МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное
образовательное учреждение высшего образования
«Юго-Западный государственный университет»
(ЮЗГУ)

Кафедра космического приборостроения и средств связи

УТВЕРЖДАЮ
Проректор по учебной работе
_____О.Г.Локтионова
_____2017 г.

ИССЛЕДОВАНИЕ СТРУКТУРЫ МИКРО-ЭВМ ADUC 812/841 И СИСТЕМЫ НА ЕЕ ОСНОВЕ

Методические указания по выполнению лабораторных работ
для студентов направления подготовки 11.03.02, 11.03.03

Курск 2017

УДК 681.5

Составитель В.Н. Усенков

Рецензент

Кандидат технических наук, профессор *В.А.Шлыков*

Исследование структуры микро-ЭВМ ADuC 812/841 и системы на ее основе : методические указания по выполнению лабораторных работ /Юго-Зап. гос. ун-т; сост.: В. Н. Усенков. - Курск, 2017. - 47 с.: ил. 14, прилож. 7. - Библиогр.: с. 47.

Приводятся краткие сведения о микро-ЭВМ ADuC 812/841, системы на ее основе, указания по исследованию структуры микро-ЭВМ и варианты заданий. Указывается порядок выполнения лабораторных работ. Приводятся рекомендации по оформлению отчетов и контрольные вопросы.

Методические указания соответствуют требованиям программы, утвержденной учебно-методическим объединением по специальностям автоматики и электроники (УМО АЭ).

Предназначены для студентов специальностей 11.03.02, 11.03.03 дневной и заочной форм обучения.

Текст печатается в авторской редакции

Подписано в печать . Формат 60× 84 1/16.
Усл. печ. л. *2,73*. Уч.-изд. л. *2,47*. Тираж 100 экз. Заказ. Бесплатно.

Юго-Западный государственный университет.
305040, г. Курск, ул. 50 лет Октября, 94.

Содержание

1. Исследование структуры микро-ЭВМ ADuC 812/841 и работа с системой на ее основе.....5
1.1 Цель работы.....5
1.2 Объект исследования и основные технические данные.....5
1.3 Подключение макета к ПЭВМ.....6
1.4 Программное обеспечение стенда.....6
1.5 Программа исследований.....7
1.6 Контрольные вопросы.....8
1.7 Содержание отчета.....9
Приложение 1.1 Исходный текст демонстрационной программы 10
Приложение 1.2 Принципиальная схема лабораторного макета.11
2. Программы многобайтовой арифметики для микро-ЭВМ ADuC.....12
2.1. Цель работы 12
2.2. Методические указания 12
2.3 Порядок выполнения работы.....13
2. 4 Контрольные вопросы.....14
2.5 Содержание отчета.....14
3 Управление таймерами микро-ЭВМ ADuC 841.....15
3.1 Цель работы.....15
3.2 Методические указания.....15
3.3 Порядок выполнения работ.....17
3.4 Контрольные вопросы.....18
3.5 Содержание отчета 19
Приложение 3.1. Демонстрационная программа управления дополнительным средством таймирования 20
4 Работа с EEPROM микро-ЭВМ ADuC.....23
4.1 Цель работы.....23
4.2 Методические указания.....23
3. Порядок выполнения работ.....24
4.4 Контрольные вопросы.....25
4.5 Содержание отчета.....25
Приложение 4.1 Демонстрационная программа управления энергонезависимой памятью.....26

5 Управление АЦП микро-ЭВМ ADuC 29
5.1 Цель работы 29
5.2 Методические указания 29
5.3 Порядок выполнения работ 31
5.4 Контрольные вопросы 31
5.5 Содержание отчета 32
Приложение 5.1 Демонстрационная программа управления АЦП 33
6 Управление ЦАП микро-ЭВМ ADuC 35
6.1 Цель работы 35
6.2 Методические указания 35
6.3 Порядок выполнения работ 36
6.4 Контрольные вопросы 37
6.5 Содержание отчета 38
Приложение 6.1 Демонстрационная программа управления ЦАП 39
7 Управление последовательным каналом микро-ЭВМ ADuC 41
7.1 Цель работы 41
7.2 Методические указания 41
7.3 Порядок выполнения работ 42
7.4 Контрольные вопросы 42
7.5 Содержание отчета 43
Приложение 7.1 Демонстрационная программа управления последовательным асинхронным каналом 44
8. **Литература** 47

1. Исследование структуры микро-ЭВМ ADuC 812/841 и работа с системой на ее основе

1.1 Цель работы

Изучение возможностей заданной микро-ЭВМ и подготовка к выполнению практической эксплуатации системы, построенных на ее основе. Микро-ЭВМ далее может именоваться как «микроконтроллер» или «μС».

1.2 Объект исследования и основные технические данные

ADUC812 - полностью интегрированная 12-битная система сбора данных, содержащая в одном кристалле высококачественный многоканальный АЦП с самокалибровкой, двойной ЦАП и программируемый 8 битный μС (совместимый с 8051 системой команд).

Работа программируемого 8051 совместимого ядра обеспечивается за счет встроенных 8КБ Flash/EEPROM памяти программы, 640 байтной Flash/EEPROM памяти данных и 256 байтного ОЗУ (рисунок 1).

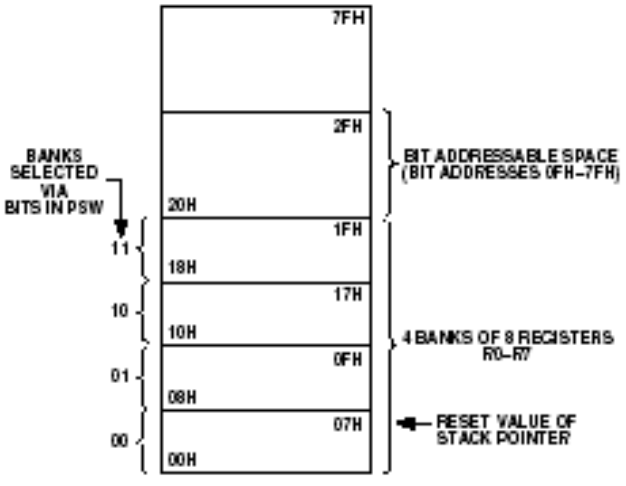


Figure 2. Lower 128 Bytes of Internal RAM

Рисунок 1 – Организация ОЗУ

Кроме того, μС содержит сторожевой таймер, схему слежения за напряжением питания и DMA контроллер для передачи данных от

АЦП. Тридцать две программируемых линии ввода-вывода, I^2C, SPI и UART - совместимые последовательные порты ввода- вывода позволяют реализовывать интерфейсы с μP и различными приборами.

Нормальный режим, режим останова и режим отключения как ядра μC, так и аналогового интерфейса позволяют снизить потребление прибора и использовать его в устройствах с автономным питанием. Прибор может работать от 3 В и 5 В источника питания, имеет индустриальный рабочий температурный диапазон и выпускается в 52 выводных пластиковых QFP корпусах и в виде 52 выводных бескорпусных приборов.

1.3 Подключение макета к ПЭВМ

Работа со стендом начинается с ознакомления с описанием, составом, порядком включения и выключения питания и подключением к ПЭВМ.

Подробное описание макета приведено в [1]. Для физического соединения ПЭВМ и стенда требуется строгое соблюдение порядка:

- отключите питание ПЭВМ
- подсоедините к доступному разъему RS-232 (COM - port) кабель, входящий в состав макета
- подключите второй конец кабеля к соответствующему разъему на плате макета.
- убедитесь, что соединение выполнено правильно и надежно
- подключите блок питания макета к сети
- подключите штекер блока питания к макету
- подайте питание на ПЭВМ

1.4 Программное обеспечение стенда

В состав программного обеспечения поддержки макета входят

- программы для загрузки программ в микроЭВМ,
- пакет программ для разработки на языке ассемблера микроЭВМ

- пакет программ для разработки на языке С.

В процессе выполнения лабораторной работы необходимо:
изучить текст демонстрационной программы;
изучить аппаратные ресурсы микро-ЭВМ, задействованные в программе;
сформулировать признаки корректной работы программы при демонстрации;
откомпилировать исходный текст программы
создать исполнимый файл;

1.5 Программа исследований

- Загрузить демонстрационную программу для проверки работоспособности и функциональности стенда. Для выполнения этого этапа необходимо ознакомиться с описанием программы загрузки. Она существует в двух видах:
  - для ДОС
  - для ОС Windows
- После запуска загрузчика выполнить шаги для активизации встроенной программы связи по последовательному каналу с соответствии с рисунком 2
- С помощью загрузчика, после установления и подтверждения связи:
  - загрузить исполнимый файл в микро-ЭВМ;
  - запустить программу и убедится в ее работоспособности.

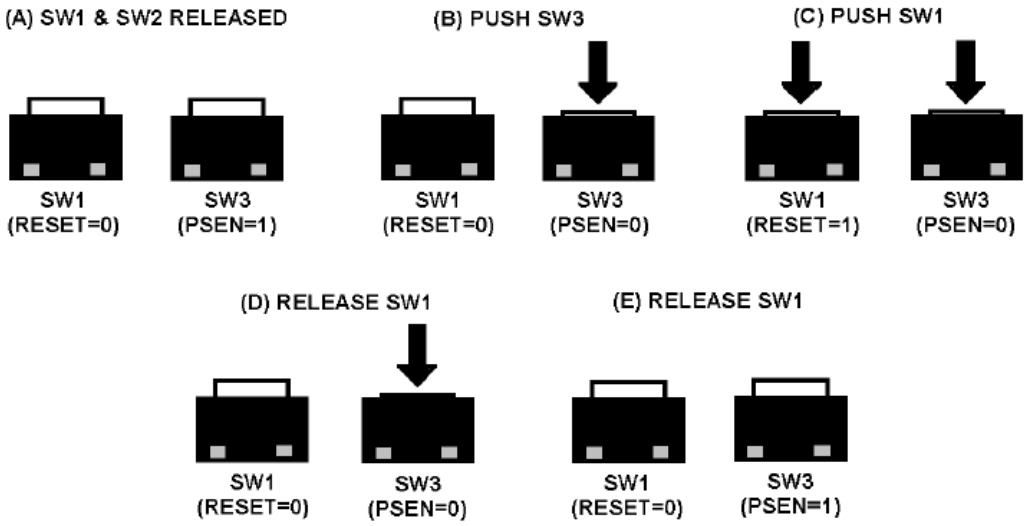


Рисунок 2 - Порядок управления переключателями в процессе соединения микро-ЭВМ и ПЭВМ

1.6 Контрольные вопросы

- перечислить и описать состав периферийного оборудования, находящегося на кристалле микро-ЭВМ

- перечислить и описать основные компоненты, установленные на печатной плате макета

- как осуществить загрузку программы, написанной для данной микро-ЭВМ, в ее память

- какие программные средства необходимы для написания программы на языке ассемблера микро-ЭВМ

- какие программные средства необходимы для подготовки загружаемого программного кода для микро-ЭВМ

1.7 Содержание отчета

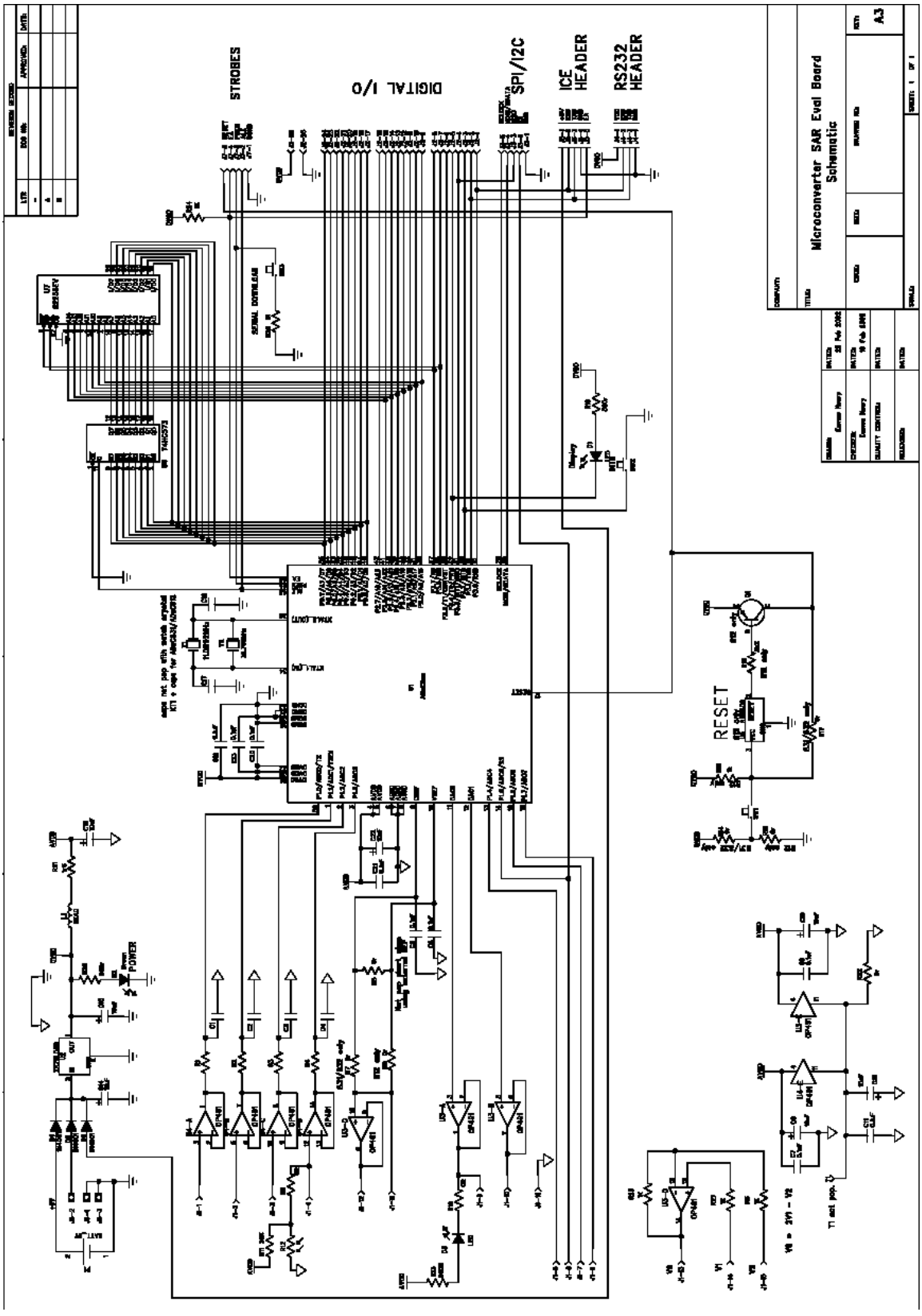
Отчет должен содержать:

- титульный лист;
- цель исследований;
- структурную схему макетной установки;
- текст и описание демонстрационной программы, содержательные комментарии;
- скриншоты основных этапов загрузки программы;
- протокол исследования работоспособности демонстрационной программы.

Приложение 1.1 Исходный текст демонстрационной программы

```
;=====================================================================
;
; Author        : ADI - Apps
;
; Date          : 13 March 2003
;
; Filename      : DemoCode.asm
;
; Hardware      : ADuC8xx
;
; Description   : Blinks LED continuously.
;                 Pressing Int0 slows LED toggle rate each time
;                 it is pressed.
;
;=====================================================================
$MOD52                          ; Use 8052 predefined Symbols
;Definitions
LED1     EQU     P3.3           ; P3.3 is LED on ADuC814 eval boards
LED2     EQU     P3.4           ; P3.4 is LED on all other ADuC8xx eval boards
FLAG     BIT     00H            ; define Flag variable

CSEG                            ; Defines the following as a segment of code
ORG     0000H                   ; Load Code at '00H'

         JMP     MAIN           ; Jump to MAIN
;=====================================================================
;INT0 ISR (Interrupt Service Routine)
ORG 0003h                      ;
         INC     A              ; Increment Acc
         RETI                   ; Return from Interrupt
;=====================================================================
ORG 060h                        ; Start MAIN outside ISR space
MAIN:                           ; (main program)
         SETB    IT0            ; INT0 edge triggered
         SETB    EA             ; enable interrupts
         SETB    EX0            ; enable INT0

         CLR     FLAG           ; Clear Bit defined as FLAG

         MOV     A,#01H         ; Initialize A to 1
BLINK:   CPL     LED1           ; blink LED1 using compliment instruction
         CPL     LED2           ; blink LED2
         CALL    DELAY         ; Jump to subroutine DELAY
         JNB     FLAG,BLINK     ; If FLAG is still cleared the jump to Blink
;---------------------------------------------------------------------
DELAY:
         MOV     R0,A           ; Acc holds delay variable
 DLY0:   MOV     R1,#019h       ; Set up delay loop0
 DLY1:   MOV     R2,#0FEh       ; Set up delay loop1
         DJNZ    R2,$           ; Dec R2 & Jump here until R2 is 0
         DJNZ    R1,DLY1        ; Dec R1 & Jump DLY1 until R1 is 0
         DJNZ    R0,DLY0        ; Dec R0 & Jump DLY0 until R0 is 0
         RET                    ; Return from subroutine

;=====================================================================
END
```

Приложение 1.2 Принципиальная схема лабораторного макета



2. Программы многобайтовой арифметики для микро-ЭВМ ADuC

2.1 Цель работы

Получение практических навыков разработки программ для микро-ЭВМ на языке ассемблера.

2.2 Методические указания

Совместимость системы команд и базовой архитектуры ядра микро-ЭВМ с широко известными микро-ЭВМ семейства intel MSC-51 позволяет использовать различные пакеты программ для трансляции и отладки. Существует достаточное количество литературы, посвященной этому вопросу.

При написании программы следует уточнить у преподавателя:

- количество байтов в операндах;

- выполняемую арифметическую функцию.

По умолчанию предполагается сложение двоичных операндов длиной 3 байта. Операнды рекомендуется располагать упорядоченно в регистровой памяти микро-ЭВМ, предусматривая возможное увеличение разрядности.

Достаточный объем памяти данных позволяет выделить отдельные блоки для хранения каждого операнда и результата.

Демонстрационные примеры (один – два варианта различных операндов) должны предусматривать ситуации межбайтового переноса и переполнение разрядной сетки.

При возможности, целесообразно реализовать интерфейсную часть программы, позволяющую задавать операнды с клавиатуры и получать результат на экране монитора. С этой целью рекомендуется обеспечить обмен данными между микро-ЭВМ и ПЭВМ по последовательному каналу. На ПЭВМ при этом возможно использование любой программы обмена,

существующей в рамках ОС (например, Гипертерминал для Windows).

В альтернативных, более простых вариантах, допускается:

- операнды загружать вместе с программой, а результат выдавать в последовательный канал;

- считывать содержимое областей, хранящих результат, после запуска программы.

Программы для ассемблирования, создания исполнимого кода и прочие доступны на сервере кафедры.

2.3 Порядок выполнения работы

- Разработать алгоритм сложения(вычитания) многобайтовых чисел
- Проработать структуру хранения данных
- Написать программу на языке ассемблера, реализующую алгоритм
- Оттранслировать программу с помощью кросс (макро)ассемблера ASEM или аналогичного
- Создать файл с исполнимым кодом программы (например, программой hextobin)
- Загрузить программы для проверки работоспособности и функциональности макета
- После запуска загрузчика выполните шаги для активизации встроенной программы связи по последовательному каналу в соответствии с рисунком 2
- С помощью загрузчика, после установления и подтверждения связи, необходимо:
  - загрузить исполнимый файл в микро-ЭВМ
  - запустить программу и убедится в ее работоспособности.

2. 4 Контрольные вопросы

- как осуществить загрузку программы, написанной для данной микро-ЭВМ, в ее память
- какие программные средства необходимы для написания программы на языке ассемблера микро-ЭВМ
- какие программные средства необходимы для подготовки загружаемого программного кода для микро-ЭВМ
- как осуществить сложение операндов в формате BCD
- какие виды адресации позволяют реализовать сложение операндов в цикле.

2.5 Содержание отчета

Отчет о лабораторной работе должен содержать:

- титульный лист;
- цель исследований;
- структурную схему макетной установки;
- текст и описание программы, содержательные комментарии;
- скриншоты основных этапов выполнения программы;
- протокол исследования работоспособности демонстрационной программы.

3 Управление таймерами микро-ЭВМ ADuC 841

3.1 Цель работы

Получение практических навыков управления таймерами-счетчиками микро-ЭВМ при написании программ на языке ассемблера.

3.2 Методические указания

Микро-ЭВМ содержит как стандартные для ядра MCS-51 таймеры счетчики 0/1/2, так и дополнительные средства таймирования [1].

В лабораторной работе допускается выбрать любой из стандартных таймеров для исследования.

Текст демонстрационной программы для формирования длительных временных интервалов приведен в приложении 3.1.

При управлении таймером 0 необходимо ознакомиться с его структурой (рисунок 4) и форматами управляющих слов [1]

13-битовый таймер/счетчик:

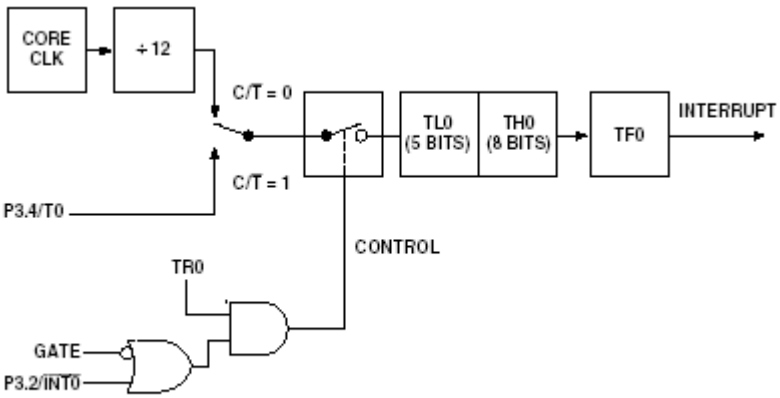


Figure 45. Timer/Counter 0, Mode 0

Рисунок 4 - Коммутация сигналов таймера/счетчика 0 в режиме 0

16-битовый таймер/счетчик:

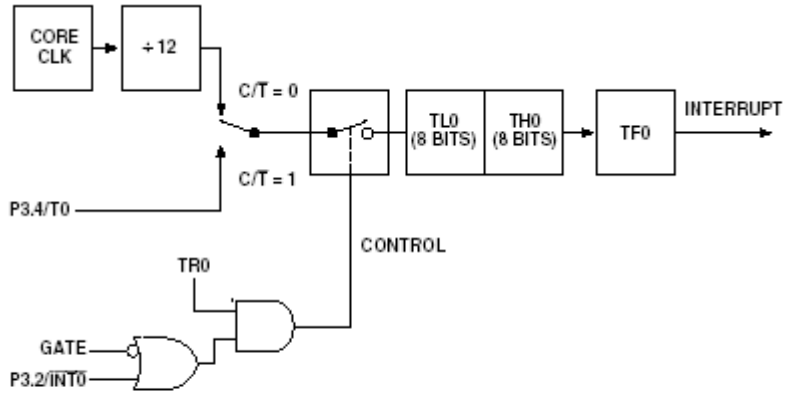


Figure 46. Timer/Counter 0, Mode 1

Рисунок 5 - Коммутация сигналов таймера/счетчика 0 в режиме 1

8-битовый таймер-счетчик с автозагрузкой:

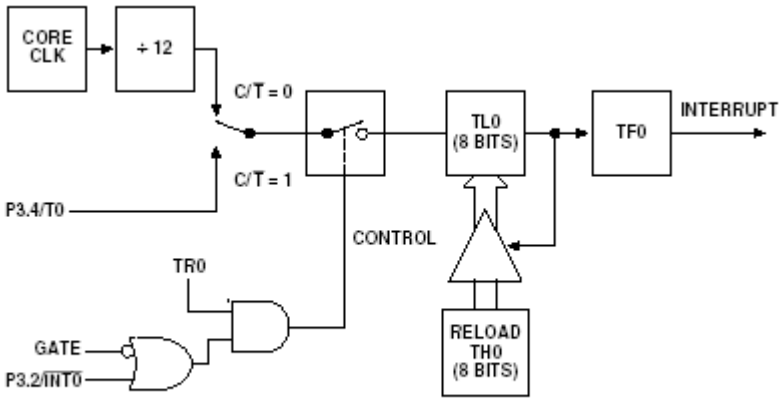


Figure 47. Timer/Counter 0, Mode 2

Рисунок 6 - Коммутация сигналов таймера/счетчика 0 в режиме 2

Два раздельных 16-битовых таймера/счетчика:

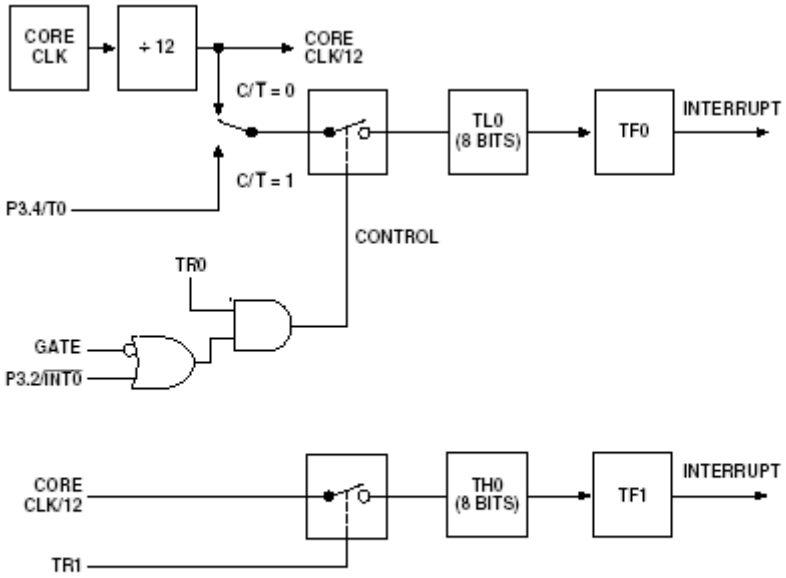


Figure 48. Timer/Counter 0, Mode 3

Рисунок 7 - Коммутация сигналов таймера/счетчика 0 в режиме 3.

При возможности, целесообразно реализовать интерфейсную часть программы, позволяющую задавать параметры работы с клавиатуры и получать результат на экране монитора. С этой целью рекомендуется обеспечить обмен данными между микро-ЭВМ и ПЭВМ по последовательному каналу. На ПЭВМ при этом возможно использование любой программы обмена, существующей в рамках ОС (например, Гипертерминал для Windows).

В альтернативных, более простом вариантах, допускается:

- параметры загружать вместе с программой, а результат выдавать в последовательный канал;

- считывать содержимое областей, хранящих результат, после запуска программы.

Программы для ассемблирования, создания исполнимого кода и прочие доступны на сервере кафедры.

3.3 Порядок выполнения работ

- Разработать алгоритм программы, использующей указанный таймер для формирования заданного временного интервала. При нажатии кнопки на макете должно происходить удвоение, учетверение и возврат к исходной величине задержки.
- Изучить форматы управляющих регистров для указанного таймера.
- Написать программу на языке ассемблера, реализующую алгоритм.
- Оттранслировать программу с помощью кросс (макро)ассемблера ASEM или аналогичного.
- Создать файл с исполнимым кодом программы (например, программой hextobin)
- Загрузить программы для проверки работоспособности и функциональности макета.
- После запуска загрузчика выполните шаги для активизации встроенной программы связи по последовательному каналу:
- С помощью загрузчика, после установления и подтверждения связи, необходимо:
    - загрузить исполнимый файл в микро-ЭВМ;
    - запустить программу и убедится в ее работоспособности.

3.4 Контрольные вопросы

- перечислить и охарактеризовать стандартные возможности таймирования MCS-51

- перечислить и охарактеризовать дополнительные возможности таймирования ADuC

- какие таймеры-счетчики могут использоваться для тактирования встроенного асинхронного канала

- как подать внешний сигнал на входы таймеров/счетчиков

3.5 Содержание отчета

Отчет о лабораторной работе должен содержать:
- титульный лист;
- цель исследований;
- структурную схему макетной установки;
- текст и описание программы, содержательные комментарии;
- скриншоты основных этапов выполнения программы;
- протокол исследования работоспособности демонстрационной программы.

Приложение 3.1. Демонстрационная программа управления дополнительным средством таймирования.

```
;*********************************************************************
;
; Author        : ADI - Apps            www.analog.com/MicroConverter
;
; Date          : May 2002
;
; File          : TIC.asm
;
; Hardware      : ADuC831
;
; Description   : Demonstrates a use of a timer interval counter for
;             counting longer intervals than the standard 8052
;             timers are capable of.
;
;             The LED will, on power up, flash at 6.4Hz. By pressing
;             the external interrupt button INT0 the counter will
;             count how long the button is pressed correct to
;             1/128th of a second. When released the program will
;             flash the light at the measured time correct only to
;             the nearest unit (1/128s, seconds, minutes or hours)
;             rounded DOWN.
;             eg) If the button was pressed for 0.91000s the light
;             would complement every 0.90625s (less than 1 second
;             therefore it measures in 1/128s and the nearest unit
;             less than 0.91000s is 0.90625s). However if the light
;             was on for 1.6s it complements every 1s as the nearest
;             unit is now the second.
;             Pressing the INT0 button again will record a new
;             time interval which will flash the light in the same
;             way.

;*********************************************************************

$MOD831                         ; Use 8052&ADuC831 predefined symbols

LED     EQU     P3.4            ; P3.4 drives red LED on eval board
BUTTON  EQU     P3.2             ; P3.2 drives the INT0 button on the
                        ; eval board

;_____________________________________________________________________
                                                    ; BEGINNING OF CODE
CSEG

ORG 0000h

        JMP     MAIN            ; jump to main program

;_____________________________________________________________________
                                    ; EXTERNAL INTERRUPT VECTOR SPACE
ORG 0003h ;  (INT0 ISR)

       CLR    LED          ; Turn ON the LED while the INT0
                           ; is pressed

; reset all counters and then start counting
        ANL     TIMECON, #0FEh  ; Clear the TCEN bits to clear the
                                ; registers;
                                ;  -Hthsec
                                ;  -sec
                                ;  -min
                                ;  -hour.
                                ; and to clears the internal counter
```

```
        ORL     TIMECON, #01h  ; Set the TCEN bit to restart counting

        JNB     BUTTON, $       ; Wait here while button is pressed

        ANL     TIMECON, #0FDh  ; Clear the TIEN bit to stop the
                                ; counter

; after button is released we can store the value in intval

LOOP:   SETB     LED             ; Turn off LED to indicate that the
                           ; button is released.
       MOV    A, HOUR
       CJNE   A, #00H, HOURS      ; Check if any hrs have been counted
                           ; If so jump to HOURS
        MOV  A, MIN
       CJNE   A, #00H, MINS; Check if any mins have been counted
                           ; If so jump to MINS
        MOV  A, SEC
        CJNE A, #00H, SECS; Check if any secs have been counted
                           ; If so jump to SECS

HUNTHS:      MOV    INTVAL, HTHSEC      ;load the value of HTHSEC into INTVAL
       MOV     TIMECON, #00h   ; clear TCEN to reset the registers
        MOV     TIMECON, #03H   ; change TIMECON to measure in 1/128s
                           ; reset TIEN
       RETI

SECS:  MOV    INTVAL, SEC  ; load the value of SEC into INTVAL
       MOV     TIMECON, #00h   ; clear TCEN to reset the registers
       MOV    TIMECON, #13H; change TIMECON to measure in secs
       RETI

MINS:  MOV    INTVAL, MIN  ; load the value of MIN into INTVAL
       MOV     TIMECON, #00h   ; clear TCEN to reset the registers
       MOV    TIMECON, #23H; change TIMECON to measure in mins
       RETI

HOURS: MOV    INTVAL, HOUR ; load the value of HOUR onto INTVAL
       MOV     TIMECON, #00h   ; clear TCEN to reset the registers
       MOV    TIMECON, #33H; change TIMECON to measure in hours
        RETI
;_____________________________________________________________________
                                        ; TII INTERRUPT VECTOR SPACE
ORG 0053h

       CPL LED                  ; Complement the LED every time the
                           ; measured time runs up.
       RETI
;_____________________________________________________________________

ORG 0060h

MAIN:

; Configure Time Interval Counter

       MOV    TIMECON, #03h  ; initialise timecon to count in 1/128s
                             ; -set TCEN to enable the time clock
                             ; -set TIEN to enable the TIC
                             ; -clear STI to allow automatic relaod
                             ;  of interval timeout
                          ; -clear TFH to disable 24 hr counting

        MOV    INTVAL, #0Ah   ; initialise to blink LED every 10 units
                             ; the units are 1/128s
```

```
; Configure External Interrupt
       SETB    IT0            ; INT0 edge triggered
         SETB    EX0            ; enable INT0 (button on eval board)
       MOV     IEIP2,#04H     ; enable time interval interrupt

         SETB    EA             ; enable global interrupts


       JMP     $              ; wait here for interrupts
                              ; main program can be inserted here

;_____________________________________________________________________

END
```

4 Работа с EEPROM микро-ЭВМ ADuC

4.1 Цель работы

Получение практических навыков записи/считывания данных во внутреннюю энергонезависимую память микро-ЭВМ при написании программ на языке ассемблера.

4.2 Методические указания

В состав моделей микро-ЭВМ ADuC входит энергонезависимая память, допускающая многократное перепрограммирование данных программой, выполняющейся на микро-ЭВМ.

Время сохранения данных заявляется изготовителем достаточно большим (см. рисунок ниже):

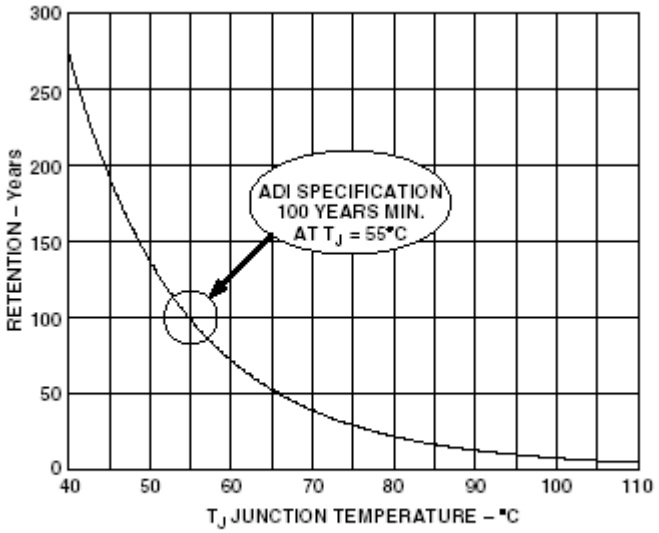


Figure 18. Flash/EE Memory Data Retention

Рисунок 8 - Время хранения информации во внутренней программируемой энергонезависимой памяти.

Более подробные данные о режимах записи/считывания приведены в [1].

В лабораторной работе процедуры занесения/считывания данных исследуется в рамках демонстрационной программы (Приложение 4.1)

При возможности, целесообразно реализовать интерфейсную часть программы, позволяющую задавать параметры работы с клавиатуры и получать результат на экране монитора. С этой целью рекомендуется обеспечить обмен данными между микро-ЭВМ и ПЭВМ по последовательному каналу. На ПЭВМ при этом возможно использование любой программы обмена, существующей в рамках ОС (например, Гипертерминал для Windows).

В альтернативных, более простом вариантах, допускается:

- параметры загружать вместе с программой, а результат выдавать в последовательный канал;

- считывать содержимое областей, хранящих результат, после запуска программы.

Программы для ассемблирования, создания исполнимого кода и прочие доступны на сервере кафедры.

3. Порядок выполнения работ

- Изучить методику записи и считывания данных в энергонезависимую память.
- Изучить демонстрационную программу на языке ассемблера.
- Оттранслировать программу с помощью кросс (макро)ассемблера ASEM или аналогичного.
- Создать файл с исполнимым кодом программы (например, программой hextobin)
- Загрузить программы для проверки работоспособности и функциональности макета.

- После запуска загрузчика выполните шаги для активизации встроенной программы связи по последовательному каналу:
- С помощью загрузчика, после установления и подтверждения связи, необходимо:
    - загрузить исполнимый файл в микро-ЭВМ;
    - запустить программу и убедиться в ее работоспособности.

4.4 Контрольные вопросы

- каков объем энергонезависимой памяти для различных моделей ADuC

- возможно ли заносить данные до запуска программ на микро-ЭВМ

- какова скорость записи/считывания данных для энергонезависимой памяти

- возможно ли побайтовое изменение данных в энергонезависимой памяти

4.5 Содержание отчета

Отчет о лабораторной работе должен содержать:
- титульный лист;
- цель исследований;
- структурную схему макетной установки;
- текст и описание программы, содержательные комментарии;
- скриншоты основных этапов выполнения программы;
- протокол исследования работоспособности демонстрационной программы.

Приложение 4.1 Демонстрационная программа управления энергонезависимой памятью.

```
;*********************************************************************
;
; Author        : ADI - Apps            www.analog.com/MicroConverter
;
; Date          : May 2002
;
; File          : dataflsh.asm
;
; Hardware      : ADuC831
;
; Description   : Demonstrates use of the on-chip read/write 4096 byte
;                 FlashEE data memory space.  Stores a sequence of
;                 button presses (INT0 button on eval board) in data
;                 FlashEE space.  Replays sequence on LED when board
;                 is reset or power cycled.
;                 The ADuC831 stores the play sequece in data flash
;                 until another is recorded with a new set of button
;                 presses.  To record a new sequence, just wait until
;                 the current one finishes playing (LED is off) and
;                 enter new sequence via button (INT0).
;
;*********************************************************************

$MOD831                         ; Use 8052&ADuC831 predefined symbols

LED       EQU    P3.4           ; P3.4 drives red LED on eval board
BUTTON    EQU    P3.2           ; button on eval board drives P3.2
PREVIOUS  EQU    F0             ; flag to hold previous button value
READ      EQU    01h            ; FlashEE command:  'read page'
WRITE     EQU    02h            ; FlashEE command:  'write page'
VERIFY    EQU    04h            ; FlashEE command:  'verify page'
ERASE     EQU    05h            ; FlashEE command:  'erase page'
ERASEALL  EQU    06h            ; FlashEE command:  'erase all'
;---------------------------------------------------------------------
;_____________________________________________________________________
                                                    ; BEGINNING OF CODE
CSEG

ORG 0000h

MAIN:
        SETB    LED             ; turn LED off
       MOV     EADRH,#0        ; set data FlashEE address to page 0
        MOV     EADRL,#0

; READ FLASH/EE DATA and indicate values via LED on and off times...

READFLASH:
        MOV     ECON,#READ      ; read current 4byte page of FlashEE
                                ; into EDATA1,2,3,4
        MOV     A,EDATA4
        CJNE    A,#1,RECORD     ; if EDATA4 is 1, then page contains
                                ; a valid play sequence
                                ;        => Play this sequence
                                ; otherwise jump to record mode


;---------------------------------------------------------------------
PLAYBACK:
        CALL    BLINK           ; flash LED for period determined
                                ; by FlashEE data just read
        MOV     A,EADRL
        CJNE    A,#0FFh,INCPAGE1 ; if low address is FFh then increment high address
```

```
       INC     EADRH
INCPAGE1:
        INC     EADRL           ; increment to next FlashEE page addr
        MOV     A,EADRH
        CJNE    A,#04h,READFLASH
                                ; if address is less than 160 then jump
                                ; to read the next page
       ; when PLAYBACK is finished jump to RECORD mode

;-----------------------------------------------------------------
RECORD:
        SETB    LED
        JB      BUTTON,$        ; wait for first button press

        ; once button is pressed, erase dataflash
        MOV     ECON,#ERASEALL  ; clear all data FlashEE memory
        MOV     EADRH,#0
        MOV     EADRL,#0

        ; record time of button press

RECORD_NEXT_TIME:
        CALL    RECORDTIME

        MOV     EDATA1,DPL      ; place DPTR in EDATA1,2,3
        MOV     EDATA2,DPH
        MOV     EDATA3,DPP
        MOV     EDATA4,#1       ; put 1 in EDATA4 as identifier
        MOV     ECON,#WRITE     ; write EDATA1-4 into pre-erased
                                ; page of FlashEE data memory

        MOV     ECON,#VERIFY    ; verify current page is same as
        MOV     A,ECON          ; EDATA1-4.  If same, ECON=0.
        JNZ     RECORD          ; if verify fails, jump to RECORD

        MOV     A,EADRL
        CJNE    A,#0FFh,INCPAGE2 ; if low address is FFh then increment high address
       INC     EADRH
INCPAGE2:
        INC     EADRL           ; increment to next FlashEE page addr
        MOV     A,EADRH
        CJNE    A,#04h,RECORD_NEXT_TIME
                                ; record first 160 button presses only

        ; when flash/EE data space is full turn off LED and wait
        ; for a power cycle
        SETB     LED
        JMP     $


;=================================================================
;                           FUNCTIONS
;=================================================================


;_________________________________________________________________
                                                        ; SUBROUTINES
BLINK:
        ; Turn LED ON/OFF based on the time in EDATA3/2/1
        CPL     LED

       CLR     A
        MOV     DPL,A
        MOV     DPH,A          ; clear DPTR
        MOV     DPP,A

        INC     EDATA1          ; EDATA1 -> EDATA3 should be
```

```
        INC     EDATA2          ; incremented for the below to work
        INC     EDATA3

BLINKLOOP:
        ; the record loop takes 6 instruction cycles hence 4 NOPs are
        ; required to make the Playback loop 6 instruction cycles also.
        ; NOTE: the main Playback loop will jump to BLINKLOOP after
        ; decrementing EDATA1 and hence the time required to decrement
        ; EDATA2 (approx 1/256 time of main loop) and EDATA3 are ignored.
        NOP                                                     ; 1
        NOP                                                     ; 1
        NOP                                                     ; 1
        NOP                                                     ; 1
        DJNZ    EDATA1, BLINKLOOP                               ; 2
       DJNZ   EDATA2, BLINKLOOP ; EDATA1 overflows back to FFh
        DJNZ    EDATA3, BLINKLOOP ; EDATA2 overflows back to FFh

        RET

;_____________________________________________________________________

RECORDTIME:
        ; Record how long button is pressed for and store in EDATA3/2/1
       CLR     A
        MOV     DPL,A
        MOV     DPH,A          ; clear DPTR
        MOV     DPP,A

        CPL     LED

        ; measure how long the button is either pressed or released
        ; for. If the button is pressed then the LED is on. If the
        ; button is released then the LED is off.
RECORDLOOP:
        INC     DPTR            ; incrementing DPTR..           ;  2
        JNB     LED, CT                                         ;  2
       JMP    CHKB
CT:     JNB     BUTTON,RECORDLOOP                               ;  2
                                ; keep recording while button is pressed
        RET
CHKB:   JB      BUTTON,RECORDLOOP                               ;  2
                                ; keep recording while button is released
        RET

; DPP,DPH,DPL now holds a number that represents the length of
; time between button edges.  this data will be stored in FlashEE
; space for use in controlling LED on and off times in "play" mode.

;_____________________________________________________________________


END
```

5 Управление АЦП микро-ЭВМ ADuC

5.1 Цель работы

Получение практических навыков оцифровки данных с помощью встроенного аналого-цифрового преобразователя микро-ЭВМ при написании программ на языке ассемблера.

5.2 Методические указания

Встроенный 12-битовый АЦП имеет 8 каналов для ввода данных. В лабораторном макете часть каналов буферирована каскадами на ОУ, и именно эти каналы будут использоваться.

Передаточная характеристика приведена на рисунке 9.

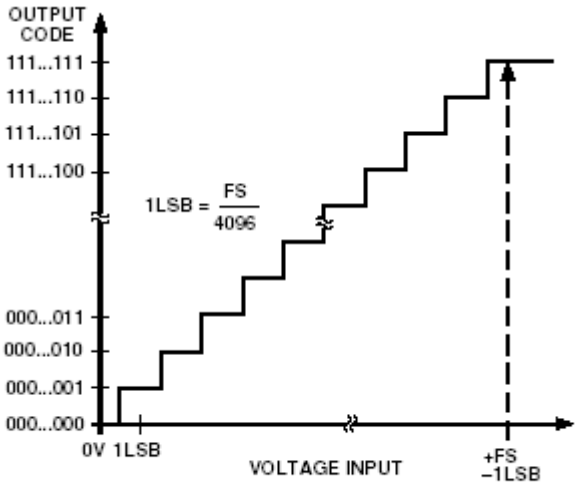


Figure 7. ADC Transfer Function

Рисунок 9 – характеристика преобразования АЦП

Формат данных после оцифровки демонстрируется нижеследующим рисунком.

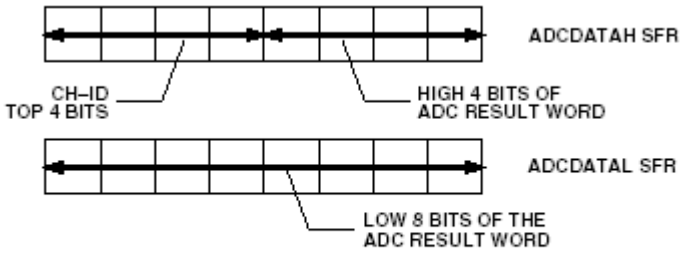


Figure 8. ADC Result Format

Рисунок 10 – Формат данных АЦП

Рекомендуемая схема буферирования приведена на рисунке 11.

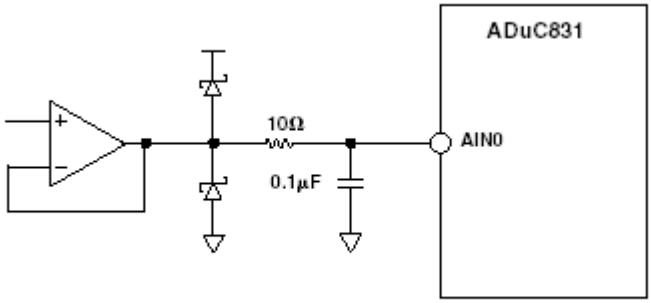


Figure 10. Buffering Analog Inputs

Рисунок 11- Буферирование АЦП

Более подробные данные о работе АЦП приведены в [1].

В лабораторной работе процедуры оцифровки данных исследуется в рамках демонстрационной программы (Приложение 5.1.)

При возможности, целесообразно реализовать интерфейсную часть программы, позволяющую задавать параметры работы с клавиатуры и получать результат на экране монитора. С этой целью рекомендуется обеспечить обмен данными между микро-ЭВМ и ПЭВМ по последовательному каналу. На ПЭВМ при этом возможно использование любой программы обмена, существующей в рамках ОС (например, Гипертерминал для Windows).

В альтернативных, вариантах, допускается:

- параметры загружать вместе с программой, а результат выдавать в последовательный канал;

- считывать содержимое областей, хранящих результат, после запуска программы.

Программы для ассемблирования, создания исполнимого доступны на сервере кафедры.

5.3 Порядок выполнения работ

1. Изучить принципы функционирования встроенного в ADuC АЦП и формат управляющих регистров.
2. Изучить демонстрационную программу на языке ассемблера.
3. Оттранслировать программу с помощью кросс (макро)ассемблера ASEM или аналогичного.
4. Создать файл с исполнимым кодом программы (например, программой hextobin)
5. Загрузить программы для проверки работоспособности и функциональности макета.
6. После запуска загрузчика выполните шаги для активизации встроенной программы связи по последовательному каналу

С помощью загрузчика, после установления и подтверждения связи, необходимо:

7. загрузить исполнимый файл в микро-ЭВМ;
8. запустить программу и убедиться в ее работоспособности.

5.4 Контрольные вопросы

- для чего необходимо буферирование входных сигналов?
- какова максимальная скорость оцифровки данных?

- нужно ли приостанавливать работу программы при включении АЦП?

5.5 Содержание отчета

- Отчет о лабораторной работе должен содержать:
- титульный лист;
- цель исследований;
- структурную схему макетной установки;
- текст и описание программы, содержательные комментарии;
- скриншоты основных этапов выполнения программы;
- протокол исследования работоспособности демонстрационной программы.

Приложение 5.1 Демонстрационная программа управления АЦП

```
;*****************************************************************
;
; Author        : ADI - Apps            www.analog.com/MicroConverter
;
; Date          : 31 Jan 2002
;
; File          : ADCcont.asm
;
; Hardware      : ADuC831
;
; Description   : Performs ADC conversions in continuous mode at a
;                 rate of 81.31KSPS (assuming an 11.0592MHz Mclk).
;                 Outputs ADC results on P0 & P2.  Continuously
;                 flashes LED (independently of ADC routine) at
;                 approximately 5Hz.
;                 All rate calculations assume an 11.0592MHz Mclk.
;
;*****************************************************************

$MOD831                         ; Use 8052&ADuC831 predefined symbols

LED     EQU     P3.4            ; P3.4 drives red LED on eval board
CHAN    EQU     0               ; convert this ADC input channel..
                                ;
;_____________________________________________________________________
                                                    ; BEGINNING OF CODE
CSEG

ORG 0000h

        JMP     MAIN            ; jump to main program
;_____________________________________________________________________
                                                  ; INTERRUPT VECTOR SPACE
ORG 0033H ; (ADC ISR)

        MOV     P0,ADCDATAL     ; ADC result low byte to Port0
        MOV     P2,ADCDATAH     ; high nibble and channel ID to Port2
        RETI

;=====================================================================
                                                            ; MAIN PROGRAM
ORG 004Bh

MAIN:

; PRECONFIGURE...

        MOV     ADCCON1,#0B0h   ; power up ADC, 12.3us conv+acq time
        MOV     ADCCON2,#CHAN   ; select channel to convert

; LAUNCH CONTINUOUS CONVERSIONS...

        SETB EA              ; enable interrupts
        SETB    EADC            ; enable ADC interrupt
        SETB    CCONV           ; begin continuous conversions

; CONTINUE WITH OTHER CODE...

AGAIN:  CPL     LED             ; blink (complement) the LED
        CALL    DELAY           ; delay 100ms
        JMP     AGAIN           ; repeat

; the micro is free to continue with other tasks (flashing the LED in
```

```
; this case) while the ADC is continuously converting, and results
; are being handled by the ADC interrupt service routine.

;_____________________________________________________________________________
                                                                ; SUBROUTINE

DELAY:                          ; delay 100ms

        MOV     R7,#200         ; 200 * 500us = 100ms
DLY1:   MOV     R6,#229         ; 229 * 2.17us = 500us
        DJNZ    R6,$            ; sit here for 500us
        DJNZ    R7,DLY1         ; repeat 200 times (100ms total)
        RET

;_____________________________________________________________________________

END
```

6 Управление ЦАП микро-ЭВМ ADuC

6.1 Цель работы

Получение практических навыков формирования аналогового сигнала с помощью встроенного цифро-аналогового преобразователя микро-ЭВМ при написании программ на языке ассемблера.

6.2 Методические указания

Встроенный 12-битовый АЦП имеет 8 каналов для ввода данных. В лабораторном макете часть каналов буферирована каскадами на ОУ, и именно эти каналы будут использоваться.

Передаточная характеристика приведена на рисунке 12.

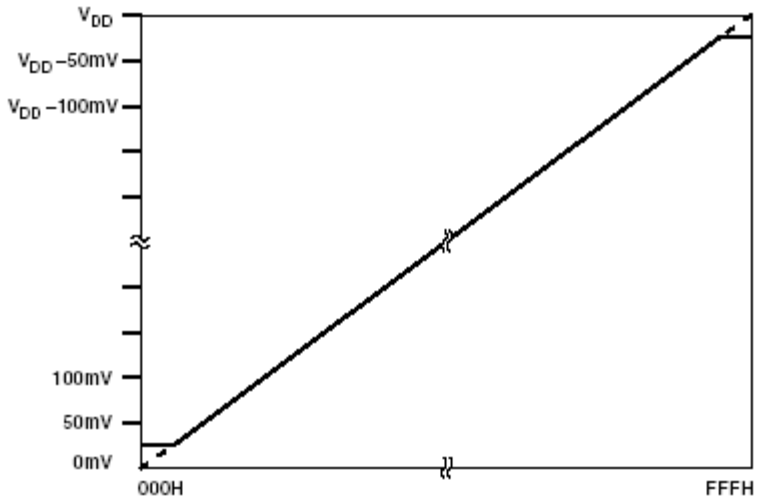


Figure 22. Endpoint Nonlinearities Due to Amplifier Saturation

Рисунок 12 – Передаточная характеристика встроенного усилителя

Рекомендуемая схема буферирования приведена на рисунке 13.

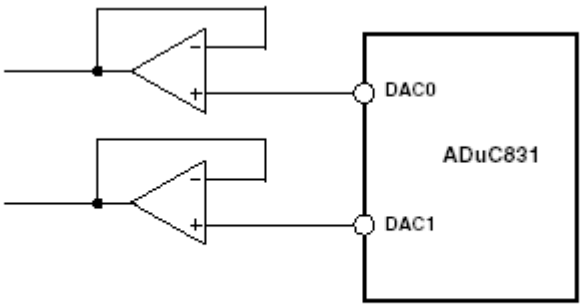


Figure 25. Buffering the DAC Outputs

Рисунок 13 – Буферирование ЦАП

Более подробные данные о работе ЦАП приведены в [1].

В лабораторной работе процедуры оцифровки данных исследуется в рамках демонстрационной программы (Приложение 8.1)

При возможности, целесообразно реализовать интерфейсную часть программы, позволяющую задавать параметры работы с клавиатуры и получать результат на экране монитора. С этой целью рекомендуется обеспечить обмен данными между микро-ЭВМ и ПЭВМ по последовательному каналу. На ПЭВМ при этом возможно использование любой программы обмена, существующей в рамках ОС (например, Гипертерминал для Windows).

В альтернативных, более простом вариантах, допускается:

- параметры загружать вместе с программой, а результат выдавать в последовательный канал;

- считывать содержимое областей, хранящих результат, после запуска программы.

Программы для ассемблирования, создания исполнимого кода и прочие доступны на сервере кафедры.

6.3 Порядок выполнения работ

– Изучить принципы функционирования встроенного в ADuC ЦАП и формат управляющих регистров.

- Изучить демонстрационную программу на языке ассемблера.
- Оттранслировать программу с помощью кросс (макро)ассемблера ASEM или аналогичного.
- Создать файл с исполнимым кодом программы (например, программой hextobin)
- Загрузить программы для проверки работоспособности и функциональности макета.
- После запуска загрузчика выполните шаги для активизации встроенной программы связи по последовательному каналу:
- С помощью загрузчика, после установления и подтверждения связи, необходимо:
    - загрузить исполнимый файл в микро-ЭВМ;
    - запустить программу и убедиться в ее работоспособности.

6.4 Контрольные вопросы

- как нарушается линейность на выходе ЦАП при малых значениях кодов
- как нарушается линейность на выходе ЦАП при больших значениях кодов
- какова максимальная скорость изменения аналогового сигнала на выходе
- нужно ли приостанавливать работу программы при включении ЦАП

6.5 Содержание отчета

Отчет о лабораторной работе должен содержать:

- титульный лист;
- цель исследований;
- структурную схему макетной установки;
- текст и описание программы, содержательные комментарии;
- скриншоты основных этапов выполнения программы;
- протокол исследования работоспособности демонстрационной программы.

Приложение 6.1 Демонстрационная программа управления ЦАП

```
;*********************************************************************
;
; Author        : ADI - Apps            www.analog.com/MicroConverter
;
; Date          : 31 May 2002
;
; File          : DACquadr.asm
;
; Hardware      : ADuC831
;
; Description   : Outputs sine waves on DAC0 and DAC1 at 400Hz.
;                 Output signals are in quadrature with eachother,
;                 DAC1 leading DAC0 by 90 degrees.  since each DAC is
;                 updated when its DACxL register is written to, they
;                 are not updated at the exact same moment, and a
;                 phase error of (in this case) 0.625degrees results.
;                 to address this problem, see code: "DACsync.asm".
;                 Rate calculations assume an 11.0592MHz Mclk.
;
;*********************************************************************

$MOD831                         ; Use 8052&ADuC831 predefined symbols

LED     EQU     P3.4            ; P3.4 drives red LED on eval board

;_____________________________________________________________________
                                                    ; BEGINNING OF CODE
CSEG

ORG 0000h
       MOV    ADCCON1,#80H
        MOV     DACCON,#01Fh    ; both DACs on,12bit,asynchronous
        MOV     DAC0H,#008h
        MOV     DAC0L,#000h     ; DAC0 to mid-scale
        MOV     DAC1H,#00Fh
        MOV     DAC1L,#0FFh     ; DAC1 to full-scale

        MOV     DPTR,#TABLE

STEP:   CLR     A               ;                                       1
        MOVC    A,@A+DPTR       ; get high byte for mainDAC..           2
        MOV     DAC0H,A         ; ..and move it into DAC0 register      1
        MOV     A,#020h         ; offset by 90deg for quadratureDAC     1
        MOVC    A,@A+DPTR       ; get high byte for quadratureDAC..     2
        MOV     DAC1H,A         ; ..and move it into DAC1 register      1
        INC     DPTR            ; move on to get low bytes              2

        CLR     A               ;                                       1
        MOVC    A,@A+DPTR       ; get low byte for mainDAC..            2
        MOV     DAC0L,A         ; ..and update DAC0                     1
        MOV     A,#020h         ; offset by 90deg for quadratureDAC     1
        MOVC    A,@A+DPTR       ; get low byte for quadratureDAC..      2
        MOV     DAC1L,A         ; ..and update DAC1                     1
        INC     DPTR            ; move on for next data point           2

        ANL     DPL,#07Fh       ; wrap around at end of table           2

        MOV     A,DAC0H         ;                                       1
        MOV     C,ACC.3         ; MSB of DAC0 value                     1
        MOV     LED,C           ; LED = MSB of DAC0                     2

        NOP                     ;                                       1
        NOP                     ;                                       1
        NOP                     ;                                       1
```

```
        NOP                     ;                                       1
        NOP                     ;                                       1
        NOP                     ;                                       1
        NOP                     ;                                       1
        NOP                     ;                                       1

        JMP     STEP            ;                                       2

; numbers at right in the above loop represent the number of machine
; cycles for each instruction.  the complete loop takes exactly 36
; machine cycles.  with an 11.0592MHz master clock, a machine cycle
; is 1.085us, so the above loop takes 39.06us to update each data
; point.  since there are 64 data points in the below sine lookup
; table, this results in a 2.50ms period, i.e. a 400Hz frequency.

;_____________________________________________________________________
                                                        ; SINE LOOKUP TABLE
;_____________________________________________________________________

END
```

7 Управление последовательным каналом микро-ЭВМ ADuC

7.1 Цель работы

Получение практических навыков передачи данных по асинхронному последовательному каналу микро-ЭВМ при написании программ на языке ассемблера.

7.2 Методические указания

Более подробные данные о функционировании асинхронного последовательного приемо-передатчика приведены в [1].

В лабораторной работе передача данных исследуется в рамках демонстрационной программы (Приложение __.)

При возможности, целесообразно реализовать интерфейсную часть программы, позволяющую задавать параметры работы с клавиатуры и получать результат на экране монитора. С этой целью рекомендуется обеспечить обмен данными между микро-ЭВМ и ПЭВМ по последовательному каналу. На ПЭВМ при этом возможно использование любой программы обмена, существующей в рамках ОС (например, Гипертерминал для Windows).

В альтернативных, более простом вариантах, допускается:

- параметры загружать вместе с программой, а результат выдавать в последовательный канал;

- считывать содержимое областей, хранящих результат, после запуска программы.

Программы для ассемблирования, создания исполнимого кода и прочие доступны на сервере кафедры.

7.3 Порядок выполнения работ

- Изучить принципы функционирования встроенного в ADuC адаптера последовательного асинхронного канала и формат управляющих регистров.
- Изучить демонстрационную программу на языке ассемблера.
- Оттранслировать программу с помощью кросс (макро)ассемблера ASEM или аналогичного.
- Создать файл с исполнимым кодом программы (например, программой hextobin)
- Загрузить программы для проверки работоспособности и функциональности макета.
- После запуска загрузчика выполните шаги для активизации встроенной программы связи по последовательному каналу
- С помощью загрузчика, после установления и подтверждения связи, необходимо:
    - загрузить исполнимый файл в микро-ЭВМ;
    - запустить программу и убедиться в ее работоспособности.

7.4 Контрольные вопросы

от каких факторов зависит скорость обмена по последовательному каналу микро-ЭВМ ADuC?

какова максимальная скорость приема и передачи данных для микро-ЭВМ ADuC?

возможна ли работа в дуплексном режиме?

как зависит набор стандартных скоростей обмена от частоты тактирования микро-ЭВМ?

7.5 Содержание отчета

Отчет о лабораторной работе должен содержать:
- титульный лист;
- цель исследований;
- структурную схему макетной установки;
- текст и описание программы, содержательные комментарии;
- скриншоты основных этапов выполнения программы;
- протокол исследования работоспособности демонстрационной программы.

Приложение 7.1 Демонстрационная программа управления последовательным асинхронным каналом

```
;=====================================================================
;
; Author        : ADI - Apps
;
; Date          : May 2002
;
; File          : UART.asm
;
; Hardware      : ADuC831
;
; Description   : This Program saves 16 numbers in order initially
;                 starting with 0 into memory locations 40h to 50h.
;                 When finished the values in these locations are
;                 transmitted down the UART in ASCII form to the PC
;                 where they can be viewed using the preconfigured
;                 Hyperterminal program. (c:\ADuC_Beta832\9600com1.ht)
;
;                 After the transmission of the 16 bytes a 5 second
;                 delay is called and the process is repeated, this
;                 time starting with the saving of 10h to location
;                 40h.
;
;=====================================================================
;
$MOD831                                 ;Use 8052 predefined Symbols

LED    EQU    P3.4

;_____________________________________________________________________
                                                    ; BEGINNING OF CODE
CSEG
ORG 0000H

       JMP MAIN

ORG 0060H                    ; Start code at address above interrupts


MAIN:                        ; Main program

        MOV     T3CON,#85h
        MOV     T3FD,#08h
        MOV     SCON,#52h

        MOV     R0, #00       ; start output data at 0
        MOV     R1, #40h      ; initialise R1 to 40 to store the
                              ; input data from memory location 40
SAVENOS:
        MOV     A,R0
        MOV     @R1, A        ; move R0 into memory location R1
        INC     R1            ; increment memory location and data so
                              ; new data is stored in new address
        INC     R0
        CJNE    R1, #50H, SAVENOS  ; reset memory location to 40h
                                   ; when memory location reaches 50h
                                   ; saving 16 bytes of data

; Transmit the values in locations 40h->50h up the UART wait for
; 5 seconds and then repeat
```

```
START:  CPL     LED          ;CPL LED with each transmission
        MOV     DPTR, #TITLE
        CALL    SENDSTRING   ; write title block on screen

        MOV     R1, #40h     ; move value at address 40 into R2
        MOV     A, @R1
        MOV     R2, A

NEXT:                        ; Put new value on a new line
        MOV     A, #10       ; Transmit a linefeed (= ASCII 10)
        CALL    SENDCHAR
        MOV     A, #13       ;Transmit a carriage return (=ASCII 13)
        CALL    SENDCHAR

        MOV     A, R2        ; Transmit R2 i.e. value @ address R1
        CALL    SENDVAL
        INC     R1           ; Increment address
        MOV     A, @R1
        MOV     R2, A        ; R2 holds the value @ addrR1

        MOV     A, R1        ; Check if at address 50h
        CJNE    A, #50h, NEXT   ; if not jump to Next
        JMP     WAIT5S          ; if so wait 5s and repeat

WAIT5S: MOV     A, #50
        CALL    DELAY           ; Wait 5 seconds
        MOV     R1, #40h
        JMP     SAVENOS       ; Resave new numbers to same addresses


;_____________________________________________________________________
                                                          ; SENDSTRING

SENDSTRING:     ; sends ASCII string to UART starting at location
                ; DPTR and ending with a null (0) value

        PUSH    ACC
        PUSH    B
        CLR     A
        MOV     B,A
IO0010: MOV     A,B
        INC     B
        MOVC    A,@A+DPTR
        JZ      IO0020
        CALL    SENDCHAR
        JMP     IO0010
IO0020: POP     B
        POP     ACC

        RET

;_____________________________________________________________________
                                                          ; SENDCHAR

SENDCHAR:       ; sends ASCII value contained in A to UART

        JNB     TI,$            ; wait til present char gone
        CLR     TI              ; must clear TI
        MOV     SBUF,A

        RET

;_____________________________________________________________________
                                                          ; SENDVAL

SENDVAL:        ; converts the hex value of A into two ASCII chars,
             ; and then spits these two characters up the UART.
```

```
                ; does not change the value of A.

        PUSH    ACC
        SWAP    A
        CALL    HEX2ASCII
        CALL    SENDCHAR        ; send high nibble
        POP     ACC
        PUSH    ACC
        CALL    HEX2ASCII
        CALL    SENDCHAR        ; send low nibble
        POP     ACC

        RET


;_____________________________________________________________________
                                                            ; HEX2ASCII

HEX2ASCII:      ; converts A into the hex character representing the
                ; value of A's least significant nibble

        ANL     A,#00Fh
        CJNE    A,#00Ah,$+3
        JC      IO0030
        ADD     A,#007h
IO0030: ADD     A,#'0'

        RET

;_____________________________________________________________________
                                                              ; DELAY
DELAY:                          ; Delays by 100ms * A
                                ; 100mSec based on 11.0592MHZ
                                ; Core Clock

             MOV    R5,A        ; Acc holds delay variable
 DLY0:       MOV    R6,#200          ; Set up delay loop0
 DLY1:       MOV    R7,#229          ; Set up delay loop1
             DJNZ   R7,$        ; Dec R2 until R2 is zero
             DJNZ   R6,DLY1        ; Dec R1 & Jump DLY1 until R1 is 0
             DJNZ   R5,DLY0           ; Dec R0 & Jump DLY0 until R0 is 0
             RET                ; Return from subroutine
;_____________________________________________________________________


TITLE:    DB 10,10,13,'_____________________________________',10,13
          DB 'Analog Devices MicroConverter ADuC831',10,13
          DB '         UART Demo Routine',10,13
          DB '  Data Stored in Memory in Hex Form',10,13,0


END
```

8. Литература

1. ADUC8XX SAR EVALUATION BOARD REFERENCE GUIDE Analog Devices, Inc. www.analog.com/microconverter (в электронном виде на сервере кафедры)

2. MicroConverter®, 12-Bit ADCs and DACs with Embedded 62 kBytes Flash MCU ADuC831 www.analog.com/microconverter (в электронном виде на сервере кафедры)