MicroConverter™ Техническое замечание - µC001
Использование I²C интерфейса ADuC812

Вступление

Это техническое замечание описывает аппаратную и программную реализацию I²C канала с использованием I²C интерфейса встроенного в кристалл ADuC812. I²C канал описан как простое соединение ведущего (master) и ведомого (slave), представленное на рис.1.

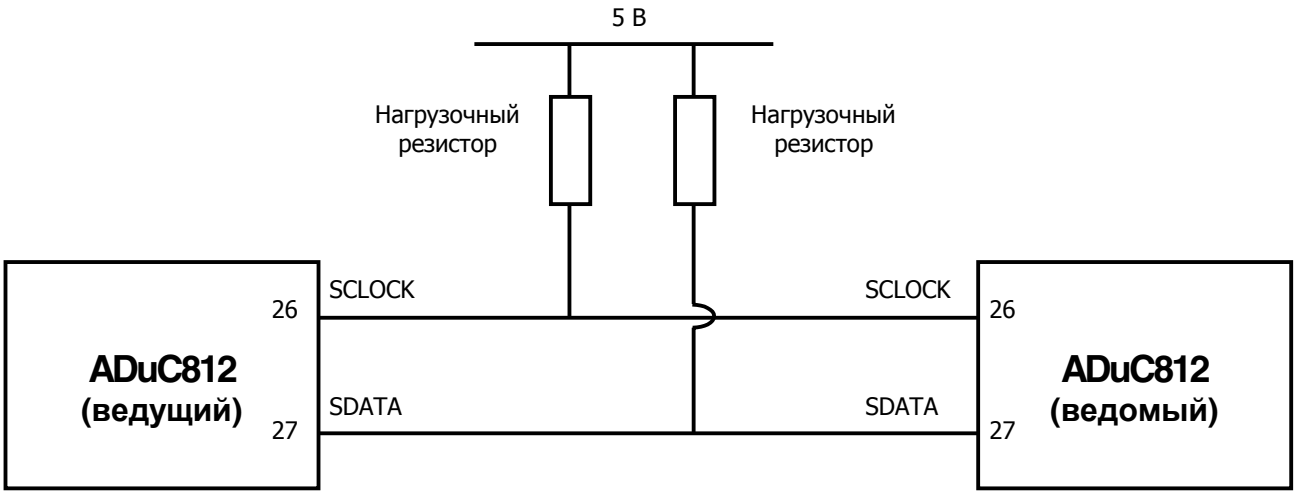


Рис.1. Структурная схема I²C

Оригинал : версия 1.0 [05/1999]
Перевод : версия 1.0 [12/1999]

Техническое замечание µ0001

Обзор интерфейса I²C

I²C (Inter Integrated Circuit) является двухпроводной последовательной системой связи, разработанной Philips, которая позволяет устанавливать соединение между несколькими ведущими и несколькими ведомыми устройствами при помощи двух линий (SCLOCK, SDATA). Сигнал SCLOCK управляет передачей данных между ведущим и ведомым. Сигнал SDATA используется для передачи и приема данных. Обе линии двунаправлены. Скорость передачи управляется линией SCLOCK. При I²C интерфейсе есть как минимум один ведущий и один ведомый, хотя I²C поддерживает несколько ведущих и ведомых. Ведущее устройство генерирует тактовые импульсы, в то время как ведомое ими управляется. Типичная последовательность передачи данных приведена на рис.2.

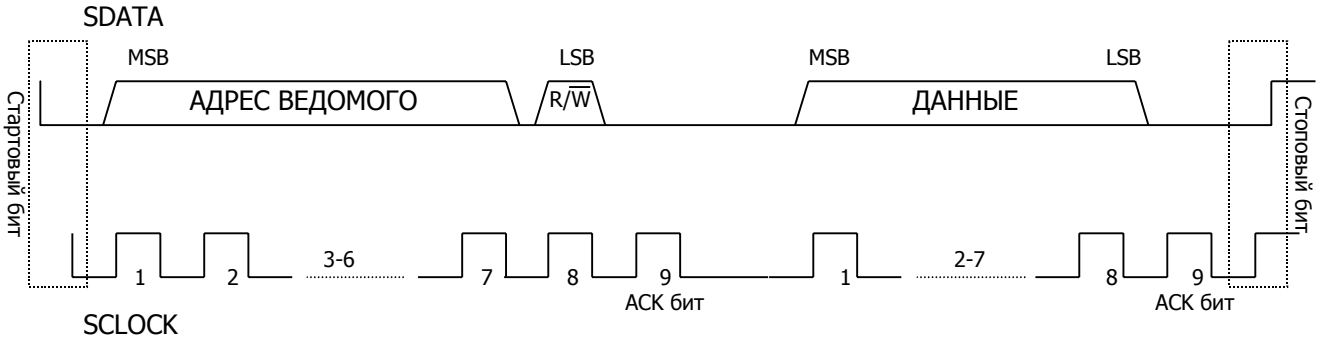


Рис.2. Типичная последовательность передачи I²C

Эта последовательность начинается стартовым битом, который генерирует ведущий. Стартовое состояние выглядит как изменение уровня сигнала SDATA от высокого к низкому, в то время как уровень сигнала SCLOCK высокий. Стартовое состояние показано на рис.3.

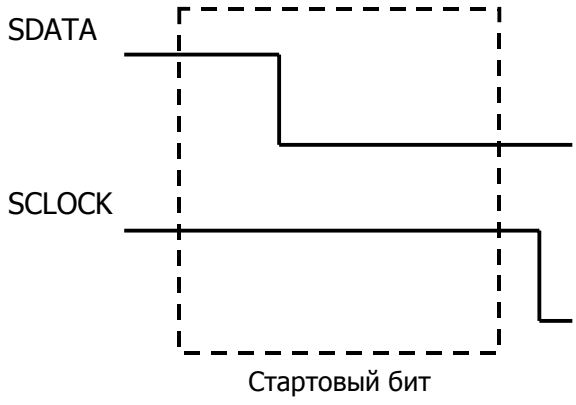


Рис.3. Стартовое состояние I²C

После стартового состояния ведущий посылает байт (в начале старшую часть) по линии SDATA, который содержит адрес ведомого бит статуса R/W (чтение/запись). Первые семь разрядов образуют адрес ведомого. Восьмой разряд, содержащийся в младшей части байта, определяет направление сообщения (см. рис.4). «Ноль» означает, что ведущий будет передавать данные выбранному ведомому. «Единица» в этом разряде означает, что ведущий будет принимать данные от ведомого. Эти операции будут выполнены только после правильного бита подтверждения (ACK), принятого первым от ведомого.

Рис.4. Первый байт после стартового условия

Когда ведущий посылает адрес, каждый ведомое устройство в системе сравнивает первые семь разрядов после стартового состояния со своим собственным адресом. Если они совпадают, ведомое устройство считает, что ведущий обращается к нему и отвечает посылкой бита подтверждения (см. рис.5). Подтверждение выглядит как низкий уровень сигнала SDATA на девятом такте и должно передаваться ведомым в конце каждого байта передачи.

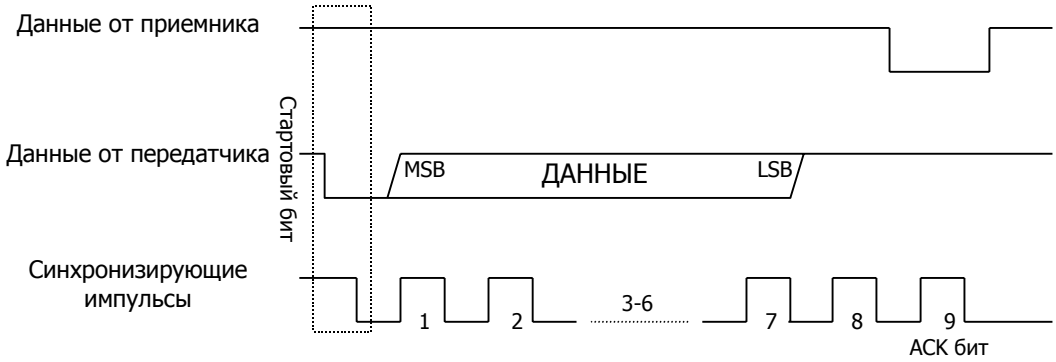


Рис.5. Подтверждение I²C

Если нет подтверждения или передача не закончилась, ведущее устройство генерирует состояние завершения, определяемое изменением уровня сигнала SDATA от низкого к высокому, в то время как уровень сигнала SCLOCK высокий (см. рис.6).

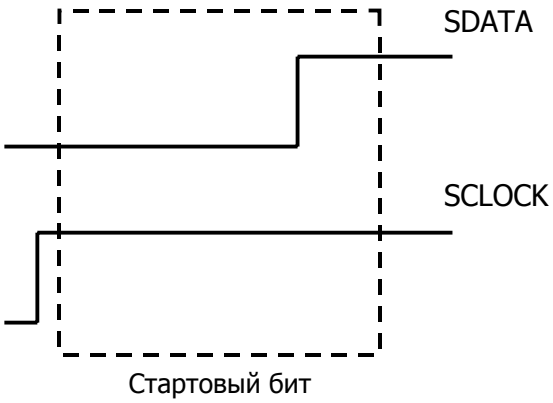


Рис.6. Состояние завершения I²C

Реализация I²C на ADuC812

Эта часть описывает реализацию I²C на ADuC812 MicroConverter. I²C интерфейс ADuC812 обеспечивает оба режима работы: аппаратный ведомого и программный ведущего. SCLOCK и SDATA являются 26 и 27 контактом соответственно. При включении или сбросе, I²C устанавливается в режим ведомого.

Три регистра SFR используются для управления интерфейсом:

I2CADD
Содержит 7-разрядный адрес устройства ADuC812 на шине (по умолчанию = 55H).

Процедура расчета 7-разрядного адреса:
Если ведомое устройство содержит в регистре I2CADD значение 44H, ведущие устройство для открытия связи с ним должно выслать значение 88H. Из 7-разрядности адреса ведомый автоматически знает, что самый младший разряд (LSB) это разряд статуса чтения/записи. Поэтому, ведомый сравнивает только семь разрядов байта со своим адресом. Для достижения полного байта, ведомый добавляет ноль к самому старшему разряду (MSB). Результат этой операции и сравнивается с собственным адресом (см. рис.7).

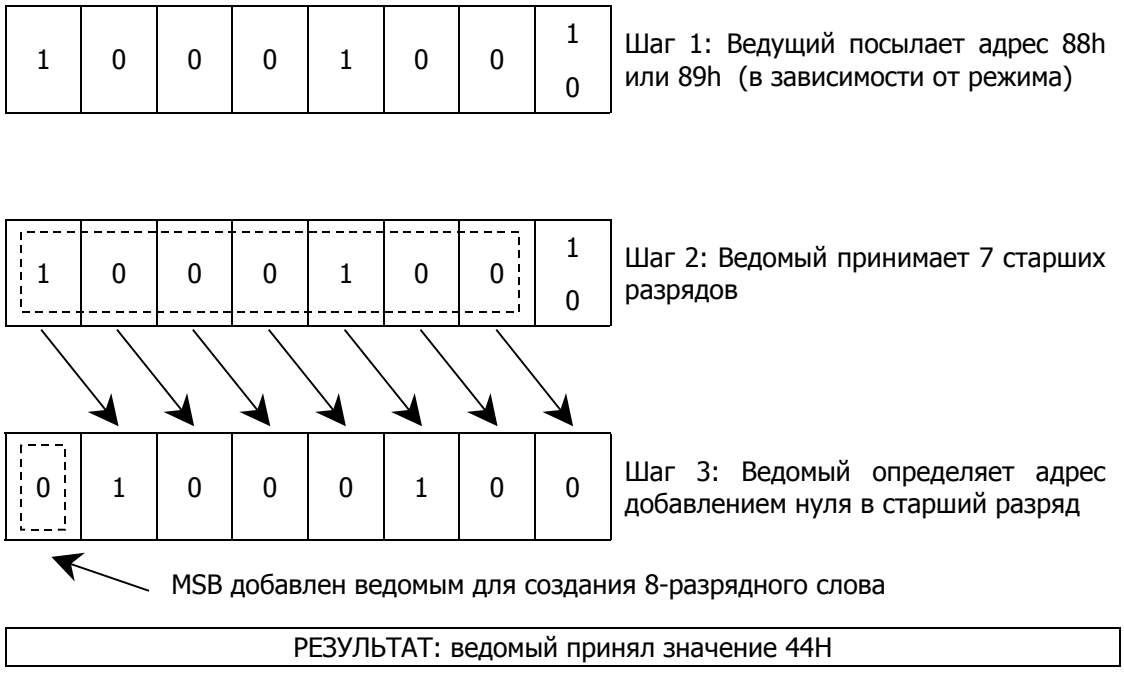


Рис.7. Процедура получения адреса ведомым устройством

I2CDAT
Содержит 8 разрядов для принимаемых и передаваемых данных.

I2CCON
Содержит конфигурационные/управляющие разряды для режима ведущего/ведомого (см. рис.8). Ведомый использует три младших разряда регистра I2CCON. Так как реализация режима ведомого устройства аппаратная, то ведомый автоматически определяет стартовый бит, бит подтверждения, прерывание или стоповый бит. Ведущий использует четыре старших разряда регистра I2CCON для управления генерацией сигналов ведущего на контакты SCLOCK и SDATA. Поэтому, в программной реализации ведущего, пользователь должен генерировать оба сигнала SCLOCK и SDATA программно с использованием адресуемых расположений. Например, разряд, который управляет сигналом SCLOCK (26) расположен в побитно адресуемом пространстве (MCO) в регистре I2CCON. Ниже приведен пример, который вызывает непрерывный высокий-низкий-высокий импульс на контакте SCLOCK:

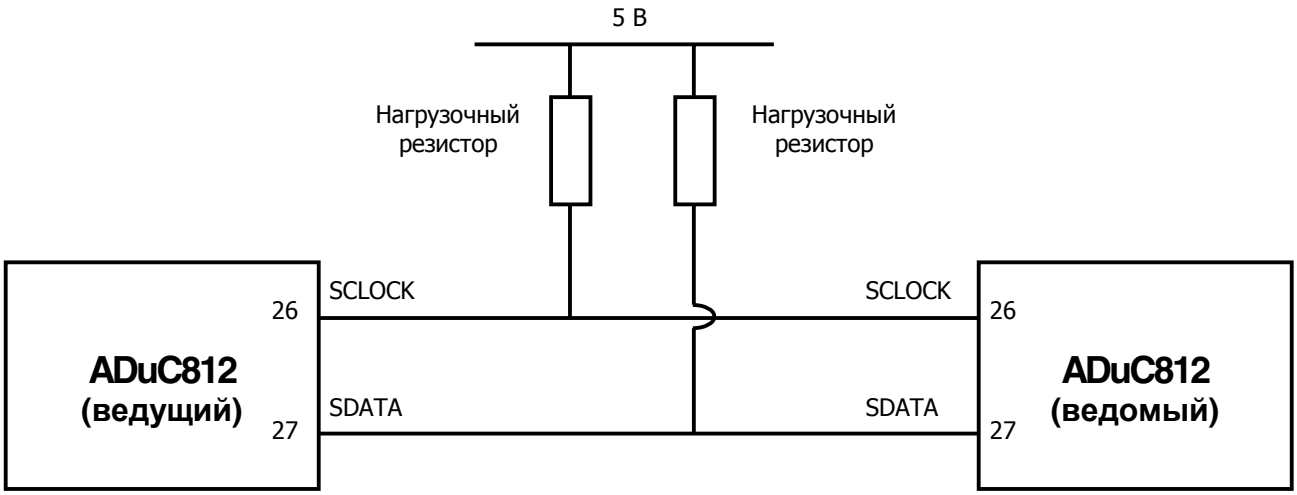
```
AGAIN:    SETB MCO
          CLR MCO
          JMP AGAIN
```

MDO	MDE	MCO	MDI	I2CM	I2CRS	I2CTX	I2CI

MDO - Master Data Output (вывод данных ведущего)
MDE - Master Data Enable (разрешение данных ведущего)
MCO - Master Clock Output (вывод SCLOCK)
MDI - Master Data Input (вывод данных ведомого)
I2CI - I2C Interrupt bit (прерывание I2C)
I2CTX - Direction status (статус направления)
I2CRS - I2C reset bit (сброс I2C)
I2CM - Master mode select (режим работы ведущего)

Рис.8. Описание разрядов регистра I2CCON

Связь I²C (Ведущий-передатчик и ведомый-приемник)



В этом режиме ведущий передает и адрес ведомого, и при получении правильного подтверждения передает три байта (передача прием трех байтов используется в примере, приведенном в этом техническом замечании) перед завершением связи передачей стопового бита.
Когда есть два ADuC812, необходимы две разные программы, одна для ведущего, другая для ведомого. На следующих страницах описаны блок-схемы, соответствующие программам ведущего и ведомого в этом режиме связи.

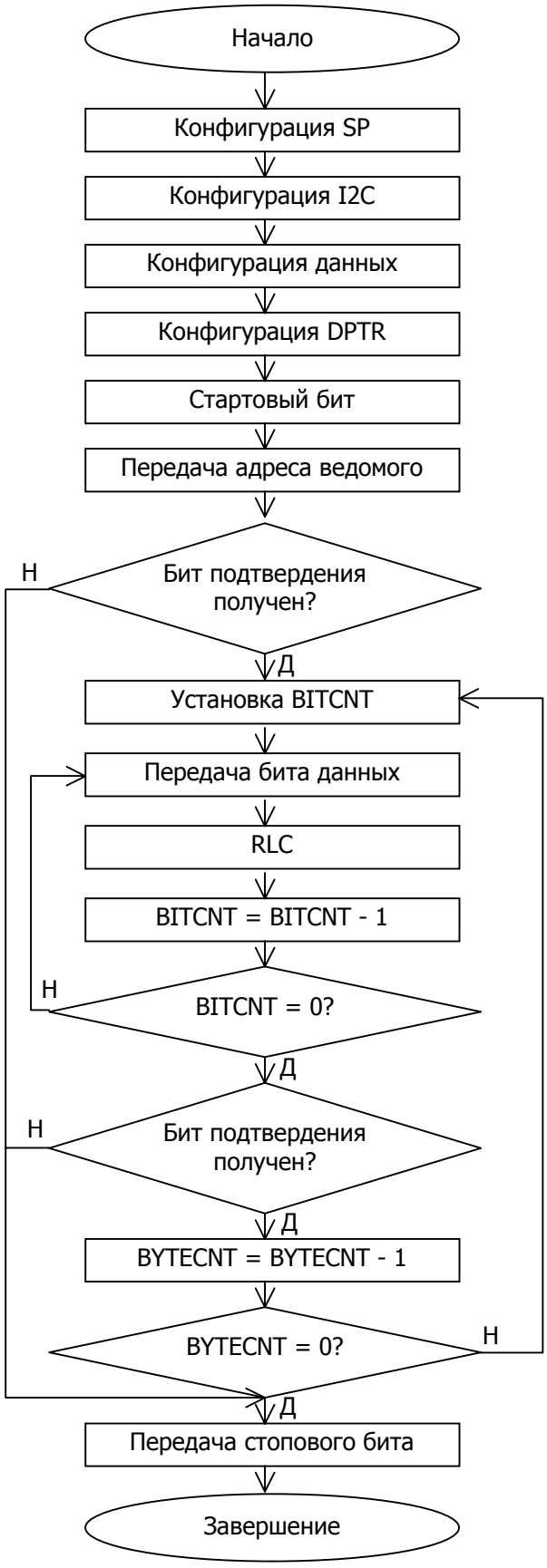
Последовательность действий ведущего:

- Передает стартовый бит
- Передает адрес ведомого
- Ожидает бит подтверждения
- Передает первый байт данных
- Ожидает бит подтверждения
- Передает второй байт данных
- Ожидает бит подтверждения
- Передает третий байт данных
- Ожидает бит подтверждения
- Передает стоповый бит

Последовательность действий ведомого:

- Принимает адрес
- Передает бит подтверждения
- Принимает первый байт данных
- Передает бит подтверждения
- Принимает второй байт данных
- Передает бит подтверждения
- Принимает третий байт данных
- Передает бит подтверждения

Блок-схема



Программа ведущего:

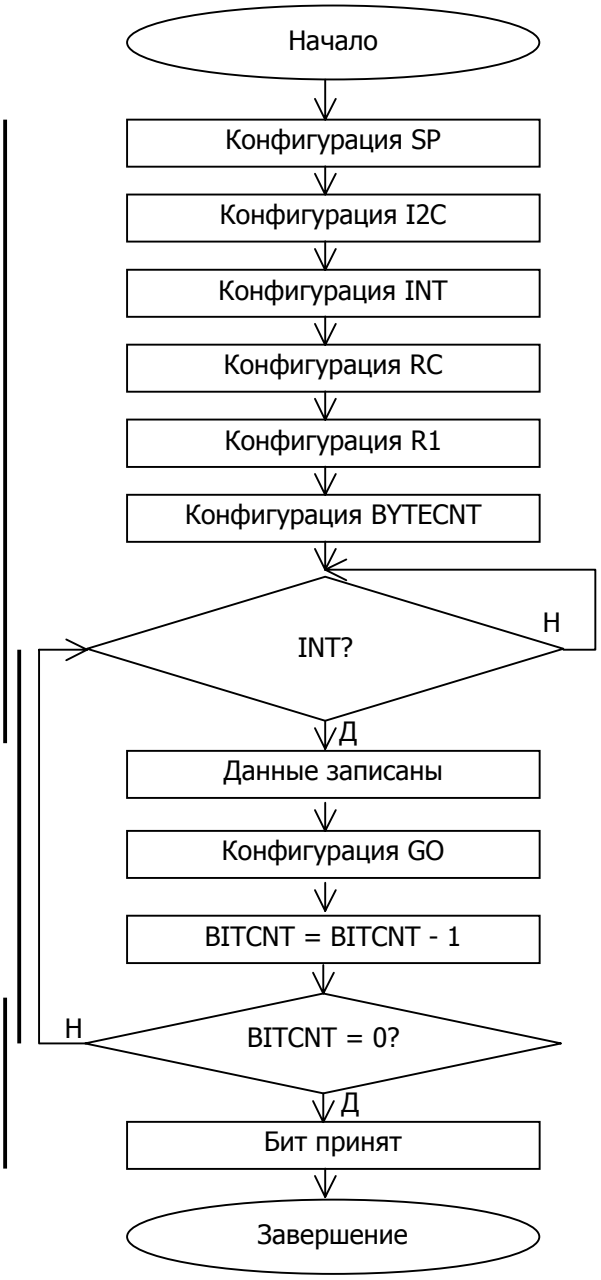
Блок-схема слева описывает все операции, которые происходят в этом режиме. Ведущий, который является передатчиком, передает данные ведомому. В этом режиме направление передачи не изменяется (ведущий передает адрес и последующие данные).

1. В программе, после того как сконфигурированы SFR регистры I2C, ведущий передает стартовый бит по линии SDATA. В этом режиме R/W статус сброшен («ноль»). Если ведущий не получает бит подтверждения от ведомого, то он передает стоповый бит, устанавливается бит ошибки и передача прекращается.

2. Если ведомый подтверждает прием, ведущий передает данные, которые были предварительно записаны во внешнюю память на макетной плате ведущего. После того как байт данных был отослан, ведомый должен подтвердить его прием. Если это происходит, то ведущий передает следующий байт данных. Если в какой-то момент подтверждения ведомого не происходит, ведущий передает стоповый бит, устанавливается бит ошибки и передача прекращается.

3. Когда счетчик BYTECNT становится равным '0', это означает, что был передан последний байт (всего три в этом примере), ведущий передает стоповый бит, означающий окончание передачи.

Блок-схема



Программа ведомого:

Блок-схема слева описывает все операции, которые происходят в этом режиме. Ведущий, который является передатчиком, передает данные ведомому. В этом режиме направление передачи не изменяется (ведущий передает адрес и последующие данные).

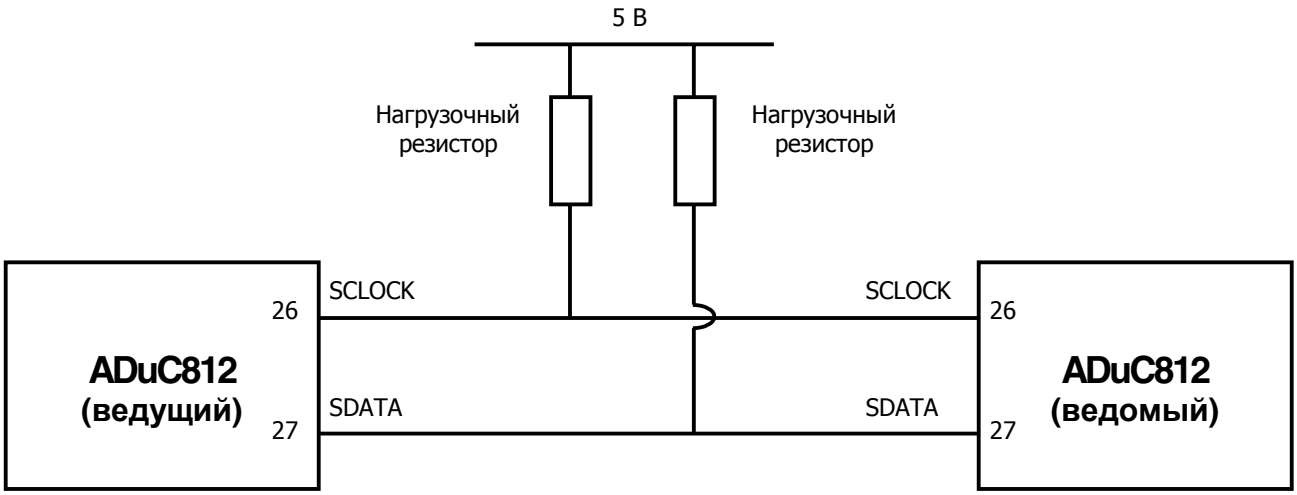
1. В программе, после того как SFR регистры I2C были сконфигурированы и принят стартовый бит, который был передан ведущим, ведомый ждет первого байта данных (приход данных генерирует прерывание). Как только он получен, ведомый сравнивает данные со своим адресом. Если они совпадают, ведомый передает бит подтверждения по линии SDATA и, как только установится статус R/W, начинает ждать данные.

2. Если ведущий передает данные, то ведомый сохраняет их во внутренней памяти, подтверждает прием и ожидает следующий байт данных (приход подпоследовательности данных также генерирует прерывание).

3. Когда ведомый получает последний байт передачи (всего три в этом примере), BYTECNT становится равным нулю. С этого времени ведомый ждет стоповый бит. Как только он приходит, канал I2C автоматически закрывается.

Примечание:
Когда в программе происходит прерывание, бит прерывания (I2CI) автоматически устанавливается в «единицу», но пользователь должен очистить его в подпрограмме прерывания (см. строку CLR I2CI в примере). Если бит не будет равен «нулю», ведомый будет держать сигнал линии SCLOCK в низком уровне, что сделает невозможным дальнейшую I2C передачу.

Связь I²C (Ведущий-приемник и ведомый-передатчик)



В этом режиме, ведущий передает адрес ведомого и при получении бита подтверждения, ожидает приема от ведомого трех байтов (в этом примере) данных, перед завершением связи передачей стопового бита.
Когда есть два ADuC812, необходимы две разные программы, одна для ведущего, другая для ведомого. На следующих страницах описаны блок-схемы, соответствующие программам ведущего и ведомого в этом режиме связи.

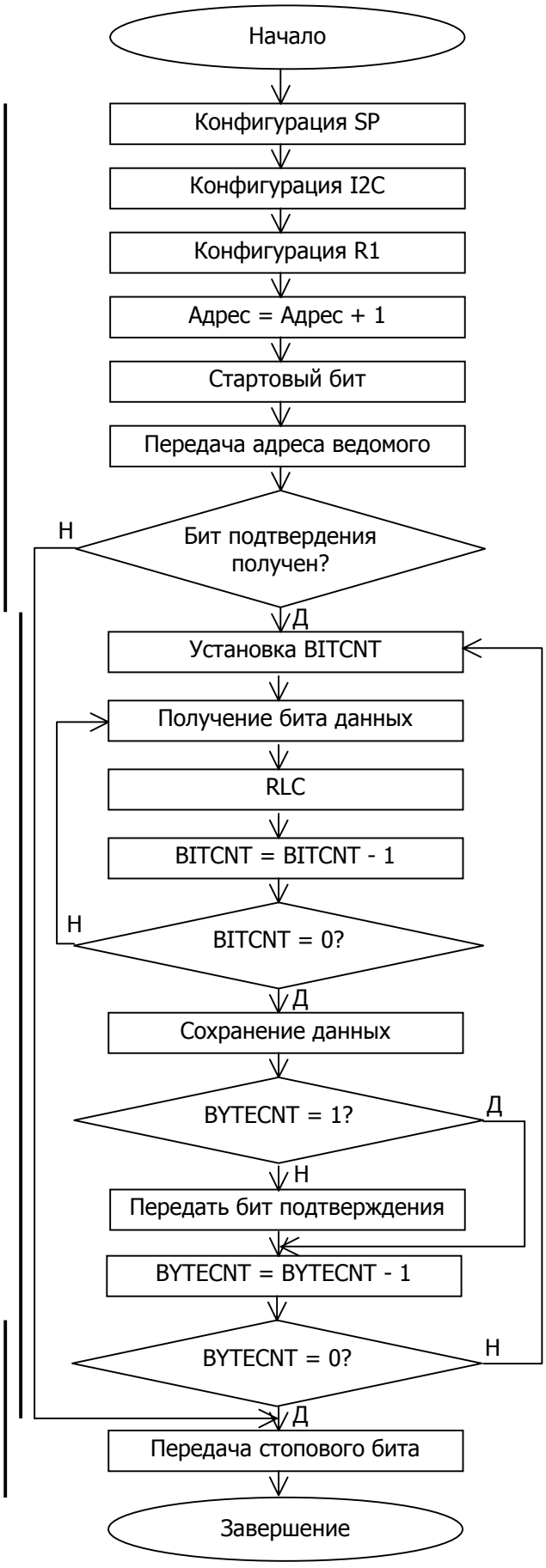
Последовательность действий ведущего:

- Передает стартовый бит
- Передает адрес ведомого
- Ожидает бит подтверждения
- Принимает первый байт данных
- Передает бит подтверждения
- Принимает второй байт данных
- Передает бит подтверждения
- Принимает третий байт данных
- Передает бит подтверждения
- Передает стоповый бит

Последовательность действий ведомого:

- Принимает адрес
- Передает бит подтверждения
- Передает первый байт данных
- Ожидает бит подтверждения
- Передает второй байт данных
- Ожидает бит подтверждения
- Передает третий байт данных
- Ожидает бит подтверждения

Блок-схема



Программа ведущего:

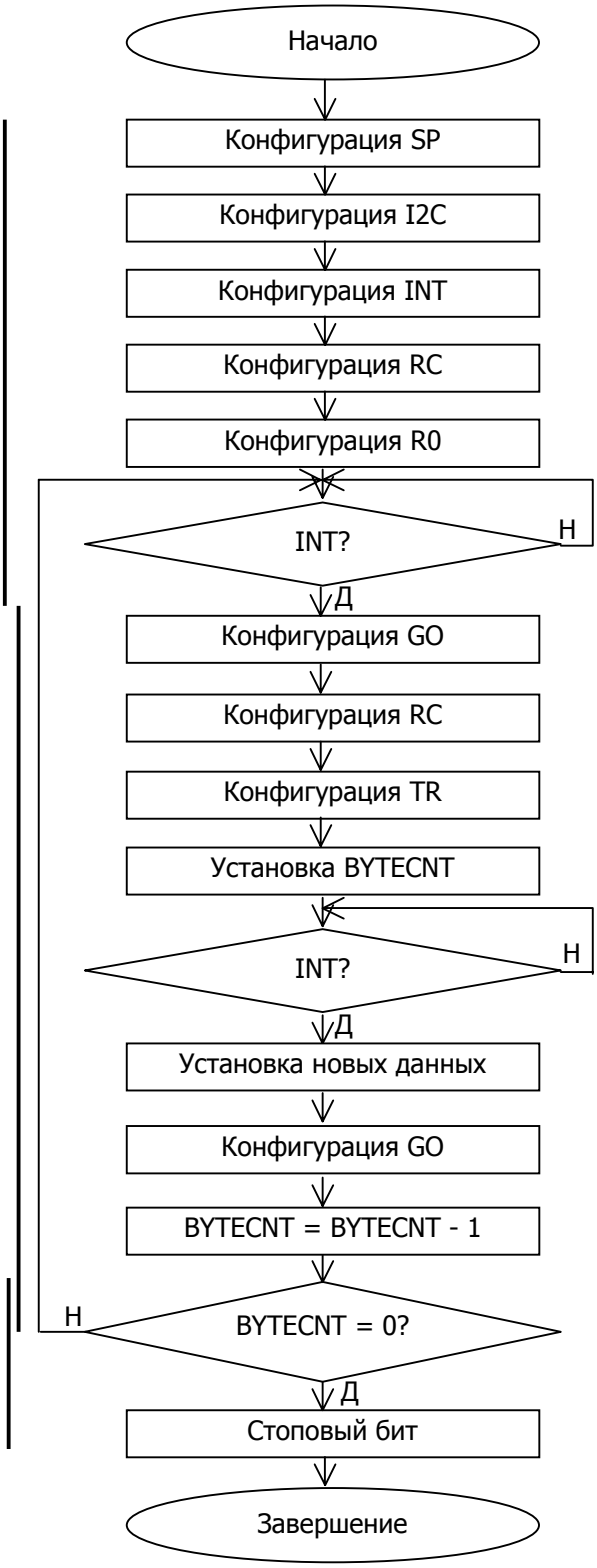
Блок-схема слева описывает все операции, которые происходят в этом режиме. Ведущий считывает данные ведомого сразу после получения первого байта. В этом режиме направление передачи меняется. Сначала ведущий передает ведомому адрес, а затем получает три последовательных байта данных.

1. В программе, после того как SFR регистры I2C были сконфигурированы, ведущий передает стартовый бит. Далее ведущий передает адрес ведомого. В этом режиме бит статуса чтения/записи установлен. Если ведущий не получает бит подтверждения от ведомого, он передает стоповый бит, устанавливается бит ошибки и передача прекращается.

2. Если бит подтверждения получен, ведущий ждет передачи первого байта данных. После его получения, ведущий запоминает данные во внутренней памяти, передает подтверждение и ждет следующего байта данных.

3. Если счетчик BYTECNT равен нулю, это означает, что был получен последний байт данных. Ведущий прекращает передачу, передавая стоповый бит.

Блок-схема



Программа ведомого:

Блок-схема слева описывает все операции, которые происходят в этом режиме. Ведущий считывает данные ведомого сразу после получения первого байта. В этом режиме направление передачи меняется. Сначала ведущий передает ведомому адрес, а затем получает три последовательных байта данных.

1. В программе, после того как SFR регистры I2C были сконфигурированы, и был принят стартовый бит, переданный ведущим, ведомый ждет первый байт данных (приход данных вызывает прерывание). После того, как он был принят, ведомый сравнивает данные со своим собственным адресом. Если они совпадают, ведомый передает бит подтверждения по линии SDATA.

2. После установления бита статуса чтения/записи, ведомый передает данные, которые были предварительно записаны во внутреннюю память. Когда данные переданы, ведомый ждет подтверждения приема от ведущего. После каждого правильного бита подтверждения ведомый передает следующий байт данных и вновь ждет подтверждения.

3. Когда счетчик BYTECNT равен нулю, это означает, что был передан последний байт данных. Ведущий не передает бит подтверждения, а посылает стоповый бит, который завершает передачу.

Примечание:
Когда в программе происходит прерывание, бит прерывания (I2CI) автоматически устанавливается в «единицу», но пользователь должен очистить его в подпрограмме прерывания (см. строку CLR I2CI в примере). Если бит не будет равен «нулю», ведомый будет держать сигнал линии SCLOCK в низком уровне, что сделает невозможным дальнейшую I2C передачу.

Заключение

ADuC812 объединяет в себе программную реализацию ведущего устройства и, расположенную на кристалле, аппаратную реализацию ведомого устройства интерфейса I^2C. В то время как операции I^2C в режиме ведущего требуют дополнительного программного обеспечения, как описано выше, используются одинаковые SFR регистры и идентичные внешние контакты I^2C для реализации протокола I^2C и способности поддерживать скорость передачи более 100 Кбит/с.
Хотя ранее был описан простейший пример связи ведущий/ведомый, он может быть легко расширен для поддержки нескольких ведомых (см. рис.9).

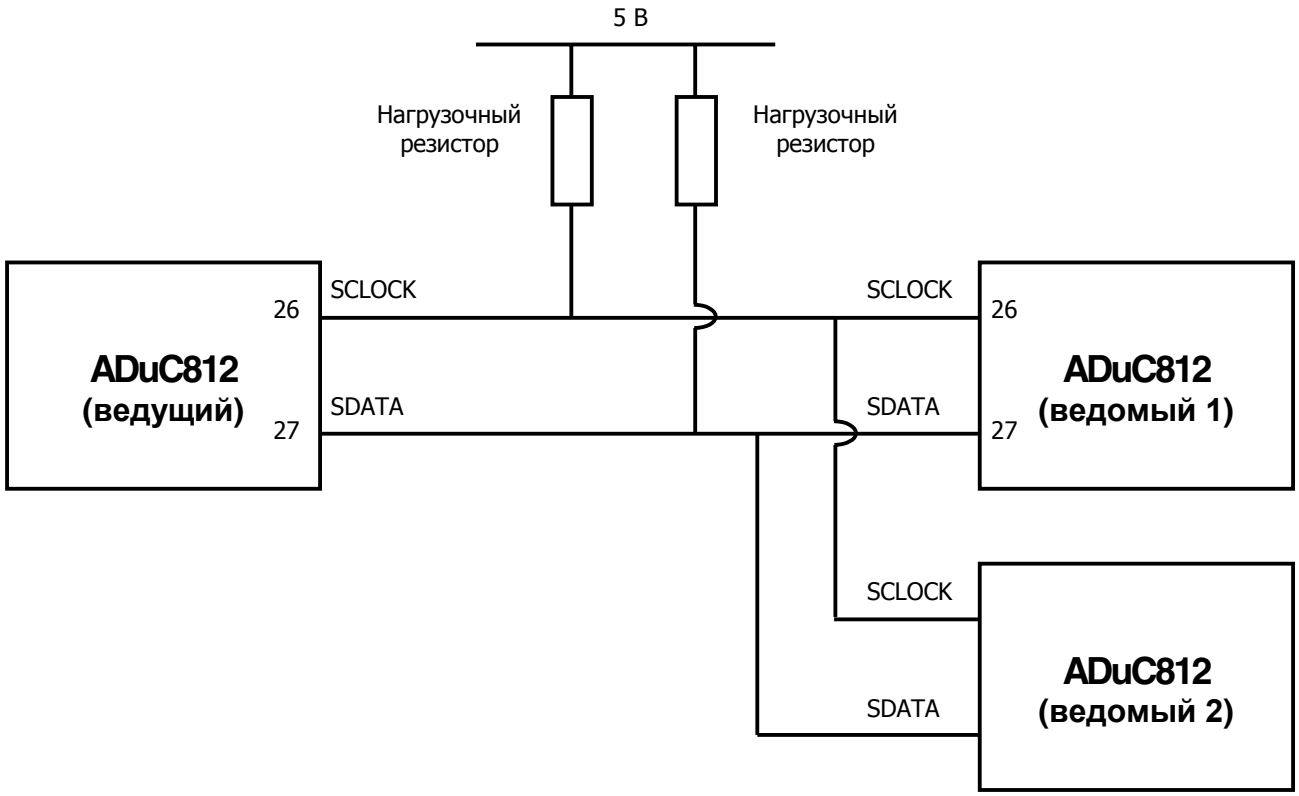


Рис.9. Пример соединения одного ведущего и двух ведомых устройств

Часто задаваемые вопросы

Вопрос:

Когда я запускаю программу обе линии и SDATA, и SCLOCK стоят на низком уровне?

Ответ:

Проверьте наличие внешних нагрузочных резисторов (3.9k) и их подключение к питанию +5В

Вопрос:

Ведущий передает адрес ведомого, но не получает от него ответа?

Ответ:

Проверьте корректность настройки регистра I2CADD в программе ведомого.

Вопрос:

В программе ведомого регистр I2CADD содержит значение 44H, ведущий передает значение 44H, но ведомый не подтверждает его?

Ответ:

Из-за 7-разрядности адреса, ведущий должен передавать значение 88H или 89H (в зависимости от режима чтения/записи). Для получения подробной информации смотрите стр.4 этого документа.

Вопрос:

Ведущий передает адрес ведомого и получает подтверждение, но ведомый периодически держит SCLOCK на низком уровне?

Ответ:

Каждый раз, когда ведомый получает или передает данные, устанавливается бит прерывания I2CI и в программе ведомого вызывается подпрограмма прерывания. В этой подпрограмме бит прерывания должен быть очищен (установлен в ноль), иначе линия SCLOCK будет находиться на низком уровне.

Вопрос:

Ведущий передает адрес, но похоже, что в программе ведомого не вызывается подпрограмма прерывания?

Ответ:

Проверьте конфигурацию регистров прерывания IE и IE2.

IE = 80H разрешение всех прерываний.

IE2 = 01H разрешение прерывания I2C.

Приложение A. Исходный текст MASTER.ASM

```
$MOD812                         ; use ADuC812 & 8052 predefined symbols

BITCNT          DATA    8h      ; bit counter for I2C routines
BYTECNT         DATA    030h    ; byte counter for I2C routines
SLAVEADD        DATA    032h    ; slave address for I2C routines

FLAGS           DATA    28h
NOACK           BIT     FLAGS.0 ; I2C no acknowledge flag
BUSY            BIT     FLAGS.1 ; I2C busy flag
ERROR           BIT     FLAGS.2 ; I2C error flag
MISTAKE         BIT     P3.4

;=====================================================================
CSEG

ORG 0000H
        JMP START
;=====================================================================

ORG 007BH                       ; Subroutines

;---------------------------------------------------------------------
; DELAY: Create a delay for the main signals ( SCLOCK , SDATA )
;---------------------------------------------------------------------
DELAY:
        NOP
        RET

;---------------------------------------------------------------------
; SENDSTOP: Send the bit stop of the transmission
;---------------------------------------------------------------------
SENDSTOP:
        SETB    MDE             ; to enable SDATA pin as an output
        CLR     MDO             ; get SDATA ready for stop
        SETB    MCO             ; set clock for stop
        ACALL   DELAY
        SETB    MDO             ; this is the stop bit
        CLR     BUSY            ; bus should be released
        RET

;---------------------------------------------------------------------
; SENDBYTE: Send a 8-bits word to the slave
;---------------------------------------------------------------------
SENDBYTE:
        MOV     BITCNT,#8       ; 8 bits in a byte

        SETB    MDE             ; to enable SDATA pin as an output
        CLR     MDO
        CLR     MCO
LOOP:   RLC     A               ; send one bit
        MOV     MDO,C           ; put data bit on pin
        SETB    MCO             ; send clock
        CLR     MCO             ; clock is off
        DJNZ    BITCNT,LOOP

        CLR     MDE             ; release data line for acknowledge
        SETB    MCO             ; send clock for acknowledge
        JNB     MDI,NEXT        ; this is a check
        SETB    NOACK           ; no acknowledge
NEXT:   CLR     MCO             ; clock for acknowledge
        RET


;---------------------------------------------------------------------
; BITSTART: Send the bit start of the transmission and the slave
;           address to the slave
;---------------------------------------------------------------------
BITSTART:
        SETB    BUSY            ; I2C is in progress
        SETB    MDE             ; to enable SDATA pin as an output
        CLR     NOACK
        CLR     ERROR
        JNB     MCO,FAULT
        JNB     MDO,FAULT
```

```
        CLR     MDO             ; this is
        ACALL   DELAY           ; the
        CLR     MCO             ; start bit
FAULT:  CLR     MISTAKE         ; set error flag
        MOV     A,SLAVEADD      ; Get slave address
        ACALL   SENDBYTE        ; call routine to send slave addr. byte
        RET

;-----------------------------------------------------------------------
; SENDATA: Send all the sequence to the slave ( slave address + data )
;-----------------------------------------------------------------------
SENDATA:
        ACALL   BITSTART
        JB      MDI,NEXT1
        MOV     A,#00
SLOOP:  MOVX    A,@DPTR
        ACALL   SENDBYTE
        INC     DPTR
        JB      NOACK,NEXT1
        DJNZ    BYTECNT,SLOOP

NEXT1:  ACALL   SENDSTOP
        MOV     A,FLAGS
        ANL     A,#07h
        JZ      RETOUR
        SETB    P3.4
        CLR     I2CRS
RETOUR: RET

;-----------------------------------------------------------------------
; RCVBYTE: receives one byte of data from an I2C slave device.
;-----------------------------------------------------------------------
RCVBYTE:
        MOV     BITCNT,#8       ;Set bit count.

        CLR     MDE             ;Data pin of the master is now an input
        CLR     MCO
LOOP2:  SETB    MCO
        CLR     MCO
        MOV     C,MDI           ;Get data bit from pin.
        RLC     A               ;Rotate bit into result byte.

        DJNZ    BITCNT,LOOP2    ;Repeat until all bits received.

                                ;result byte is in the accumulator

        PUSH    ACC             ;Save result byte in the stack

        SETB    MDE             ;Data pin of the master must be an..
                                ;..output for the acknowledge
        MOV     A,BYTECNT
        CJNE    A,#1,SACK       ;Check for last byte of frame.
        SETB    MDO             ;Send no acknowledge on last byte.
        SJMP    NACK

SACK:   CLR     MDO             ;Send acknowledge bit.

NACK:   SETB    MCO             ;Send acknowledge clock.
        POP     ACC             ;Restore accumulator
        ACALL   DELAY
        CLR     MCO
        SETB    MDO             ;Clear acknowledge bit.
        ACALL   DELAY
        CLR     MDE

        RET

;-----------------------------------------------------------------------
; RCVDATA: receives one or more bytes of data from an I2C slave device.
;-----------------------------------------------------------------------
RCVDATA:    INC     SLAVEADD        ;Set for READ of slave.
            ACALL   BITSTART        ;Acquire bus and send slave address.
            JB      NoAck,RDEX      ;Check for slave not responding.

RDLoop:     ACALL   RCVBYTE         ;Receive next data byte.
            MOV     @R1,A           ;Save data byte in buffer.
            INC     R1              ;Advance buffer pointer.
```

```
            DJNZ    BYTECNT,RDLoop ;Repeat untill all bytes received.

RDEX:       ACALL   SENDSTOP       ;Done, send an I2C stop.
            RET

;=========================================================================
; Main program
;=========================================================================
START:

        MOV     SP,#040h
        CLR     NOACK
        MOV     SLAVEADD,#088H
        MOV     BYTECNT,#3
        MOV     I2CCON,#0A8h

; code for a write mode ( master-transmitter to slave-receiver )

;       MOV     DPTR,#080H      ; master transmits to slave
;       MOV     A,#055H         ; datas which are located in
;       MOVX    @DPTR,A         ; the external memory
;       MOV     DPTR,#081H
;       MOV     A,#044H
;       MOVX    @DPTR,A
;       MOV     DPTR,#082H
;       MOV     A,#033H
;       MOVX    @DPTR,A

;       MOV     DPTR,#080h
;       ACALL   SENDATA

; code for a read mode ( master reads immediately after first byte )

        MOV     R1,#035h
        ACALL   RCVDATA
END
```

Приложение Б. Исходный текст SLAVE.ASM

```
$MOD812                         ; use ADuC812 & 8052 predefined symbols

BYTECNT         DATA    030h    ; byte counter for I2C routines

FLAGS           DATA    28h
GO              BIT     FLAGS.0 ; flag for all the interrupts
RC              BIT     FLAGS.1 ; flag for Write mode interrupt
TR              BIT     FLAGS.2 ; flag for Read mode interrupt

;=========================================================================
CSEG

ORG 0000H

        JMP START
;=========================================================================

ORG 003Bh                       ; I2C slave interrupt

        JB      RC,RECEIVE      ; depending on flags there
        JB      TR,TRANSMIT     ; are two different interrupts

;=========================================================================

ORG 007BH                       ; Subroutines

;-------------------------------------------------------------------------
; RECEIVE: receive interrupt routine
;-------------------------------------------------------------------------
RECEIVE:
        SETB    GO
        MOV     @R1,I2CDAT      ; move data on internal RAM
        CLR     I2CI            ; clear interrupt bit
        RETI
```

```
;-----------------------------------------------------------------------
; TRANSMIT: transmit interrupt routine
;-----------------------------------------------------------------------
TRANSMIT:
        SETB    GO
        MOV     I2CDAT,R0
        CLR     I2CI            ; clear interrupt bit
        RETI

;-----------------------------------------------------------------------
; RCVBYTE2: receive byte routine for read mode
;-----------------------------------------------------------------------
RCVBYTE2:
        NOP
        RET

;-----------------------------------------------------------------------
; RCVBYTE: receive byte routine
;-----------------------------------------------------------------------
RCVBYTE:
        JNB     GO,$            ; wait for the interrupt
        INC     R1              ; next storage will be on 41h then 42h
        CLR     GO              ; flag cleared for the next interrupt
        RET

;-----------------------------------------------------------------------
; RCVDATA: receive bytes routine
;-----------------------------------------------------------------------
RCVDATA:
        MOV     BYTECNT,#4      ; 4 bytes : address + 3 datas
LOOP2:  ACALL   RCVBYTE
        DJNZ    BYTECNT,LOOP2
        RET

;-----------------------------------------------------------------------
; SENDBYTE: byte transmit routine
;-----------------------------------------------------------------------
SENDBYTE:
        JNB     GO,$            ; wait for the interrupt
        INC     R0              ; 2nd data is 34h and 3rd data is 35h
        CLR     GO
        RET

;-----------------------------------------------------------------------
;SENDATA:bytes transmit routine
;-----------------------------------------------------------------------
SENDATA:
        MOV     BYTECNT,#3      ; 3 data will be send by the slave
LOOP:   ACALL   SENDBYTE
        DJNZ    BYTECNT,LOOP
        RET

;=======================================================================
;Main program
;=======================================================================
START:
        CLR     GO              ; clear flag used in the interrupt
        MOV     I2CADD,#044h    ; slave address
        MOV     SP,#020h
        MOV     IE,#80h         ; enable all the interrupts
        MOV     IE2,#01h        ; enable I2C interrupt
        MOV     I2CCON,#000h    ; slave mode

; code for write mode ( master-transmitter to slave-receiver )
;       SETB    RC              ; specific flag for interrupt routine
;       MOV     R1,#040h        ; first data to be stored in RAM at 40h
;       ACALL   RCVDATA         ; slave receives his address + 3 datas

; code for read mode ( master reads slave immediately after 1st byte )
        SETB    RC              ; specific flag for interrupt routine
        MOV     R0,#033h        ; first data send is 33h
        ACALL   RCVBYTE2        ; slave receives address send by master
        CLR     RC
        SETB    TR
        ACALL   SENDATA         ; slave sends 3 datas
        CLR     P3.4            ; led is off, everything is OK
END
```