ФЕДЕРАЛЬНОЕ АГЕНТСТВО ПО ОБРАЗОВАНИЮ
Государственное образовательное учреждение высшего профессионального образования
“ТОМСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ”

Лабораторная работа №5

Реализация ввода и вывода сигналов в реальном масштабе времени

Учебное пособие к выполнению лабораторной работы по дисциплине «Микропроцессорная техника» и курсового проекта по дисциплине «Электроника и микроэлектроника» для студентов ФТФ специальности 140306

Томск – 2004

УДК 681.322

Горюнов А.Г. Ливенцов С.Н. Реализация ввода и вывода сигналов в реальном масштабе времени: Учеб. пособие.

Учебное пособие посвящено вопросам организации ввода-вывода сигналов в реальном масштабе времени в микропроцессорной системе, содержит методы организации взаимодействия микроконтроллера с объектами управления. Данное учебное пособие ориентировано на курс лабораторных работ с использованием учебно-лабораторных стендов SDK-1-1 и содержит примеры программ реализации ввода-вывода сигналов в реальном масштабе времени для данного стенда.

Пособие подготовлено на кафедре «Электроника и автоматика физических установок» ТПУ и предназначена для студентов очного обучения специальности 200600.

Учебное пособие рассмотрено и рекомендовано
к изданию методическим семинаром кафедры электроники и автоматики
физических установок "___"_____ 2004г.

Зав. кафедрой
доцент, к.т.н. _____ В. Ф. Дядик

Содержание

1 Цель работы

Изучение методов ввода-вывода различных сигналов в режиме реального времени с помощью микропроцессорной системы.

2 Содержание работы

1. Изучение следующих вопросов организации ввода/вывода сигналов в реальном масштабе времени

Структура МК-системы управления.

Аналого-цифровые преобразователи [1]

1.2.1. Классификация АЦП.
1.2.2. Параллельные АЦП.
1.2.3. Последовательно-параллельные АЦП.
1.2.4. Последовательные АЦП.
1.2.5. Интегрирующие АЦП.
1.2.6. Параметры АЦП

Взаимодействие микроконтроллера с объектами управления

1.3.1. Ввод информации с датчиков.
1.3.2. Вывод управляющих сигналов.
1.3.3. Реализация функций времени.

2. Практическая часть лабораторной работы
   2.1. Разработка программ взаимодействия МК с объектами управления.
   2.2. Проработка примера программы ввода/вывода сигналов для учебно-лабораторного стенда SDK-1-1.
   2.3. Выполнение индивидуального задания.
3. Оформление отчёта.

3 Структура МК-системы управления

Типовая структура МК-системы управления показана на рис. 1 и состоит из объекта управления, микроконтроллера и аппаратуры их взаимной связи [2].

Микроконтроллер путем периодического опроса осведомительных слов (ОС) генерирует в соответствии с алгоритмом управления последовательности управляющих слов (УС). Осведомительные слова это сигналы состояния объекта (СС), сформированные датчиками объекта управления, и флаги. Выходные сигналы датчиков вследствие их различной физической природы могут потребовать промежуточного преобразования на аналого-цифровых преобразователях (АЦП) или на схемах формирователей сигналов (ФС), которые чаще всего выполняют функции гальванической развязки и формирования уровней двоичных сигналов стандарта ТТЛ.

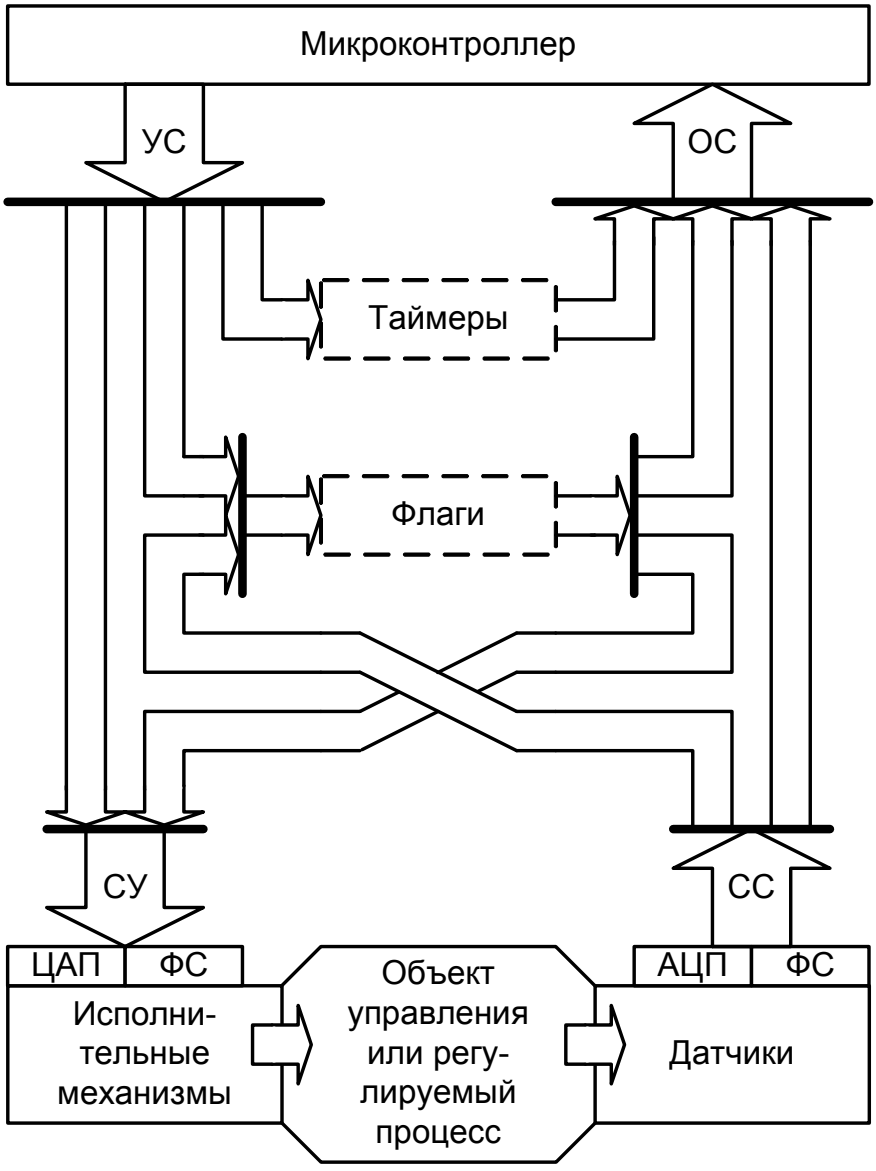

Рис. 1.

Микроконтроллер с требуемой периодичностью обновляет управляющие слова на своих выходных портах. Некоторая часть управляющего слова интерпретируется как совокупность прямых двоичных сигналов управления (СУ), которые через схемы формирователей сигналов (усилители мощности, реле, оптроны и т.п.) поступают на исполнительные механизмы (ИМ) и устройства индикации. Другая часть управляющего слова представляет собой упакованные двоичные коды, которые через цифро-аналоговые преобразователи (ЦАП) воздействуют на исполнительные механизмы аналогового типа. Если объект управления использует цифровые датчики и цифровые исполнительные механизмы, то наличие ЦАП и АЦП в системе необязательно.

В состав аппаратуры связи, которая как правило, строится на интегральных схемах серии ТТЛ, входит регистр флагов, на котором фиксируется некоторое множество специфицируемых признаков как объекта управления, так и процесса работы контроллера. Этот регистр флагов используется в качестве аппаратурного средства реализации механизма взаимной синхронизации относительно медленных и вероятностных процессов в объекте управления и быстрых процессов в контроллере. Регистр флагов доступен как контроллеру, так и датчикам. Вследствие этого он является удобным местом фиксации сигналов ГОТОВ/ОЖИДАНИЕ при передачах с квитированием или сигналов ЗАПРОС ПРЕРЫВАНИЯ/ПОДТВЕРЖДЕНИЕ при взаимодействии контроллера и объекта в режиме прерывания. Если МК-система имеет многоуровневую систему прерываний, то регистр флагов содержит схему упорядочивания приоритетов.

Для аппаратурной реализации временных задержек, формирования сигналов требуемой частоты и скважности в состав аппаратуры связи включают программируемые интервальные таймеры в том случае, если их нет в составе МК или их число недостаточно. Законы функционирования МК-системы управления со структурой, показанной на рис. 1, всецело определяются прикладной программой, размещаемой в резидентной памяти программ МК. Иными словами, специализация контроллера типовой структуры на решение задачи управления конкретным объектом осуществляется путем разработки прикладных программ МК и аппаратуры связи МК с датчиками и исполнительными механизмами объекта.

4 Взаимодействие микроконтроллера с объектами управления

4.1 Ввод информации с датчиков

4.1.1 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 замыкания контакта будет иметь вид:

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

Другим частным случаем является процедура ожидания размыкания 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 склеивания процедур WAITC и WAIT0 зависит от формы импульса. Для “отрицательного” импульса ($1\rightarrow0\rightarrow1$) процедура WAITC предшествует процедуре WAIT0, для “положительного” ($0\rightarrow1\rightarrow0$) следует за ней.

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

```
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 кратковременных 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.

4.1.2 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;
2. временная задержка.

Суть первого способа состоит в многократном считывании сигнала с контакта. Подсчет удачных опросов, обнаруживших, что контакт устойчиво замкнут, ведется программным 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 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
```

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 написана для случая подключения датчика к входу T0 и программной реализации временной задержки:

```
DBNCDL:   JTO  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.

4.1.3 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 может быть реализован, если использовать вход T1 как вход счетчика событий. В аккумуляторе фиксируется число импульсов, представленное в двоичном коде (максимальное количество 255).

```
          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          ;(аккумулятор)  число 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;
2. программной реализацией временного интервала и аппаратным подсчетом числа импульсов (на внутреннем таймере/счетчике);
3. аппаратной реализацией временного интервала и программным подсчетом числа импульсов;
4. аппаратная реализация временного интервала с аппаратным подсчетом числа импульсов.

Четвертый способ подсчета импульсов требует использования двух счётчиков. 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
          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              ;(B)(A) ← число импульсов
          MOV  A, TL1
               ...                      ;выход из процедуры
```

4.1.4 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.

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
```

При опросе двоичных датчиков передачу управления удобно осуществлять по таблице переходов. Ниже приводится текст программы, осуществляющей передачу управления одной из восьми прикладных программ PROG0 - PROG7. Передача производится в зависимости от кодовой комбинации на входах P1.0 - P1.2:

```
GOCODE:   MOV  R0, #LOW BASE  ;загрузка в R0 начального адреса
                              ;таблицы переходов
          IN   A, P1          ;ввод байта
          ANL  A, #00000111B  ;выделение младших бит
          ADD  A, R0          ;формирование адреса строки
                              ;в таблице переходов
          MOV DPH,#0;
          MOV DPL,#0;
          JMP @A+DPTR              ;передача управления
BASE:     DB   LOW PROG0      ;таблица переходов
          ...
          DB   LOW PROG7
```

Программа опрашивает и выделяет сигналы от трех датчиков путем маскирования старших битов аккумулятора.

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.

При работе с группой датчиков часто возникает необходимость осуществлять передачу управления не только в зависимости от двоичного эквивалента принятого кода, как в рассмотренном примере, но и в зависимости от соотношения принятого кода и некоторой заранее определенной уставки. Пусть, например, в порте 1 от группы двоичных датчиков формируется восьмибитный двоичный код. Если код равен десятичному эквиваленту 135, то необходимо передать управление программе с меткой LABELA, в противном случае – программе с меткой LABELB:

```
          MOV  A, #135        ;загрузка уставки
          CJNE A, P1, LABELB  ;сравнение и передача управления
LABELA:   ...
```

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

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

```
KBRD:     MOV  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:     MOV  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 может быть выполнено методом сдвигов исходного унитарного кода и подсчетом числа сдвигов на счетчике до появления первого переноса [2].

4.2 Вывод управляющих сигналов

4.2.1 Формирование статических сигналов

Для управления исполнительным устройством, работающим по принципу включено/выключено, на соответствующей выходной линии порта необходимо сформировать статический сигнал 0 или 1, что реализуется командами вывода непосредственного операнда, содержащего в требуемом бите значение 0 или 1.

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. Команда ANL используется для сброса тех битов, которые в маске заданы нулём. Команда ORL используется для установки битов управляющего слова. Командой XRL осуществляется инверсия бита.

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.

4.2.2 Формирование импульсных сигналов

Импульс можно получить последовательной выдачей сигналов включить и отключить с промежуточным вызовом подпрограммы временной задержки:

```
PULS:                          ;выдача импульса в линию 3 порта 1
ON:       ANL  P1, #11110111B ;включение
          ALL  DELAY          ;временная задержка
OFF:      RL   P1, #00001000B ;отключение
     ...
```

Длительность импульса определяется временной задержкой, реализуемой подпрограммой DELAY.

Генерация меандра. В этом случае можно воспользоваться процедурой выдачи импульса PULS и подпрограммой задержки, равной половине периода сигнала DLYX:

```
MEANDR:
XCOR:     CPL       1.3
          ACALL     LYX
          SJMP      XCOR
```

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.

4.3 Реализация функций времени

4.3.1 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 рабочего 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, реализующей временную задержку, требуется оформить 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     ;(R2) ← X
COUNT:    DJNZ R2, COUNT  ;декремент R2 и цикл,
                          ;если не нуль
          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 указывается, за сколько машинных циклов (МЦ) исполняется каждая команда. На основании этих данных определяется суммарное число машинных циклов в подпрограмме: CALL - 2 МЦ, MOV - 1 МЦ, DJNZ - 2 МЦ, RET - 2 МЦ.

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) = 5+3 * 255 = 770 мкс (т.е. почти в полтора 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, реализующую временную задержку 100 мс:

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

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.

Временная задержка длительностью 1 с. Из рассмотренного 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:

```
ONESEC:   MOV  R3, #10 ;загрузка в R3 числа вызовов подпрограмм
LOOP:     CALL DELAY     ;задержка 100 мс
          DJNZ R3, LOOP  ;декремент R3 и цикл, если (R3)≠0
```

Погрешность подпрограммы 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.

4.3.2 Формирование временной задержки таймером

Задержка малой длительности. 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 аппаратурные средства позволяют реализовать временные задержки на фоне основной программы работы.

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

В качестве примера рассмотрим организацию временной задержки длительностью 50 мс. Предполагается, что бит IE.7 установлен.

```
;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 подпрограммы обработки прерывания

     ORG  100H      ;начальный адрес программы
     MOV  TMOD, #01H;настройка T/C0
     MOV  TL0, #LOW(NOT(50000-1))  ;загрузка таймера
     MOV  TH0, #HIGH(NOT(50000-1))
     SETB TCON.4    ;старт T/C0
     SETB IE.1      ;перевод в режим холостого хода
     SETB PCON.0
NEXT:     ...
```

4.3.3 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.

Простейшим способом измерения длительности импульса является программный. 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 будет иметь вид:

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

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 будет выглядеть так:

```
          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. Прерывания 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.

5 Программы взаимодействия микроконтроллера с объектами управления

Для дальнейшей работы в лаборатории подготовьте к отладке следующие программы (язык программирования C++):

1. подпрограмма обработки внешнего прерывания,
2. программа ожидания импульсного сигнала,
3. программа формирования временной задержки программным способом,
4. программа формирования временной задержки с помощью таймера,
5. программа подсчёта числа импульсов между двумя событиями,
6. программа подсчёта числа импульсов за заданный промежуток времени на основе двух таймеров/счётчиков,
7. программа опроса группы двоичных датчиков с передачей управления подпрограммам,
8. программа опроса группы импульсных датчиков,
9. программа генерации импульсного сигнала,
10. программа работы с последовательным портом,
11. программа измерения временных интервалов программным способом,
12. программа измерения временных интервалов на основе таймера.

Произведите отладку разработанных программ в Keil uVision.

6 Пример программы реализации ввода/вывода сигналов в реальном масштабе времени для учебно-лабораторного стенда SDK-1-1

Для реализации функций ввода/вывода сигналов в учебно-лабораторном стенде SDK-1-1 можно использовать уже разработанные пользовательские функции. Данные функции находятся в файлах по следующему пути:
L:\Study\МПТ\SDK_1_1\EXAMPLES

Ниже приведён пример программы для учебно-лабораторного стенда SDK-1-1, который производит формирование линейно нарастающего напряжения на выходе ЦАП0, преобразование входного напряжения на 0-канале АЦП, а также передачу полученной информации по интерфейсу RS-232.

Пример 1

```
#include <ADuC812.h>
#include <stdio.h>
#include "ADC.h"
#include "DAC.h"
#include "max.h"
void main(void)
{
     float in,out;
     unsigned int i;

     //-----------Настройка UART ---------------
     SCON = 0x40;
     REN = 1; /* Разрешение приёма */
     // ------ Настройка таймера 2
     RCLK = 1;
     TCLK = 1; /* Тактовые импульсы от таймера 2 */
     RCAP2L = 0xDC;
     RCAP2H = 0xFF; /* Скорость 9600 бит/сек  */
     TR2 = 1; /*запуск таймера 2 */
     ES = 0;   /*  Запрещение прерывания от uart */
     TI = 1; RI = 1;
     EA = 0;
     //-----------------------------------------


     SwitchDAC(0,1);
     InitDAC(0);
     SetVoltage(5,0);

     InitADC();

     while(1)
     {

          out = out + 0.1;          //Формирование линейно
                                    //нарастающего
          if(out>=5) out = 0;       //напряжения
          SetVoltage(out,0);        //Вывод на ЦАП0

          for(i=0;i<20000;i++);     //Программная задержка
```

```
            in = GetVoltage(0);       //Запуск АЦП на 1-ое преобр 0-
                                      //канала
            in = in * 1.08;           //Нормирование кода АЦП с
                                      //учётом погрешности входного
                                      //делителя
            printf("DAC0 = %f",out);//Передача по RS-232
            printf(" : ADC0 = %f\n",in);
      }
}
```

Для сборки примера 1 в папку проекта необходимо добавить модули ADC.C, DAC.C, MAX.C, а также их заголовочные файлы ADC.H, DAC.H, MAX.H. При этом перечисленные модули необходимо включить в проект.

Программный симулятор uVision Debug позволяет моделировать ЦАП и АЦП микроконтроллера ADuC812. Для этого, в режиме отладки, необходимо зайти в меню *Peripherals*, далее открыть *A/D Converter* и *D/A Converter* (см. рис. 2).

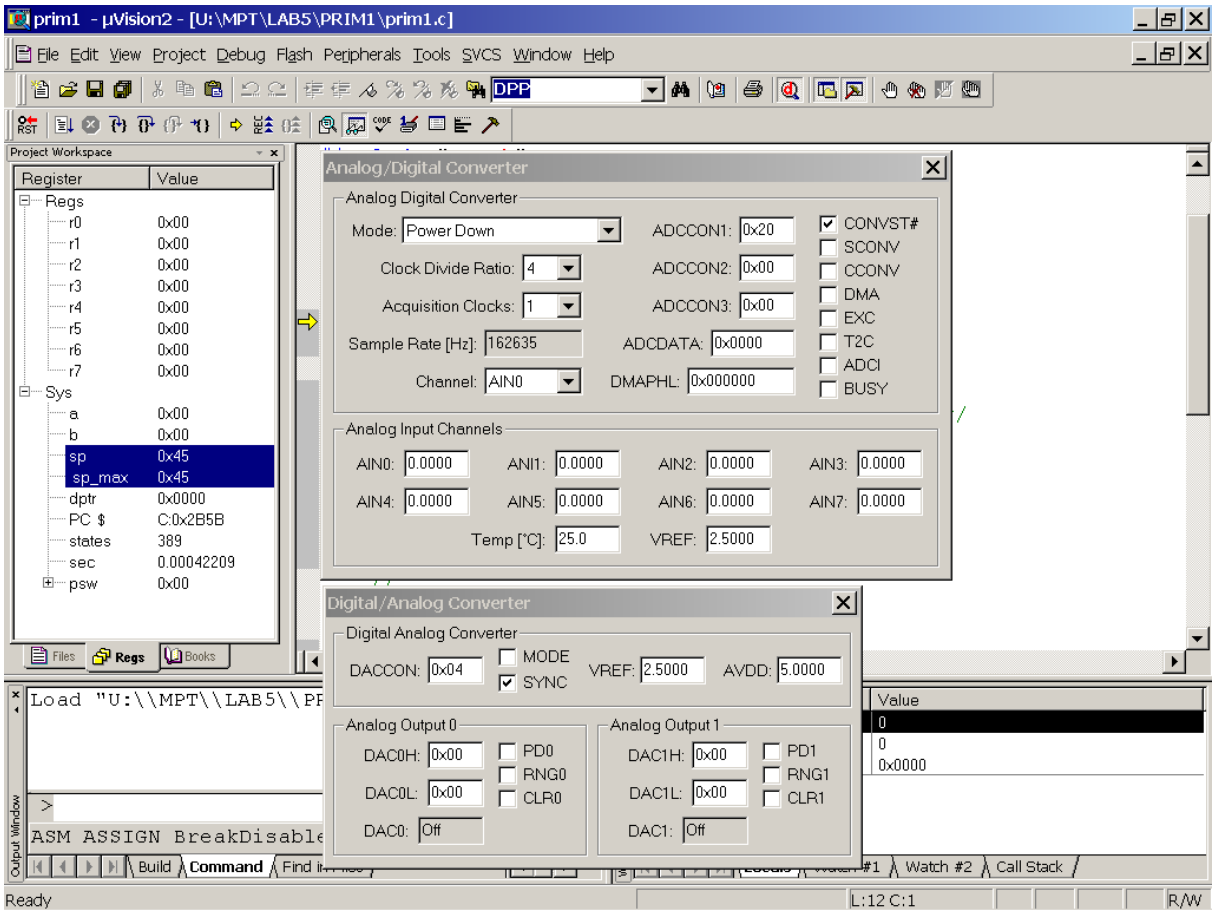

Рис. 2.

Данные окна (см. рис. 2) позволяют наблюдать текущие состояния АЦП и ЦАП микроконтроллера.

Задание для примера 1:

1. Разберитесь с программой, допишите не достающие комментарии (включая модули ADC.C и DAC.C).
2. Произведите отладку программы в uVision, проверьте её работоспособность на учебно-лабораторном стенде.

Далее необходимо доработать программу следующим образом:

1. Синхронизацию формирования линейно нарастающего напряжения на ЦАП0 и программный запуск АЦП выполнить от таймера 0.
2. Чтение кода АЦП реализовать в виде обработчика прерывания.
3. Задержку между записью в регистр ЦАП и запуском АЦП осуществить при помощи таймера 1.
4. Обмен информацией по интерфейсу RS-232 выполнить через обработчик прерывания.

7 Индивидуальное задание

Вариант индивидуального задания указывается преподавателем.

8 Содержание отчёта

1. Цель работы.
2. Структура МК-системы управления с краткими комментариями.
3. Структура ЦАП на основе R-2R-матрицы.
4. Классификация АЦП.
5. Структура АЦП последовательного приближения (с кратким описанием).
6. Результаты выполнения заданий

Листинг доработанной программы ввода/вывода сигналов с комментариями и блок-схемой.

Листинг программы индивидуального задания с комментариями. Блок-схема программы.

7. Выводы по проделанной лабораторной работе.

9 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. Как инвертировать отдельные биты портов?
7. С какой частотой инкрементируется содержимое таймера/счётчика при работе в качестве таймера?
8. Чему равна максимальная частота подсчёта входных сигналов при работе таймера/счётчика в режиме счётчика?
9. Охарактеризуйте режимы работы таймера-счётчика.
10. Как с помощью таймера можно измерить длительность импульсного сигнала?
11. Охарактеризуйте режимы работы последовательного порта.
12. Для чего предназначен регистр SCON?
13. Поясните принцип работы UART в мультиконтроллерных системах.
14. Как изменить скорость передачи данных через последовательный порт?
15. Для чего используется девятый бит?
16. Нарисуйте схему прерываний. Перечислите и охарактеризуйте типы прерываний.
17. Для чего нужен регистр масок прерывания? Как изменить приоритеты прерываний?
18. Чем отличаются команды RET и RETI?
19. Перечислите команды операций с битами.
20. Как организовать процедуру ожидания события с помощью одной команды?
21. Укажите, какие из регистров специальных функций допускают битовую адресацию.
22. Перечислите средства программы uVision, предназначенные для отладки взаимодействия микроконтроллера с объектами управления.
23. Какие ограничения накладываются на длительность обнаруживаемого импульсного сигнала при программной реализации цикла ожидания?
24. Поясните принципы устранения дребезга контактов.
25. Поясните принцип организации процедур подсчёта числа импульсов между двумя событиями и за заданный промежуток времени.
26. В чём заключается табличный способ генерации микроконтроллером сложных последовательностей управляющих сигналов?
27. Поясните принцип генерации периодических и апериодических сигналов.
28. Как программно формируются задержки разной длительности?
29. Как с помощью микроконтроллера измерить временной интервал? Как оценить точность измерения?
30. Описать структуру ЦАП на основе R-2R-матрицы.
31. Классификация АЦП.
32. Структура АЦП последовательного счёта.
33. Структура АЦП последовательного приближения.

Перечень источников

1. Горюнов А.Г. Ливенцов С.Н. АЦП.
L:\Study\МПТ\Методички\метАЦП.pdf
2. Сташин В.В., Урусов А.В., Мологонцева О.Ф. Проектирование цифровых устройств на однокристальных микроконтроллерах. М.: Энергоатомиздат, 1990. – 224 с.