Лабораторные работы. Микропроцессорные системы. Второй семестр изучения

Лабораторная работа №7

Использование модуля АЦП интегрированного в PIC18F4520. Работа с аналоговыми датчиками

Оборудование: компьютер с MS Widows XP или новее. Программное обеспечение MPLab 8.x. Стенд Кристалл 22М

Особенности отладки ADC в MPLab 8

Stimuls поставляемый в составе MPLab не позволяет имитировать аналоговый сигнал на входах микроконтроллера.

Вместо этого производителем предлагается использовать внедрение (injection) данных в регистр.

Для этого используется вкладка Register Injection

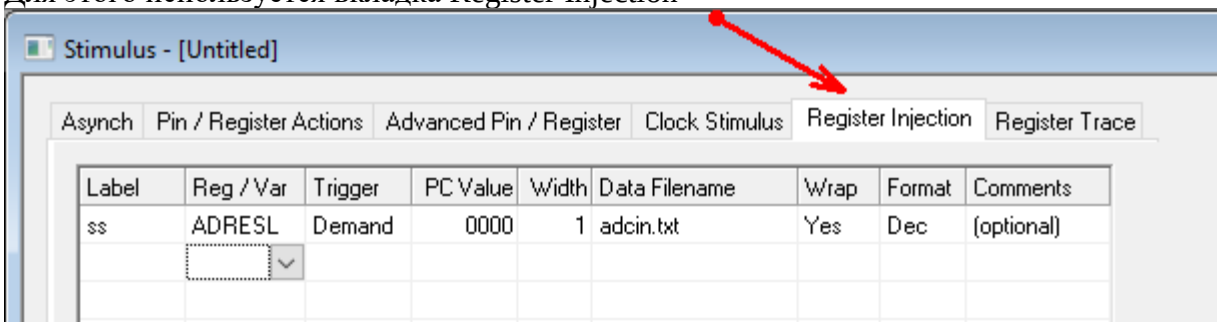

Рисунок 1 Окно Stimulus

Поскольку ADC помещает результаты преобразования в ADRESL:ADRESH то именно в эти регистры и будем производить injection.

Trigger – условие при котором выполняется injection мы используем Demand (по требованию). В этом случае injection будет выполнен, когда из указанного регистра будет попытка чтения данных.

Data Filename – путь к файлу в котором находятся внедряемые значения.

Wrap – перенос в начало после окончания чтения файла.

Format – формат данных в файле.

Не забываем нажать Apply!

Заранее подготовим файл с данными. Файл будет в формате Windows Text и его можно подготовить при помощи Блокнота.

Рекомендуется поместить файл в папку проекта.

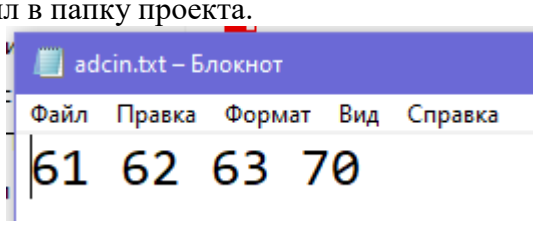

Рисунок 2 Пример файла

В коде программы:

Разрешаем использование выводов PortB<4:0> для ADC.

config OSC = HSPLL,WDT = OFF,LVP = OFF,MCLRE = ON,**PBADEN = ON**;

В основном коде (блок Main)

Настраиваем выводы на ввод

```
movlw b'00001111'    ; RA<0,3> - на ввод, остальные на вывод
movwf TRISA
```

Настраиваем АЦП

```
; ************** Настройка АЦП ************
        movlw b'00001101'; Установка RA<0,1> - аналоговый
        movwf ADCON1       ; Uоп = напряжение МК

        movlw b'10111110'
        movwf ADCON2

        movlw b'00000001'
        movwf ADCON0;
; ****** выбор канал AN1 и Urev-5B ******
        bcf     ADCON1,VCFG0     ; Urev=5V
        movlw b'00000101'    ; Выбор AN1
        movwf ADCON0                ;
```

Наш ADC будет включаться по Timer0
Настраиваем прерывание от Timer0 (аналогично предыдущим работам)

В обработчике прерывания от таймера:

```
        bcf INTCON, TMR0IE; выключаем прослушивание прерывания от таймера
        bsf     ADCON0,GO ; Запуск АЦП
        nop
        nop
        btfsc   ADCON0,GO ; Ждем АЦП преобразование циклически опрашивая бит готовности
        bra     $-.2
        nop                     ; АЦП преобразование завершено

        movff ADRESL,WREG; перемещаем результат в аккумулятор. Если входные значения в пределах 0-255 то ADRESH не используется.
```

Не забываем, то ADC должен сработать. Поэтому на вывод, связанный с его каналом даем сигнал. Например так

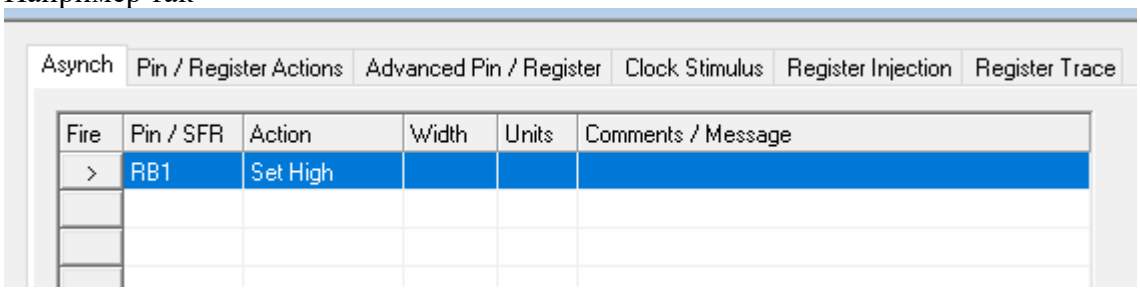

Рисунок 3 Сигнал на RB1

<u>Задание №1: Организация работы с АЦП</u>
Используя Datasheet микроконтроллера, материал на сайте и форумы сети Интернет изучите особенности и методы управления АЦП в PIC18 в том числе использование прерываний от АЦП.

<u>Задание №2: Использование АЦП</u>
Создайте программу, которая позволит получить данные с аналогового термодатчика и отображать их в двоичной форме при помощи светодиодов HL1-HL8.
Для работы использовать прерывание от АЦП
Например в качестве термодатчика можно использовать TC1047A,

<u>Задание №3: Использование АЦП</u>
Создайте программу, которая будет включать нагревательный элемент если температура менее 10С и выключать, если температура более 80С.
Предусмотреть включение сигнала тревоги, если температура более 100С.

Лабораторные работы. Микропроцессорные системы. Второй семестр изучения
Сигнал включается при подаче низкого уровня на RC1.
Нагреватель включается при подаче высокого уровня на RC2
Канал подключения датчика выбрать самостоятельно

Защита работы:
Для успешной защиты данной работы необходимо иметь знания по темам:

- Как работает АЦП В микроконтроллере
- Регистры микроконтроллера, их виды.
- Модель прерываний
- Логику программирования прерываний
- Приоритеты прерываний
- Регистры управления прерываниями и значения битов в этих регистрах
- Методику использование семисегментных индикаторов

Приложение А

Алгоритмы и примеры работы с АЦП

Статья взята из рассылки «Микроконтроллеры PIC фирмы Microchip для начинающих» выпуск №26 созданной для сайта http://www.2aplusa.ru/

Сегодня продолжаем рассматривать периферийный модуль АЦП микроконтроллера PIC18F452! Этот выпуск будет посвящен алгоритмам и примерам работы с АЦП.

При разработке собственного устройства можете использовать данные примеры.
Итак, начнем со словесного описания алгоритма работы:

1. **Необходимо настроить сам периферийный модуль АЦП.** Настроить аналоговые входы, сконфигурировать входы опорного напряжения, выбрать измеряемый канал. Настроить частоту преобразования. Период преобразования **Tad** выбирается из следующих соображений, чтобы он был не менее 1,6 мкс. Иначе, результат может оказаться неверным. Поэтому, делитель ADCS2:0 необходимо правильно настроить, исходя из тактовой частоты микроконтроллера! На одно преобразование аналогового сигнала модулю АЦП требуется 12 **Tad**. Если модуль АЦП предполагается использовать в SLEEP «энергосберегающем, спящем» режиме микроконтроллера, то делителем ADCS2:0 необходимо выбрать тактирование АЦП от собственного внутреннего RC-генератора АЦП (**Tad** = 2-6 мкс).
   Приведу пример расчета делителя. Например, если тактовая частота микроконтроллера равна 40МГц – это период = 0,000000025сек. Теперь, делим 1,6мкс/0,025мкс=64. Поэтому, выбираем битами ADCS2:0 значение Fosc/64. И мы получим **Tad**, равное 1,6 мкс. Уже подсчитанное значение делителя для номинального периода **Tad**при наиболее часто встречающихся тактовых частотах кварца можно найти в таблице 17-1 в datasheet на наш микроконтроллер.
2. **Включить модуль АЦП.**
3. **Настроить прерывание от АЦП, если оно необходимо в программе.**
4. **Выждать время acquisition time.** Это время рассчитывается в datasheet и при стандартных условиях (выходное сопротивление источника аналогового сигнала=2,5кОм, максимальная рабочая температура = 50 °C, погрешность – ? младшего бита результата преобразования) равно 12.86 мкс. Но, это время можно и увеличить. Данное время необходимо, чтобы зарядился конденсатор, находящийся в модуле преобразования. Потому что, реально, аналоговый сигнал «запоминается» на конденсаторе, а потом входной аналоговый сигнал отключается и преобразователь работает только с этим запомненным сигналом. Конденсатор этот называется **Chold**.
5. **Запускаем преобразование.** Преобразование начинается установкой в единицу бита GO/DONE регистра ADCON0. После установки этого бита, конденсатор **Chold** отключается от входного аналогового сигнала, и преобразователь работает с «запомненным» на **Chold** аналоговым сигналом. Как мы помним, на преобразование затрачивается 12 периодов **Tad**.
6. **Ждем автоматического сброса бита GO/DONE или прерывания от АЦП.** Как только происходит прерывание или сброс бита, конденсатор **Chold**, автоматически, обратно подключается к аналоговому входному сигналу.
7. **Читаем полученное значение преобразования.** Из регистров ADRESH:ADRESL читаем выровненный влево или вправо результат.
8. **Переключаем мультиплексор на другой измеряемый канал.** Если измеряем только один канал, то с мультиплексором ничего не делаем. Если каналов несколько, то выбираем следующий канал, настраивая на него мультиплексор АЦП.

9. **Ждем, когда зарядится конденсатор Chold.** Это время должно быть не меньше 2-х периодов **Tad**, чтобы новое аналоговое значение успело «запомниться» на конденсаторе. А потом, повторяем всю последовательность действий с пункта 5.

А теперь примеры на ассемблере.

Как мы помним по нашей схеме из предыдущих выпусков рассылки, у нас тактовая частота задается кварцем и равна 20МГц. Находим в datasheet по таблице 17-1 номинальное значение делителя для такой тактовой частоты – оно равно 32. Таким образом, период **Tad** будет равен: 32/20000000= 0,0000016 сек или **1,6 мкс**.

```
; Настраиваем модуль АЦП
      MOVLW       0x8E                ; опорное напряжение равно напряжению микроконтроллера, используется только вход AN0, правое выравнивание
      MOVWF       ADCON1, ab
      MOVLW       0x81                ; делитель - Fosc/32, выбираем на мультиплексор канал AN0, и включаем АЦП
      MOVWF       ADCON0, ab
      BCF         PIE1, ADIE, ab      ; прерывание использовать не будем
; ждем  когда зарядиться конденсатор Chold (acquisition time=12,86мкс)
; на каждую команду требуется 200нсек (4/20000000), таким образом, 12,86/0,2=65 - столько команд потребуется, чтобы выдержать интервал в 12,86 мкс
; организуем программную задержку (пусть и будет погрешность, но она будет в сторону увеличения времени, а это не критично)
    MOVLW       0x11
    MOVWF       temp, ab            ; счетчик циклов
Wait_aq:
    NOP
    NOP
    DECFSZ      temp, rf, ab        ; Цикл выполняется за 4-5 команд
    BRA         Wait_aq             ; поэтому, грубо 4*17=72 команды
Start_ADC:
    BSF         ADCON0, GO_DONE, ab ; старт преобразования

    BTFSC       ADCON0, GO_DONE, ab ; ждем пока не сброситься данный бит
    BRA         $-2                 ; возврат на одну команду назад (-2, потому что каждая команда занимает в памяти программ 2 байта)
; читаем результат
    MOVFF       ADRESH, ADC_value_h
    MOVFF       ADRESL, ADC_value_l
   ; измеряемый канал один – AN0, поэтому мультиплексор не трогаем
   ; ждем 2 периода Tad = 1,6*2/0,2 = 16 команд, пока заряжается конденсатор
    MOVLW       0x04
    MOVWF       temp, ab            ; счетчик циклов
Wait_Chold:
    NOP
    NOP
    DECFSZ      temp, rf, ab        ; Цикл выполняется за 4-5 команд
    BRA         Wait_Chold          ; поэтому, грубо 4*4=16 команд
    CLRWDT
    BRA         Start_ADC           ; начинаем следующее преобразование
```

Приложение Б

Пример кода для работы с семисегментными индикаторами и АЦП

Поставляется со стендом Кристалл - 22м 22м - \Prog_MPlab\ASM_PROG\prog51.txt

```
        title "Чтение с Rp1,АЦП"
        list      p=18F4520             ; Тип процессора
        #include<P18F4520.INC>     ; Подключение файла

        cblock 0x00
STATUS_TEMP,W_TEMP
Hg1,Hg2,Hg3,Hg4,Hg_DG
        Endc


; Значения констант для TMR1 "Динамическая индикация"
T1H_const=0xFB
T1L_const=0xF9


; Схема соединений!!!
; PORTD на семисегментный индикатор (сегменты)
; RD0-a, RD1-b, RD2-c, RD3-d, RD4-e, RD5-f, RD6-g, RD7-h
; PORTB<0-3> - на аноды семисегментного индикатора
; RB0 - DG1, RB1 - DG2, RB2 - DG3, RB3 - DG4
; RA0 - RP1 подключение переменного резистора



        org     0x00    ; Вектор сброса
        bra     Start

        org     0x08    ; Вектор прерывания высокий приоритет
        rcall   Int_H   ; Переход на подпрограмму обработки прерывания
        retfie  S



        org     0x18    ; Вектор прерывания низкий приоритет
        movwf           W_TEMP                          ; Сохранение W_
        movff   STATUS,STATUS_TEMP    ; Сохранение STATUS_TEMP
        nop
        btfss   PIR1,TMR1IF ; Проверка на прерывание от TMR1?
        bra     Contin_L        ; не от TMR3, прерывание от другого источника
        bcf     PIR1,TMR1IF ; Произошло прерывание от TMR1, сброс флага
        rcall   HG_OUT        ; Вывод на СДИ HG

        bra     End_L
; прерывание от другого источника
Contin_L
        nop
        nop
        nop
; ------------------------------
End_L
```

```
        movf   W_TEMP,W             ; Восстановление WREG
        movff  STATUS_TEMP,STATUS    ; Восстановление STATUS
        retfie

; *****          подпрограмма чтения массива *****
Tab_HG
        andlw  0x0F    ; наложение маски, для повышения надежности
        rlncf    WREG; умножение на 2 т.к. шина команд 16 бит
        addwf  PCL,f  ; Сложение с программным счетчиком команд
        retlw   0x3F   ; Значение для цифры 0
        retlw   0x06   ; Значение для цифры 1
        retlw   0x5B   ; Значение для цифры 2
        retlw   0x4F   ; Значение для цифры 3
        retlw   0x66   ; Значение для цифры 4
        retlw   0x6D   ; Значение для цифры 5
        retlw   0x7D   ; Значение для цифры 6
        retlw   0x07   ; Значение для цифры 7
        retlw   0x7F   ; Значение для цифры 8
        retlw   0x6F   ; Значение для цифры 9
        retlw   0x77   ; Значение для цифры A
        retlw   0x7C   ; Значение для цифры b
        retlw   0x39   ; Значение для цифры C
        retlw   0x5E   ; Значение для цифры d
        retlw   0x79   ; Значение для цифры E
        retlw   0x71   ; Значение для цифры F
; ----------------------------------------

; ***** подпрограмма обработки прерывания высокий приоритет *****
Int_H:


End_H
        return  ; Выход из подпрограммы прерывания
; ------------------------------------------------------------

Start:
        clrf     PORTA
        clrf     PORTB
        clrf     PORTC

        movlw b'00000001'    ; RA<0> - на ввод, остальные на вывод
        movwf PORTA

        movlw b'00001110'; Установка RA<0> - аналоговый
        movwf ADCON1       ; Uопорное = Uпит МК

        movlw b'10111010'
        movwf ADCON2

        movlw b'00000001'
        movwf ADCON0

        movlw b'00110000'    ; Настроили RB<0-3> - на вывод
        movwf TRISB          ; RB<4,5> на ввод
```

```
        clrf    TRISD           ; все на вывод

        rcall   Tmr1_Init
        rcall   Reload1

        movlw 0x01
        movwf Hg_DG
        lfsr    FSR0,Hg1
        clrf    Hg1
        clrf    Hg2
        clrf    Hg3
        clrf    Hg4

        bsf     RCON,IPEN   ; разрешена приоритетная система прерываний

        bcf     IPR1,TMR1IP ; прерывание от TMR1 - низкий приоритет
        bcf     PIR1,TMR1IF ; сброс флага прерывания
        bsf     PIE1,TMR1IE ; разрешить прерывание от TMR1


        bsf     INTCON,GIEH
        bsf     INTCON,GIEL



Main:
        bsf     ADCON0,GO ; Запуск АЦП
        nop
        btfsc   ADCON0,GO ; Ждем АЦП преобразование
        bra     $-.2
        movff   ADRESH,xh   ; АЦП преобразование завершено!
        movff   ADRESL,xl

        rcall   Bin16_Bcd
        movff   num1,Hg1
        movff   num2,Hg2
        movff   num3,Hg3
        movff   num4,Hg4

Contin
        nop
        nop
        bra     Main

; ********** ПОДПРОГРАММЫ **********
; ***** конфигурация TMR1 *****
Tmr1_Init
        movlw b'10110001'
        movwf T1CON
        clrf    TMR1H
        clrf    TMR1L
        return
; ----------------------------
; загрузка таймера TMR1 константами
Reload1:
```

```
        movlw T1H_const
        movwf TMR1H
        movlw T1L_const
        movwf TMR1L
        return
; ------------------------------


HG_OUT
        rcall   Reload1 ; Перезагрузка TMR1
        bcf     PORTB,0
        bcf     PORTB,1
        bcf     PORTB,2
        bcf     PORTB,3
; ***** Проверка на переполнение разряда *****
        btfss   Hg_DG,4
        bra     $+.10
        movlw 0x01
        movwf Hg_DG
        lfsr    FSR0,Hg1
; ---------------------------------------------
        nop
; ***** Вывод символа разряда *****
        movf    POSTINC0,w ; W=Hg[1..4]
        rcall   Tab_HG      ; Подпрограмма чтение массива
        movwf PORTD         ; вывод константы символа в PORTD
;; Для вывода точки
;       btfss   Hg_DG,1     ; тест разряда DG2
;       bra     $+.8
;       bcf     PORTD,7     ; сброс сегмента h
;       btfss   HGh   ; тест флага точки HGh
;       bsf     PORTD,7     ; установка сегмента h
;; --------------------------------
        nop
; ***** Зажигаем разряд HG<DG1> *****
        movf    Hg_DG,w
        iorwf   PORTB,f
        rlncf   Hg_DG,f
; ----------------------------------
        nop
        return


        #include "LCD.inc"

        end
```