMER).
12. Измерение временных интервалов программным способом (MSCONT).
13. Измерение временных интервалов на основе таймера (MSTIMER).
14. Формирование статических сигналов (OUT).
15. Генерация бесконечной последовательности импульсов скважностью 2 на основе примеров из раздела 3.4.2.
16. Программа работы с последовательным портом (UART).

Внимание! В ниже приведенных примерах программ преднамеренно допущены ошибки. Ваша задача состоит в обнаружении и устранении этих ошибок на этапах изучения исходных текстов и отладки программ.

Некоторые примеры программ оставляют возможность оптимизации с учётом особенностей системы команд микроконтроллера. Выполните оптимизацию.

Тщательно продумайте методики отладки каждой из программ на основе средств ProView. Дополните программы командами и директивами, обеспечивающими тестирование и дальнейшую отладку в лаборатории.

Возможно, что при работе с ProView Вы заметите ошибки, допущенные разработчиком этой программы.

2.4. Контрольные вопросы

1. Перечислите характерные черты архитектуры однокристальных микроконтроллеров, направленные на взаимодействие с объектами управления.
2. Укажите назначение регистров специальных функций.
3. Перечислите альтернативные функции параллельных портов.
4. В каком состоянии находятся параллельные порты после формирования сигнала RST?

5. Может ли порт одновременно являться источником операнда и приемником результата операции?
6. С какой частотой инкрементируется содержимое таймера/счётчика при работе в качестве таймера?
7. Чему равна максимальная частота подсчёта входных сигналов при работе таймера/счётчика в режиме счётчика?
8. Охарактеризуйте режимы работы таймера-счётчика.
9. Охарактеризуйте режимы работы последовательного порта.
10. Для чего предназначен регистр SCON?
11. Как изменить скорость передачи данных через последовательный порт?
12. Для чего используется девятый бит?
13. Перечислите команды операций с битами.
14. Укажите, какие из регистров специальных функций допускают битовую адресацию.
15. Перечислите и охарактеризуйте типы прерываний. Как изменить приоритеты прерываний?
16. Как задать вектор прерывания в программе на языке ассемблера?
17. Почему в примере SUBINFO переход на обработчик прерывания производится по команде SJMP, а не CALL?
18. Как соотносится порядок сохранения содержимого регистров в стеке с порядком восстановления из стека в подпрограмме обработки прерывания?
19. Почему подпрограмма обработки прерывания завершается командой RETI, а не RET?
20. Как организовать процедуру ожидания события с помощью одной команды?
21. Каково время реакции микроконтроллера на событие при программной реализации процедуры ожидания и по прерыванию?
22. Какие ограничения накладываются на длительность обнаруживаемого импульсного сигнала при программной реализации цикла ожидания?
23. Поясните принципы устранения дребезга контактов.
24. Поясните принцип организации процедур подсчёта числа импульсов между двумя событиями и за заданный промежуток времени.
25. Расшифруйте команду MOV TMOD, #01000000B.
26. Поясните принцип организации передачи управления в программе по коду.
27. Как программно и аппаратно формируются задержки разной длительности?
28. Как с помощью микроконтроллера измерить временной интервал? Как оценить точность и диапазон измерения?
29. Поясните принцип генерации статических, периодических и апериодических сигналов.
30. В чём заключается табличный способ генерации микроконтроллером сложных последовательностей управляющих сигналов?
31. Перечислите и охарактеризуйте средства программы ProView, предназначенные для отладки взаимодействия микроконтроллера с объектами управления.

3. ВЗАИМОДЕЙСТВИЕ МИКРОКОНТРОЛЛЕРА С ОБЪЕКТАМИ УПРАВЛЕНИЯ

3.1. Прерывания

Подпрограмма обработки прерывания должна сохранить в стеке содержимое тех регистров, которые будут в ней использованы, а перед возвратом в прерванную программу должна восстановить их значения. Подпрограмма обработки внешнего прерывания уровня 0 (SUBINO) может, например, иметь следующую структуру:

```
        ORG  3H         ;задание адреса вектора прерывания
        SJMP SUBINO     ;переход на подпрограмму обработки

        ORG  30H
SUBINO: PUSH ACC            ;сохранение аккумулятора
        PUSH PSW            ;сохранение в стеке PSW
        PUSH B              ;сохранение B
        PUSH DPL            ;сохранение DPTR и, возможно,
        PUSH DPH            ;других регистров
        MOV  PSW, #1000B    ;выбор банка регистров 1
;       ...                 собственно обработка прерывания
        POP  ACC            ;восстановление аккумулятора
        POP  PSW            ;восстановление PSW и номера банка
        POP  B              ;восстановление B
        POP  DPL
        POP  DPH            ;восстановление DPTR
        RETI                ;возврат

        END
```

Обратите внимание, что обработчик заканчивается командой возврата из подпрограммы RETI, а переход на него происходит по команде безусловного перехода SJMP. Дело в том, что система прерываний формирует вызов LCALL аппаратно [3].

В целях отладки программу необходимо дополнить следующими командами:

```
        ORG  0H
        SJMP START

        ORG  20H
;без следующих трех команд можно обойтись, если установить
;эти биты непосредственно в окне Main Registers
START:  SETB IT0  ;тип прерывания 0 - "по срезу"
        SETB EX0  ;разрешение внешнего прерывания 0
        SETB EA   ;снятие блокировки прерываний
START1: NOP
        SJMP START1
```

3.2. Ввод информации с датчиков

3.2.1. Опрос двоичного датчика. Ожидание события

В устройствах и системах управления объектами события фиксируются с использованием разнообразных датчиков цифрового и аналогового типов. Наибольшее распространение имеют двоичные датчики типа да/нет.

Ожидание статического сигнала. Типовая процедура ожидания события (WAIT) состоит из следующих действий: ввода сигнала от датчика, анализа значения сигнала и передачи управления в зависимости от состояния датчика. Конкретная программная реализация процедуры зависит от того, каким образом датчик подключен к микроконтроллеру. Например, при подключении датчика к линии бита 3 порта 1 программа ожидания размыкания контакта будет иметь вид:

```
WAIT0:  JNB  P1.3, WAIT0 ;ожидание размыкания контакта датчика
```

Другим частным случаем является процедура ожидания замыкания контакта, которая может быть реализована следующим образом:

```
WAITC:  JB   P1.3, WAITC ;ожидание замыкания контакта датчика
```

Для опроса особо важных датчиков с целью уменьшения времени реакции на исключительную ситуацию в объекте целесообразно использовать режим прерывания.

Ожидание импульсного сигнала. Особенность процедуры ожидания импульсного сигнала состоит в том, что микроконтроллер должен обнаружить не только факт появления, но и факт окончания сигнала.

Для программирования этой процедуры удобно воспользоваться рассмотренными выше примерами, смонтировав их последовательно в линейную программу. Оформлять процедуры WAITC и WAIT0 в виде подпрограмм нецелесообразно, так как это удлиняет программу, а длина и, следовательно, время исполнения программы определяют минимальную длительность импульса, который может быть обнаружен программой.

Последовательность следования процедур WAITC и WAIT0 зависит от формы импульса. Для “отрицательного” импульса (1→0→1) процедура WAITC предшествует процедуре WAIT0, для “положительного” (0→1→0) следует за ней.

Ниже приведён пример программной реализации процедуры ожидания “отрицательного” импульсного сигнала (WAITIMP) при подключении датчика к биту 3 порта 1 при условии, что начальное состояние входа – единичное.

```
WAITIMP:
WAITC:      JB   P1.3, WAITC    ;ожидание P1.3=0
WAIT0:      JNB  P1.3, WAIT0    ;ожидание P1.3=1
```

Программная реализация цикла ожидания накладывает ограничения на длительность импульса: импульсы длительностью меньше времени выполнения цикла ожидания могут быть “не замечены” микроконтроллером. Для обнаружения крат-

ковременных импульсов обычно используют способ фиксации импульса на внешнем триггере флага. На вход в этом случае поступает не кратковременный сигнал с датчика, а флаг, формируемый триггером. Триггер устанавливается по фронту импульса, а сбрасывается программным путем – выдачей специального управляющего воздействия. Длительность импульса при этом будет ограничена снизу только быстродействием триггера.

3.2.2. Устранение дребезга контактов

При работе с датчиками, имеющими механические или электромеханические контакты (кнопки, клавиши, реле и клавиатуры), возникает явление, называемое дребезгом. Он заключается в том, что при замыкании контактов возможно появление отскока (BOUNCE) контактов, которое приводит к переходному процессу. При этом сигнал с контакта может быть прочитан микроконтроллером как случайная последовательность нулей и единиц. Подавить это нежелательное явление можно схемотехническими средствами, но чаще это делается программным путем.

Наибольшее распространение получили два программных способа ожидания установившегося значения [1].

Подсчет заданного числа совпадающих значений сигнала. Его суть состоит в многократном считывании сигнала с контакта. Подсчет удачных опросов, обнаруживших, что контакт устойчиво замкнут, ведется счетчиком. Если после серии удачных опросов встречается неудачный, то подсчет начинается сначала. Контакт считается устойчиво замкнутым, если последовало N удачных опросов. Число N подбирается экспериментально для каждого типа используемых датчиков и лежит в пределах от 5 до 50. Пример программного подавления дребезга контакта (DBNC) приводится для случая, когда датчик импульсного сигнала подключен к входу T0, счет удачных опросов ведется в регистре R3, N=20.

```
DBNC:   MOV  R3, #20   ;инициализация счетчика
DBNC1:  JB   P3.4, DBNC;если контакт разомкнут, то
                       ;начать отсчёт опросов сначала
        DJNZ R3, DBNC1 ;повторять, пока R3 не станет равным 0
```

Введение временной задержки. Устранение дребезга контакта путем введения временной задержки заключается в следующем. Программа, обнаружив замыкание контакта, запрещает опрос его состояния на время, заведомо большее длительности переходного процесса. Программа (DBNCDL) написана для случая подключения датчика к входу T0 и программной реализации временной задержки:

```
DBNCDL: JB   P3.4, DBNCDL   ;ожидание нуля на входе T0
        CALL DELAY          ;вызов подпрограммы задержки
EXIT:   ...                 ;выход из процедуры
```

Временная задержка в пределах 1-10 мс подбирается экспериментально для каждого типа датчиков и реализуется подпрограммой DELAY.

3.2.3. Подсчет числа импульсов

Часто в управляющих программах возникает необходимость ожидания цепочки событий, представляемой последовательностью импульсных сигналов от

датчиков. Рассмотрим две типовые процедуры: подсчет числа импульсов между двумя событиями и подсчет числа импульсов в заданный интервал времени.

Подсчет числа импульсов между двумя событиями. Один из возможных вариантов процедуры подсчёта (CNTEVNT) может быть реализован, если использовать вход T1 как вход счетчика внешних импульсов, а T0 – для программного включения и выключения счета. В аккумуляторе фиксируется число импульсов, деленное на 32, так как TL1 работает как 5-битный предделитель [3].

```
CNTEVNT:MOV  TMOD, #0100000B     ;настройка счетчика 1
        MOV  TH1, #0         ;сброс счетчика импульсов
WAIT0:  JB   P3.4, WAIT0     ;ожидание включения счёта
        SETB TCON.6          ;пуск счетчика 1
WAIT1:  JNB  P3.4, WAIT1     ;ожидание выключения счёта
        CLR  TCON.6          ;останов счетчика 1
        MOV  A, TH1          ;фиксация 1/32 числа импульсов
EXIT:   ...                  ;выход из процедуры
```

Подсчет числа импульсов за заданный промежуток времени. При решении задачи преобразования числоимпульсного кода в двоичный код, а также в ряде других задач может возникнуть необходимость подсчёта числа импульсов за заданный интервал времени. Эта процедура может быть реализована различными способами [1]:

- программной реализацией временного интервала и программным подсчетом числа импульсов на входе;
- программной реализацией временного интервала и аппаратным подсчетом числа импульсов (на внутреннем таймере/счетчике);
- аппаратной реализацией временного интервала и программным подсчетом числа импульсов;
- аппаратной реализацией временного интервала с аппаратным подсчетом числа импульсов.

Последний способ подсчёта требует использования двух счётчиков (CNTTIME). На T/C1 можно выполнять подсчёт числа импульсов, а на T/C0 - отсчёт заданного интервала. Датчик импульсов должен быть подключен к входу T1.

```
TIME    EQU  NOT(10000)+1   ;определение константы TIME для
                            ;отсчета интервала 10 мс
CNTTIME:MOV  TMOD, #01010001B    ;настройка T/C, 16 бит
                                 ;1 – счетчик, 0 – таймер
        CLR  A                   ;сброс аккумулятора
        MOV  TH1, A              ;сброс T/C1
        MOV  TL1, A
        MOV  TH0, #HIGH(TIME)    ;загрузка в T/C0
        MOV  TL0, #LOW(TIME)     ;константы TIME
        ORL  TCON, #50H          ;пуск T/C1 и T/C0
WAIT:   JBC  TCON.5, EXIT        ;проверка переполнения T/C0
        SJMP WAIT                ;цикл, если TF=0
EXIT:   MOV  B, TH1              ;фиксация числа импульсов
        MOV  A, TL1
        ...                      ;выход из процедуры
```

3.2.4. Опрос группы двоичных датчиков

Микроконтроллеры чаще всего имеют дело не с одним датчиком, как в рассмотренных выше примерах, а с группой автономных, логически независимых или взаимосвязанных, формирующих двоичный код датчиков. При этом микроконтроллер может выполнять процедуру опроса датчиков и передачи управления отдельным фрагментам прикладной программы в зависимости от принятого кода.

Ожидание заданного кода. Программную реализацию процедуры ожидания заданного кода (WTCODE) рассмотрим для случая подключения группы из восьми взаимосвязанных статических датчиков к входам порта 1.

```
WTCODE: MOV  A, #10    ;загрузка в аккумулятор эталонного кода
WAIT:   CJNE A, P1, WAIT    ;если кодовая комбинация на входах
                            ;порта 1 не совпала с эталонным
                            ;значением, то ждать
EXIT:   ...                 ;выход из процедуры
```

Передача управления по коду. При опросе двоичных датчиков передачу управления удобно осуществлять по таблице переходов. Ниже приводится текст программы (GOCODE), осуществляющей передачу управления одной из восьми прикладных программ PROG0 - PROG7, которые расположены в пределах одной страницы памяти программ. Передача производится в зависимости от кодовой комбинации на входах P1.0 - P1.2.

Адрес строки таблицы, в которой хранятся адреса переходов, вычисляется как сумма относительного (внутри текущей страницы резидентной памяти программ) начального адреса таблицы BASE и кода, принятого от датчиков. Команды MOVC A, @A+DPTR , JMP @A+DPTR, таблица BASE и программы PROG0 - PROG7 должны располагаться в одной странице памяти программ.

```
GOCODE: MOV  DPTR, #LOW BASE;загрузка в DPTR начального адреса
                            ;таблицы переходов
        IN   A, P1          ;ввод байта
        ANL  A, #00000111B  ;выделение младших бит
        MOVC A, @A+DPTR     ;чтение из памяти программ
                            ;(из таблицы переходов)
                            ;адреса перехода
        MOV  DPTR,#0000H    ;обнуление DPTR
        JMP  @A+DPTR        ;передача управления

BASE:   DB   LOW PROG0 ;таблица переходов
        DB   LOW PROG1
        ...
        DB   LOW PROG7

PROG0:  ...             ;прикладные программы
PROG1:  ...
...
PROG7:  ...
```

Опрос группы импульсных датчиков. Эта процедура состоит из последовательности действий: ожидания замыкания одного из контактов, устранения дребезга, ожидания размыкания замкнутого контакта.

Программная реализация процедуры (KBRD) для случая подключения четырех импульсных датчиков к входам 0 - 3 порта 1 будет иметь вид:

```
KBRD:   IN   A, P1          ;ввод кода
        CPL  A              ;инверсия кода
        ANL  A, #00001111B  ;есть замкнутый контакт?
        JZ   KBRD           ;если ни один контакт не замкнут,
                            ;то ждать
        MOV  R2, A          ;передача принятого кода в R2
DBNC:   CALL DELAY          ;устранение дребезга
WAIT:   IN   A, P1          ;ввод кода
        CPL  A              ;инверсия кода
        ANL  A, #00001111B  ;есть замкнутый контакт?
        JNZ  WAIT           ;если контакт замкнут, то ждать,
EXIT:   ...                 ;иначе выход из процедуры
```

Анализ состояния контактов осуществляется наложением маски на принятый от датчиков код. Для датчиков, формирующих “отрицательный” импульс, принятый код предварительно инвертируется.

Для группы импульсных датчиков, представляющих собой клавишный регистр, процедура KBRD должна быть дополнена процедурами идентификации нажатой клавиши и защиты от одновременного нажатия двух и более клавиш.

Идентификация нажатой клавиши может осуществляться двумя способами: по таблице или программно. При табличном способе перекодирования в памяти программ должна находиться таблица двоичных эквивалентов кодов клавиш. Программное преобразование унитарного кода, принятого от клавиатуры, в двоичный код может быть выполнено методом сдвигов исходного унитарного кода и подсчетом числа сдвигов на счетчике до появления первого переноса [1].

3.3. Реализация функций времени

3.3.1. Программное формирование временной задержки

Временная задержка малой длительности. Процедура реализации временной задержки использует метод программных циклов. При этом в некоторый рабочий регистр загружается число, которое затем в каждом проходе цикла уменьшается на 1. Так продолжается до тех пор, пока содержимое регистра не станет равным нулю, что интерпретируется программой как момент выхода из цикла. Время задержки при этом определяется числом, загруженным в рабочий регистр, и временем выполнения команд, образующих программный цикл.

Предположим, что в управляющей программе необходимо реализовать временную задержку 99 мкс. Фрагмент программы (DELAY) оформим в виде подпрограммы, так как предполагается, что основная управляющая программа будет производить к ней многократные обращения для формирования выходных импульсных сигналов, длительность которых кратна 99 мкс:

```
DELAY:  MOV  R2, #X     ;загрузка числа X
COUNT:  DJNZ R2, COUNT  ;декремент R2 и цикл, если не равно 0
        RET             ;возврат
```

Для получения требуемой временной задержки необходимо определить число X, загружаемое в рабочий регистр. Определение числа X выполняется на основе расчёта времени выполнения команд, образующих данную подпрограмму. При этом необходимо учитывать, что команды MOV и RET выполняются однократно, а число повторений команды DJNZ равно числу X. Кроме того, обращение к подпрограмме временной задержки осуществляется по команде CALL DELAY, время исполнения которой также необходимо учитывать при подсчете временной задержки. В описании команд микроконтроллера [4] указывается, за сколько машинных циклов (МЦ) исполняется каждая команда: CALL - 2 МЦ, MOV - 1 МЦ, DJNZ - 2 МЦ, RET - 2 МЦ. На основании этих данных определяется суммарное число машинных циклов в подпрограмме.

При тактовой частоте 12 МГц каждый машинный цикл выполняется за 1 мкс. Таким образом, подпрограмма выполняется за время 2 + 1 + 2X + 2 = 5 + 2X мкс. Для реализации временной задержки 99 мкс число X = (99 - 5)/2 = 47.

В данном случае при загрузке в регистр R2 числа 47 требуемая временная задержка (99 мкс) реализуется точно. Если число X получается дробным, то временную задержку можно реализовать лишь приблизительно. Для более точной подстройки в подпрограмму могут быть включены команды NOP, время выполнения каждой из которых равно 1 мкс.

Минимальная временная задержка, реализуемая подпрограммой DELAY, составляет 7 мкс (X = 1). Временную задержку меньшей длительности программным путем можно реализовать, включая в программу цепочки команд NOP.

Максимальная длительность задержки, реализуемая подпрограммой DELAY, составляет 515 мкс (X = 255). Для реализации задержки большей длительности можно рекомендовать увеличить тело цикла включением дополнительных команд или использовать метод вложенных циклов. Так, например, если в подпрограмму DELAY перед командой DJNZ вставить дополнительно две команды NOP, то максимальная задержка составит 5 + X(2 + 1 + 1) = 5+4 * 255 = 1025 мкс (т.е. почти в два раза больше).

Временная задержка большой длительности. Задержка большой длительности может быть реализована методом вложенных циклов. Числа X и Y выбираются из соотношения T = 2 + 1 + X(1 + 2Y + 2) + 2, где T - реализуемый временной интервал в микросекундах. Максимальный временной интервал, реализуемый таким способом, при X = Y = 255 составляет 130.82 мс, т.е. приблизительно 0.13 с.

В качестве примера рассмотрим подпрограмму (DLY100), реализующую временную задержку 100 мс.

Здесь два вложенных цикла реализуют временную задержку длительностью 5 + 195(3 + 2*254) = 99 650 мкс, а дополнительный цикл LOOPAD и команда NOP реализует задержку 350 мкс и тем самым обеспечивает точную настройку временного интервала.

```
DLY100: MOV  R1, #195       ;загрузка X
LOOPEX: MOV  R2, #254       ;загрузка Y
LOOPIN: DJNZ R2, LOOPIN     ;декремент R2 и внутренний цикл,
                            ;если (R2) не равно 0
        DJNZ R1, LOOPEX     ;декремент R1 и внешний цикл,
                            ;если (R1) не равно 0
        MOV  R3, #174       ;точная подстройка
LOOPAD: DJNZ R3, LOOPAD     ;временной
        NOP                 ;задержки
        RET                 ;возврат
```

Временная задержка длительностью 1 с. Из предыдущего примера видно, что секунда является очень большим интервалом времени по сравнению с тактовой частотой микроконтроллера. Такие задержки сложно реализовать методом вложенных циклов, поэтому их обычно набирают из точно подстроенных задержек меньшей длительности. Например, задержку в 1 с можно реализовать десятикратным вызовом подпрограммы, реализующей задержку 100 мс:

```
DLY1:   MOV  R4, #10   ;загрузка в R4 числа вызовов DELAY
LOOP:   CALL DLY100    ;задержка 100 мс
        DJNZ R4, LOOP  ;R4-1 и цикл, если (R4) не равно 0
```

Погрешность подпрограммы DLY1 составляет 21 мкс. Для очень многих применений это достаточно высокая точность, хотя реализованные на основе этой программы часы астрономического времени за сутки “убегут” примерно на 1.8 с.

3.3.2. Формирование временной задержки таймером

Недостатком программного способа реализации временной задержки является нерациональное использование ресурсов микроконтроллера: во время формирования задержки он практически простаивает, так как не может решать никаких задач управления объектом. В то же время аппаратные средства позволяют реализовать временные задержки на фоне работы основной программы.

На вход таймера/счетчика (Т/С) могут поступать сигналы синхронизации с частотой 1 МГц (Т/С в режиме таймера) или сигналы от внешнего источника (Т/С в режиме счетчика). Оба эти режима могут быть использованы для формирования задержек [1]. Если использовать Т/С в режиме таймера полного формата (16 бит), то можно получить задержки в диапазоне 1 - 65 536 мкс.

В качестве примера рассмотрим организацию временной задержки 50 мс (TIMER). Предполагается, что бит IE.7 установлен, т.е. снята блокировка всех прерываний.

Обратите внимание, что здесь использована команда перевода микроконтроллера в режим холостого хода, который прекращается после истечения 50 мс. Этот режим реализован в микроконтроллере Intel 80C51 (отечественный аналог КР1830ВЕ51), изготовленный по технологии КМОП [2]. Поэтому для отладки этой программы через пункт Debug меню Options выберите указанный тип микроконтроллера.

```
;организация перехода к метке NEXT при переполнении T/C0
        ORG  0BH        ;адрес вектора прерывания от T/C0
        CLR  TCON.4     ;останов T/C0
        RETI            ;выход из обработчика прерывания

        ORG  30         ;начальный адрес программы
TIMER:  MOV  TMOD, #01H                ;настройка T/C0
        MOV  TL0, #LOW(NOT(50000-1))   ;загрузка таймера для
        MOV  TH0, #HIGH(NOT(50000-1)) ;задержки 50 мс
        SETB TCON.4     ;старт T/C0
        SETB IE.1       ;разрешение прерывания от T/C0
        SETB PCON.0     ;перевод в режим холостого хода
NEXT:   ...
```

3.3.3. Измерение временных интервалов

В задачах управления часто возникает необходимость измерения промежутка времени между двумя событиями. Обычно события в объекте управления представляются сигналами от двоичных датчиков. Считая событиями фронт и спад импульса, можно определять временные характеристики импульсных сигналов: длительность, период и скважность.

Программный способ. Это простейший способ измерения длительности импульса. Для обнаружения событий (фронт и спад импульсного сигнала) в этом случае используются типовые процедуры WAIT, а отсчёт времени ведется программным способом. Для “положительного” импульсного сигнала, поступающего на вход T0, программа измерения его длительности (MSCONT) будет иметь вид:

```
MSCONT: MOV  R7, #0     ;сброс счётчика
WAIT0:  JNB  T0, WAIT0  ;ожидание фронта сигнала
COUNT:  INC  R7         ;инкремент счётчика
        JB   T0, COUNT  ;ожидание спада сигнала
EXIT:   ...             ;выход из процедуры
```

После выхода из процедуры содержимое счетчика R7 пропорционально длительности импульса.

Для нормальной работы этой программы необходимо, чтобы обращение к ней производилось в моменты, когда на входе T0 присутствует сигнал нулевого уровня. Верхний предел измеряемой длительности “положительного” импульса составит 255(1 + 2) мкс = 765 мкс. Этот предел может быть увеличен включением в цикл COUNT дополнительных команд NOP. Максимальная погрешность измерений 3 мкс.

Применение таймера. Для измерения длительности сигнала может быть использован таймер. Особенно эффективно использование для этой цели таймера в 8051 Intel, имеющего вход разрешения счёта (альтернативная функция входа INT). Измеряемый сигнал можно, например, подавать на вход INT0, а измерение длительности при этом будет выполняться в T/C0. Программа измерения длительности “положительного” импульса (MSTIMER) будет выглядеть так:

```
MSTIMER:MOV  TMOD, #00001001B    ;настройка T/C0
        MOV  TH0, #0             ;сброс таймера
        MOV  TL0, #0
        SETB TCON.4              ;старт T/C0
WAIT0:  JNB  P3.2, WAIT0         ;ожидание 1
WAITC:  JB   P3.2, WAITC         ;ожидание 0
        CLR  TCON.4              ;стоп T/C0
EXIT:   ...                      ;выход из процедуры
```

Управление программе должно быть передано при условии, что на входе INT0 присутствует низкий уровень. Прерывание от T/C0 и внешнее прерывание по входу INT0 должны быть запрещены. По завершении программы в T/C0 будет находиться число, пропорциональное длительности “положительного” импульса на входе INT0. Верхний предел измерения равен 65 536 мкс, а максимальная погрешность 1 мкс.

При необходимости измерения временных интервалов большей длительности можно программным способом подсчитывать число переполнений от таймера, т.е. расширять его разрядность за счет рабочего регистра или ячейки резидентной памяти данных.

3.4. Вывод управляющих сигналов

3.4.1. Формирование статических сигналов

Для управления исполнительным устройством, работающим по принципу включено/выключено, на соответствующей выходной линии порта необходимо сформировать статический сигнал 1 или 0, что реализуется командами установки, сброса или инверсии бита порта (OUT) [4].

В случае параллельного управления группой автономных исполнительных устройств, подключенных к выходному порту, формируется не двоичное управляющее воздействие, а управляющее слово, каждому из разрядов которого ставится в соответствие 1 или 0 в зависимости от того, какие исполнительные устройства должны быть включены, а какие выключены.

Управляющие слова удобно формировать командами логических операций над содержимым порта [4]. Командой XRL осуществляется инверсия тех битов, которые в маске заданы единицей. Команда ORL используется для установки битов управляющего слова. Команда ANL используется для сброса битов.

```
OUT:    MOV  P1, #10101010B ;вывод байта в порт P1
        SETB P1.0           ;установка 0-го разряда
        CLR  P1.3           ;сброс 3-го разряда
        CPL  P1.4           ;инверсия 4-го разряда
        CPL  P1.5           ;инверсия 5-го разряда
        CPL  P1.6           ;инверсия 6-го разряда
        CPL  P1.7           ;инверсия 7-го разряда
        XRL  P1, #11110000B ;инверсия разрядов 4-7
        ORL  P1, #00001000B ;установка 3-го разряда
        ANL  P1, #11111110B ;сброс 0-го разряда
```

Для формирования сложных последовательностей управляющих слов обычно используют табличный способ, при котором все возможные слова упакованы в таблицу, а прикладная программа вычисляет адрес требуемого слова, выбирает его из таблицы и передаёт в порт.

3.4.2. Формирование импульсных сигналов

Импульс. Его можно получить последовательной выдачей сигналов включить и отключить (PULS) с промежуточным вызовом подпрограммы временной задержки:

```
PULS:                   ;выдача импульса в линию 3 порта 1
ON:     SETB P1.3       ;включение
        CALL DELAY      ;временная задержка
OFF:    CLR  P1.3       ;отключение
        ...
```

Длительность импульса определяется временной задержкой, реализуемой подпрограммой DELAY.

Генерация меандра. В этом случае можно воспользоваться подпрограммой задержки, равной половине периода сигнала:

```
MEANDR: CPL  P1.3
        CALL DELAY
        SJMP MEANDR
```

Бесконечный периодический сигнал формируется в линии 3 порта 1. На остальных линиях сигналы остаются неизменными.

Формирование апериодических управляющих сигналов. Последовательность импульсных сигналов с произвольной длительностью и скважностью может быть получена аналогичным образом, то есть путём чередования процедур выдачи значения 0 или 1 и вызова подпрограмм временных задержек заданных длительностей.

3.5. Работа с последовательным портом

Последовательный порт микроконтроллера может использоваться в виде регистра сдвига для расширения ввода-вывода или в качестве UART с фиксированной или переменной скоростью последовательного обмена и возможностью дуплексного включения [3]. То есть через порт можно передавать и принимать данные одновременно. Порт может принимать очередной байт даже в том случае, если уже принятый до этого байт не был прочитан из регистра приёмника. Однако если до окончания приёма находящийся в регистре приёмника байт не будет прочитан, принятый байт теряется. Программный доступ к регистрам приёмника и передатчика осуществляется обращением к регистру специальных функций SBUF.

Ниже приведён пример фрагмента программы (UART), принимающей из последовательного порта байт и отправляющей его назад в последовательный порт, настроенный на 8-битный режим со скоростью передачи 2400 бод при тактовой частоте микроконтроллера 12 МГц.

```
UART:   MOV SCON, #052H ;установка режима 8-битного UART
        MOV TMOD, #020H ;уст. режима автозагрузки таймера 1
        MOV TH1, #0F3H ;автозагружаемое значение для получения
                       ;скорости 2400 бод на частоте МК 12 МГц
        SETB TCON.6     ;пуск таймера

;приём символа из порта
CIN:    JNB RI, CIN          ;ожидание завершения приёма
        MOV A, SBUF          ;чтение символа
        CLR RI               ;очистка флага приёма

;выдача символа в последовательный порт
COUT:   JNB TI, COUT         ;ожидание окончания передачи
        CLR TI               ;очистка флага передачи
        MOV SBUF, A          ;выдача символа

        SJMP CIN
```

3.6. Средства ProView для отладки взаимодействия с объектами управления

Интегрированная среда ProView фирмы Franklin Software Inc. имеет несколько окон, предназначенных для отладки взаимодействия микроконтроллера с объектами управления. Во-первых, это уже знакомое по предыдущим лабораторным работам [3 - 5] окно Main Registers (Рис. 1), где доступны регистр масок прерывания IE, блокировка прерываний EA, порты P0 – P3, регистр управления/статуса таймера TCON, регистры таймеров THL0 и THL1, а также регистр управления мощностью PCON.

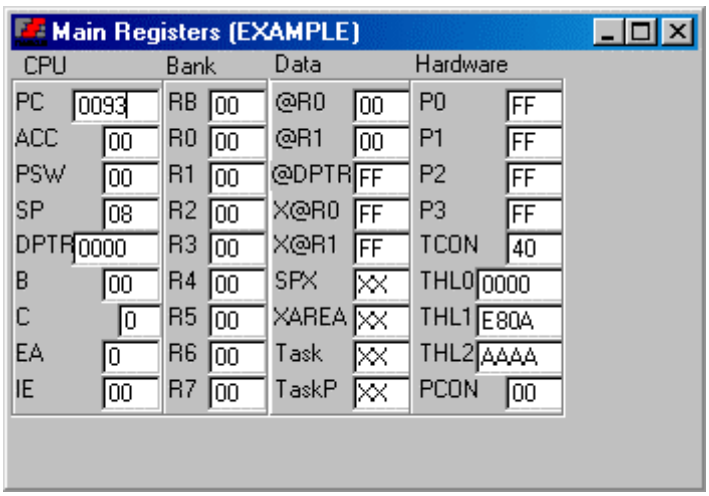

Рис. 1. *Окно регистров.*

Кроме того, через пункт Hardware меню View (Рис. 2) можно открыть окна контроллера прерываний, таймеров, параллельных портов и последовательного порта.

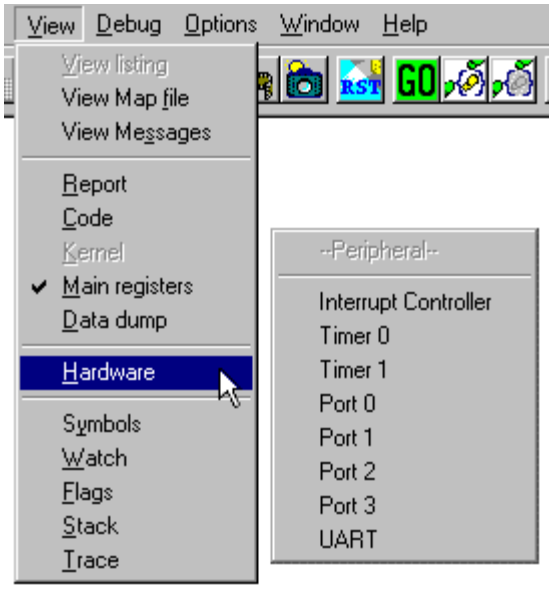

Рис. 2. *Меню View*

В информационном окне контроллера прерываний (Рис. 3) указывается статус и приоритет всех источников прерывания.

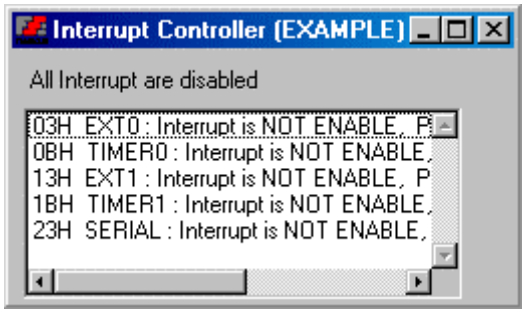

Рис. 3. *Окно контроллера прерываний*

В строке состояния (Рис. 4) при выполнении программы в пошаговом режиме или в режиме анимации показывается текущее реальное время.

Рис. 4. *Строка состояния*

В окнах таймеров (Рис. 5) доступно содержимое 16-битного таймера/счётчика THLx, регистра режима работы таймера/счётчика TMOD, регистра управления/статуса таймера TCON, флаг переполнения таймера TFx, бит управления таймера TRx. Здесь же указывается состояние и режим работы таймера.

В окнах параллельных портов (Рис. 6) доступно содержимое регистра-защёлки LATCH и отдельных линий порта. Состояние линий порта может быть изменено с помощью мыши.

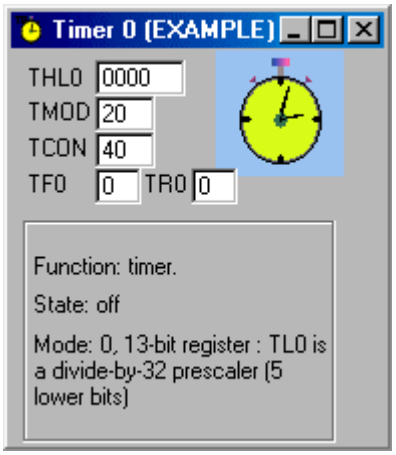

Рис. 5. *Окно таймера*

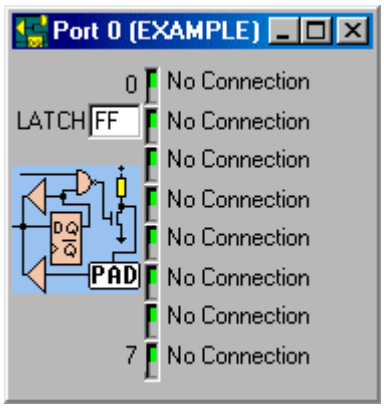

Рис. 6. *Окно параллельного порта*

В окне последовательного порта UART (Рис. 7) отображается содержимое буфера передатчика Buffer, режим работы и скорость. В буфер приёмника Input может быть введена последовательность байтов. Информация может быть представлена в символьном виде ASCII или в шестнадцатеричной форме HEX. Буфер очищается с помощью кнопки Reset Buffer.

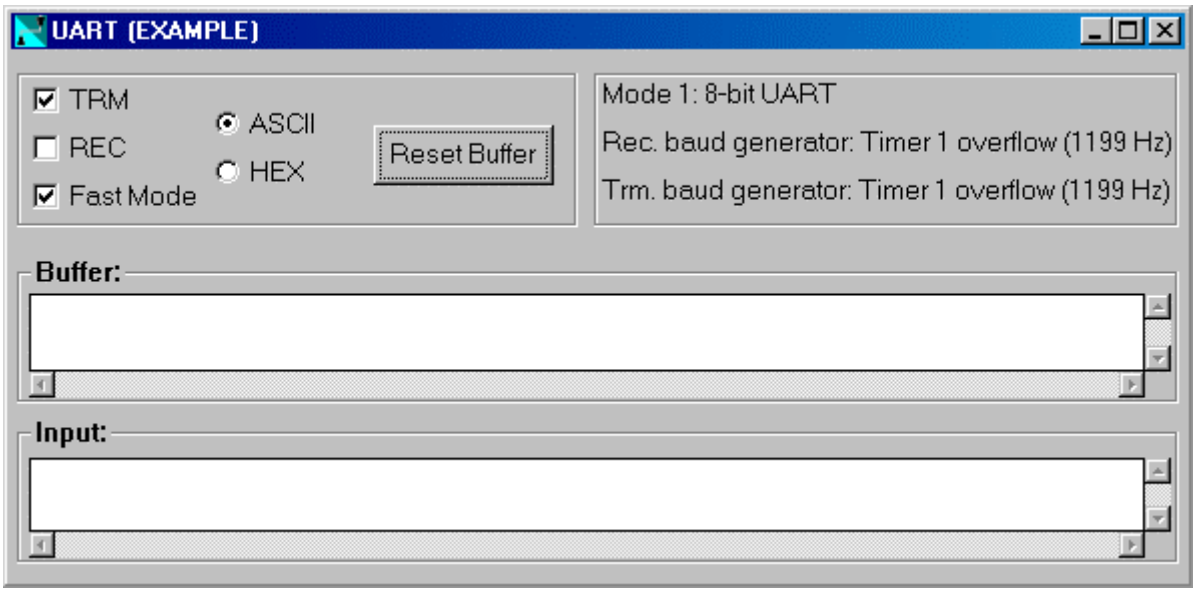

Рис. 7. *Окно последовательного порта*

При отладке программы часто возникает необходимость приостановить ее выполнение на определенной команде.

Breakpoints – это маркеры в вашей программе, которые заставляют отладчик остановить ее выполнение в “реальном масштабе времени”. Вы можете установить

маркеры на любых командах программы (Рис. 8). Когда отладчик WinSim51 доходит до маркера, выполнение программы будет приостановлено, и управление возвращено Вам. В этом случае Вы можете сами принять решение о дальнейшем продолжении работы программы. Breakpoints устанавливаются через пункт Toggle breakpoint из меню Debug (Рис. 9) или из контекстного меню, которое вызывается нажатием правой кнопки мыши в окне с исходным текстом или кодом программы (Рис. 10).

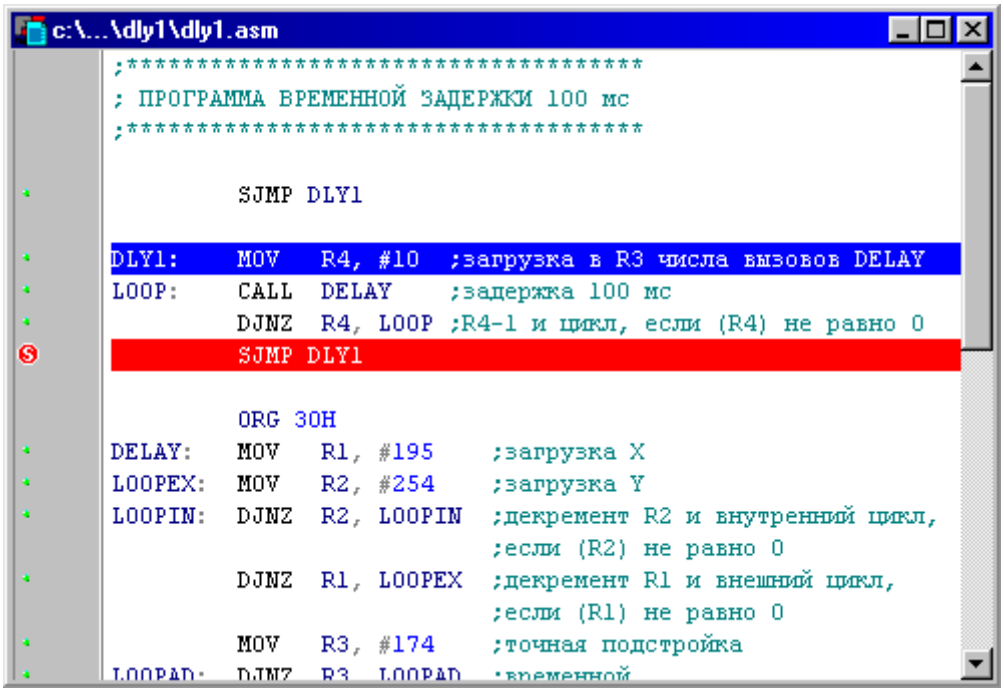

Рис. 8. *Точка останова Breakpoint*

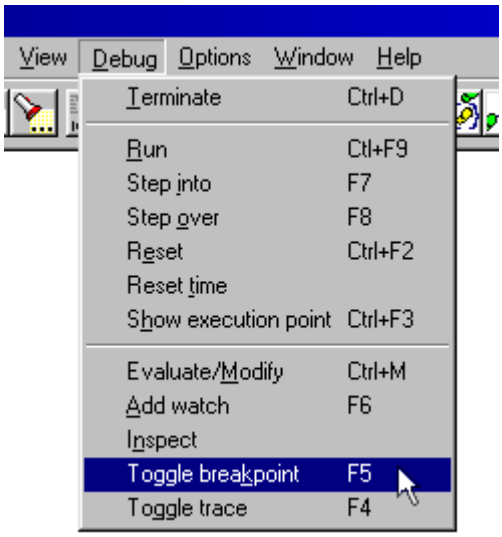

Рис. 9. *Установка Breakpoint из главного меню*

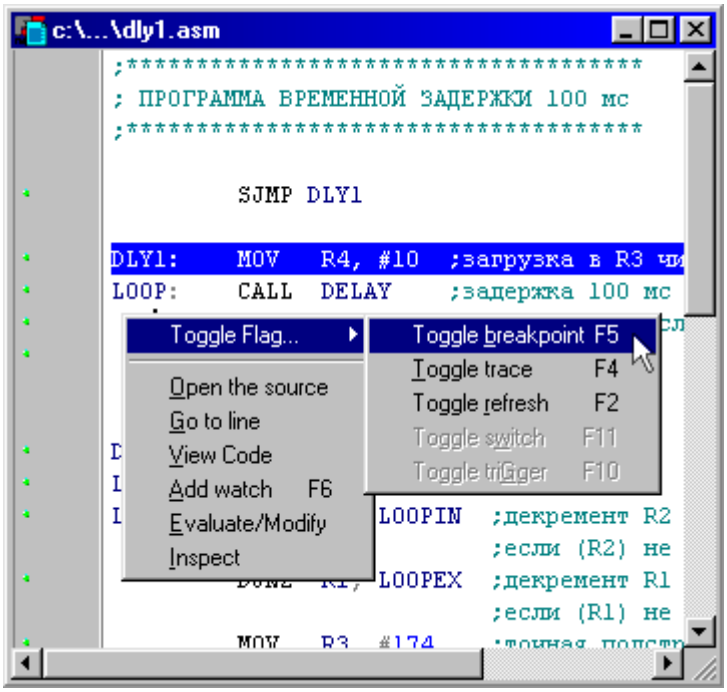

Рис. 10. *Установка Breakpoint из контекстного меню*

4. ЛАБОРАТОРНЫЕ ЗАДАНИЯ

С помощью пакета ProView выполните отладку подготовленных при выполнении домашнего задания программ:

1. Подпрограмма обработки внешнего прерывания уровня 0 (SUBIN0).
2. Ожидание импульсного сигнала (WAITIMP).
3. Устранение дребезга контактов на основе счетчика (DBNC).
4. Подсчёт числа импульсов между двумя событиями (CNTEVNT).
5. Подсчёт числа импульсов за заданный промежуток времени на основе двух таймеров/счётчиков (CNTTIME).
6. Ожидание заданного кода (WTCODE).
7. Опрос группы двоичных датчиков с передачей управления прикладным программам (GOCODE).
8. Опрос группы импульсных датчиков (KBRD).
9. Формирование временной задержки длительностью 100 мкс программным способом (DELAY).
10. Формирование временной задержки длительностью 1 с программным способом (DLY1).
11. Формирование временной задержки с помощью таймера (TIMER).
12. Измерение временных интервалов программным способом (MSCONT).
13. Измерение временных интервалов на основе таймера (MSTIMER).
14. Формирование статических сигналов (OUT).
15. Генерация бесконечной последовательности импульсов скважностью 2.
16. Программа работы с последовательным портом (UART).

Используйте оптимизированные варианты программ и их отладочные дополнения.

Для отладки каждой из программ:

- создайте файл проекта задачи, создайте файл исходного текста программы на языке ассемблера, и добавьте его к проекту;
- выполните трансляцию исходного текста и исправьте возможные синтаксические ошибки;
- запустите отладчик;
- в случае необходимости задайте начальные значения регистров и памяти;
- осуществите пробный пуск программы на контрольном примере в соответствии с разработанной методикой отладки;
- убедитесь в полной работоспособности программы;
- после устранения всех ошибок и отладки выполните генерацию листинга программы, который включите в отчёт о выполнении работы.

5. СОДЕРЖАНИЕ ОТЧЁТА

Отчёт о лабораторной работе должен содержать:

- титульный лист;
- цель и задачи работы;
- **краткое, но точное описание методики отладки каждой из программ;**
- листинги с исходными текстами и объектным кодом отлаженных программ с дополнениями, обеспечивающими тестирование и отладку;
- перечень выявленных в примерах ошибок;
- выводы по работе.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Сташин В.В., Урусов А.В., Мологонцева О.Ф. Проектирование цифровых устройств на однокристальных микроконтроллерах. М.: Энергоатомиздат, 1990. 224 с.
2. Однокристальные микроЭВМ/ А.В.Боборыкин, Г.П.Липовецкий, Г.В.Литвинский и др. М.: МИКАП, 1994. 400 с.
3. Ваша первая программа для микроконтроллера Intel 8051: Методические указания к лабораторной работе №1 по курсу “Микропроцессоры и вычислительные устройства”/ Добряк В.А. Екатеринбург: УГТУ, 1999. 32 с.
4. Система команд микроконтроллера Intel 8051: Методические указания к лабораторной работе №2 по курсу “Цифровые устройства и микропроцессоры”/ Добряк В.А., Рагозин В.К. Екатеринбург: УГТУ, 1999. 32 с.
5. Программирование микроконтроллера Intel 8051 на языке ассемблера: Методические указания к лабораторной работе №3 по курсу “ Цифровые устройства и микропроцессоры”/ Добряк В.А., Рагозин В.К.. Екатеринбург: УГТУ, 1999. 26 с.

ВЗАИМОДЕЙСТВИЕ МИКРОКОНТРОЛЛЕРА
INTEL 8051 С ОБЪЕКТАМИ УПРАВЛЕНИЯ

Составитель Добряк Вадим Алексеевич

Редактор

Подписано в печать __.__.__ Формат 60x84 1/16
Бумага типографская Офсетная печать Усл. п. л. ____
Уч.-изд. л. ____ Тираж 150 Заказ Цена “С”

Издательство УГТУ
620002, Екатеринбург, Мира, 19
ЗАО УМЦ УПИ. 620002, Екатеринбург, Мира, 17