


# Обучение : Программирование на Ассемблере для PIC

### Пример 9. Работа с энергонезависимой памятью (ПЗУ)

Энергонезависимая память обладает свойством сохранять свои данные при выключении питания. Примеры реализации этого свойства можно встретить в бытовой технике: продолжение работы стиральной машинки по заданной программе, музыкальные центры сохраняют настройки уровня громкости и тембра, телевизоры сохраняют настройки каналов и т.п.

Также мы предполагаем, что разработчики интеллектуальных устройств на МК используют энергонезависимую память для сознательного ограничения срока работоспособности, тем самым преследуя маркетинговые цели продвижения новых товаров на МК. Нет никаких препятствий для внедрения в программу счетчиков количества включения устройства и/или счетчиков времени непрерывной работы устройства. Исчерпав лимит времени устройство в лучшем случае начнет вести себя неадекватно ("глючить"), либо перестанет функционировать вообще.

Далее для простоты восприятия энергонезависимую память в этом тексте мы будем обозначать собирательным термином ПЗУ – постоянное запоминающее устройство. ПЗУ в МК традиционно обозначают фразой EEPROM-память.

Объем ПЗУ не велик: для большинства ПИКов (в том числе и для PIC16F84A) она составляет 64 байта, т.е. 64 регистра, в каждый из которых может быть записан один байт (число от .00 до .255). В дальнейшем значение каждого регистра ПЗУ может быть считано и использовано при исполнении программы. Все регистры ПЗУ имеют свой адрес в диапазоне адресов от .00 до .63 (00h – 3Fh), который используется как при чтении данных из ПЗУ, так и при записи данных в ПЗУ. При записи байта автоматически стирается предыдущее значение и записываются новые данные (стирание перед записью). Все эти операции производит встроенный автомат записи ПЗУ.

Обратите внимание, что числа, определяющие диапазон адресов регистров ПЗУ совпадают с числами диапазона адресов регистров общего и специального назначения (ОЗУ). Это совершенно не должно нас беспокоить, т.к. ПЗУ не принадлежит области регистров ОЗУ.

Существует два варианта записи данных в ПЗУ:
1) предварительная запись до начала исполнения программы;
2) в ходе исполнения программы.

Далее рассмотрим ключевые сегменты программы, определяющие работу ПЗУ.

```
; текст в шапке программы
; Определение регистров для работы с памятью (операции чтения/записи)
EEData      equ         08h         ; EEPROM - данные
EECon1      equ         08h         ; EECON1 - банк1.
EEAdr       equ         09h         ; EEPROM - адрес
EECon2      equ         09h         ; EECON2 - банк2.
; =====
```

Это сегмент шапки программы. Доступ к ПЗУ осуществляется через два регистра: EEDATA <08h>, который содержит в себе восьмибитовые данные для чтения/записи и EEADR <09h>, который содержит в себе адрес ячейки к которой идет обращение. Дополнительно имеется два управляющих регистра: EECON1 <88h> и EECON2 <89h> (не забываем особенности сопоставления имен и чисел) (вспоминаем формат записи чисел).

```
; ... шапка
; предварительная запись
            org         2100h       ; Обращение к EEPROM памяти данных.
            DE          8h,4h,2h
            DE          1h,3h,5h,7h,8h,6h,4h,2h
; =====
            org         0           ; начало программы
; ... текст программы
```

Этот сегмент программы вставляется между шапкой и рабочей частью программы. Для работы с ПЗУ наличие этого сегмента необязательно, т.к. его задача – предварительная запись определенных значений в ячейки ПЗУ.

Директива org 2100h производит "включение" (разрешение) предварительной записи в ПЗУ.

Далее директива DE 8h,4h,2h произведёт запись трёх чисел в первые три ячейки ПЗУ с адресами 0h, 1h и 2h. В эти ячейки будут записаны числа 8, 4 и 2.

Далее директива DE 1h,3h,5h,7h,8h,6h,4h,2h произведёт запись восьми чисел в следующие восемь ячеек ПЗУ: в третью – 1, четвертую – 3, пятую – 5 … десятую – 2.

Таким образом, с помощью одной директивы за один раз можно заполнить все 64 регистра ПЗУ, сделав запись в одну строчку через запятую.

Далее традиционный сегмент записи в ПЗУ.

```
; запись данных в энергонезависимую память EEPROM (ПЗУ)
            clrf        INTCON      ; Глобальный запрет прерываний

            movf        Adres,W     ; Записать в регистр W значение Adres
            movwf       EEAdr       ; Скопировать W в регистр EEAdr

            movf        Znach,W     ; Скопировать Znach в регистр W
            movwf       EEData      ; Скопировать W в EEPROM
            bsf         STATUS,RP0  ; Переход в первый банк.
            bsf         EECon1,2    ; Разрешить запись.

            movlw       55h         ; Обязательная
            movwf       EECon2      ; процедура
            movlw       AAh         ; при записи.
```

```
            movwf       EECon2      ; ----"----
            bsf         EECon1,1    ; ----"----
            bcf         EECon1,4    ; Сбросить флаг прерывания по окончании
            bcf         STATUS,RP0  ; Переход в нулевой банк.
```

Запрет прерываний – это, упрощенно говоря, действие необходимое для защиты МК от внешних сигналов, которые могут помешать записи данных.

Из регистра Adres мы указываем адрес регистру EEAdr, а из регистра Znach мы копируем наши данные в регистр EEData. Таким образом, мы указываем куда и что записать.

Далее разрешается запись и следуют несколько обязательных строчек, определенных разработчиками Microchipa.

Рассмотрим сегмент чтения данных из памяти ПЗУ.

```
; чтение данных из энергонезависимой памяти EEPROM (ПЗУ)
            bcf         STATUS,RP0  ; Переход в нулевой банк.
            movf        Adres,W     ; Записать в регистр W значение Adres
            movwf       EEAdr       ; Скопировать W в регистр EEAdr
            bsf         STATUS,RP0  ; Переход в первый банк.
            bsf         EECon1,0    ; Инициализировать чтение.
            bcf         STATUS,RP0  ; Переход в нулевой банк.
            movf        EEData,W    ; Скопировать в W из EEPROM
            movwf       Znach       ; Скопировать из W в регистр Znach
```

Из регистра Adres мы указываем адрес регистру EEAdr.
Затем следует процедура чтения после которой в регистр EEData копируется значение из ПЗУ.
Последними двумя строчками копируем значение в регистр Znach.

Теперь работу с памятью рассмотрим на практике.

Используя пример 7 можно поставить и решить две задачи:
– при включении прибора выводить на индикатор последнее значение, которое было перед выключением;
– в ПЗУ хранить данные для счётчика, на основании которых после 5-го включения блокируется работа прибора.

```
            LIST        P=PIC16F84A
            __CONFIG    H3FF1
W           EQU         0
F           EQU         1
PC          EQU         H0002
STATUS      EQU         H0003
PORTA       EQU         H0005
PORTB       EQU         H0006
TRISA       EQU         H0005
TRISB       EQU         H0006
INTCON      EQU         H000B
C           EQU         0
Z           EQU         2
Reg_1       EQU         H000C
Reg_2       EQU         H000D
Reg_3       EQU         H000E
Reg_4       EQU         H000F     ; регистр под результат
; Определение регистров для работы с памятью (операции чтения/записи)
EEData      equ         08h         ; EEPROM - данные
EECon1      equ         08h         ; EECON1 - банк1.
EEAdr       equ         09h         ; EEPROM - адрес
EECon2      equ         09h         ; EECON2 - банк2.
            org         2100h       ; Обращение к EEPROM памяти данных.
            DE          5           ; предварительная запись
            org         0           ; начало программы
; подготовительные моменты
            bsf         STATUS,5    ; переход в Банк 1
            movlw       b00011111
            movwf       TRISA
            clrf        TRISB
            bcf         STATUS,5    ; переход назад в Банк 0
; чтение значения из памяти и отрисовка
            call        Read        ; чтение числа
            movf        Reg_4,W
            call        TABLE
            movwf       PORTB
; отслеживание нажатий кнопок
m3          btfss       PORTA,2     ; бит-проверка ножки RA2 - уменьшение
            goto        m1
            btfss       PORTA,3     ; бит-проверка ножки RA3 - увеличение
            goto        m2
            goto        m3          ; зацикливание проверки
; проверка на ноль (на крайнее значение) и уменьшение значения регистра
m1          bcf         STATUS,Z    ; опустим флаг Z в ноль
            movf        Reg_4,F     ; копировать из Reg_4 в Reg_4
            btfsc       STATUS,Z    ; делаем бит-проверку Z-флага
; если Z=1, то выполняется следующая инструкция, иначе - пропускается
            goto        m4          ; переходим на отрисовку значения
            decf        Reg_4,F     ; уменьшить значение на 1 и сохранить
            goto        m4
; проверка на 9 (на др. крайнее значение) и увеличение значения регистра
m2          bcf         STATUS,C    ; опускаем флаг С в ноль
            movlw       .247        ; (255-9)+1 = 247 -> W
            addwf       Reg_4,W     ; (Reg_4)+W
            btfss       STATUS,C ; делаем бит-проверку C-флага
; если бит С=0, то выполняется следующая инструкция
; если бит С=1, то следующая инструкция пропускается
            goto        m5
            goto        m4
m5          incf        Reg_4,F     ; увеличить значение на 1 и сохранить
m4          movf        Reg_4,W
            call        TABLE
            movwf       PORTB
            call        Pause
            call        Write       ; запись числа
            goto        m3
;===================================
TABLE       addwf       PC,F        ; Содержимое счетчика команд PC = PC + W
            retlw       b01101111 ; 0
            retlw       b00001100 ; 1
```

```
            retlw      b01011011 ; 2
            retlw      b01011110 ; 3
            retlw      b00111100 ; 4
            retlw      b01110110 ; 5
            retlw      b01110111 ; 6
            retlw      b01001100 ; 7
            retlw      b01111111 ; 8
            retlw      b01111110 ; 9
;====================================
;delay = 500000 machine cycles
Pause       movlw       .85
            movwf       Reg_1
            movlw       .138
            movwf       Reg_2
            movlw       .3
            movwf       Reg_3
wr          decfsz      Reg_1, F
            goto        wr
            decfsz      Reg_2, F
            goto        wr
            decfsz      Reg_3, F
            goto        wr
            return
;====================================
```

**; запись данных в энергонезависимую память EEPROM (ПЗУ)**

```
Write       clrf        INTCON      ; Глобальный запрет прерываний
            clrw                    ; обнулить W, т.е. установить адрес "0"
            movwf       EEAdr       ; Скопировать W в регистр EEAdr
            movf        Reg_4,W     ; Скопировать данные из Reg_4 в регистр W
            movwf       EEData      ; Скопировать W в EEPROM
            bsf         STATUS,5    ; Переход в первый банк.
            bsf         EECon1,2    ; Разрешить запись.
            movlw       55h         ; Обязательная
            movwf       EECon2      ; процедура
            movlw       AAh         ; при записи.
            movwf       EECon2      ; ----"----
            bsf         EECon1,1    ; ----"----
            bcf         EECon1,4    ; Сбросить флаг прерывания по окончании
            bcf         STATUS,5    ; Переход в нулевой банк.
            return
;====================================
```

**; чтение данных из энергонезависимой памяти EEPROM (ПЗУ)**

```
Read        bcf         STATUS,5    ; Переход в нулевой банк.
            clrw                    ; обнулить W, т.е. установить адрес "0"
            movwf       EEAdr       ; Скопировать W в регистр EEAdr
            bsf         STATUS,5    ; Переход в первый банк.
            bsf         EECon1,0    ; Инициализировать чтение.
            bcf         STATUS,5    ; Переход в нулевой банк.
            movf        EEData,W    ; Скопировать в W из EEPROM
            movwf       Reg_4       ; Скопировать из W в регистр Reg_4
            return
;====================================
            end                     ; конец программы
```

Далее текст прошивки:

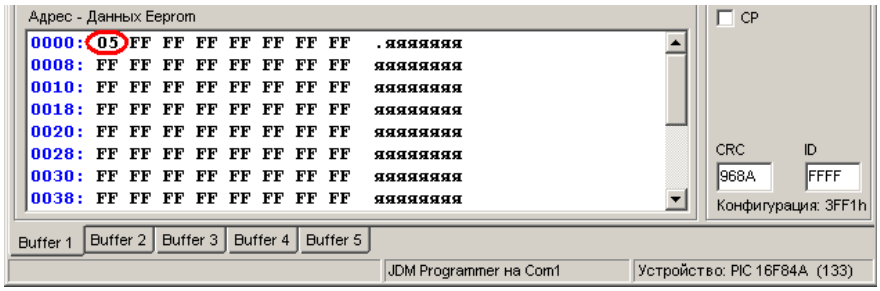
```
:020000040000FA
:1000000083161F3085008601831248200F082120A7
:100010008600051D0E28851D1428092803118F0848
:1000200003191B288F031B280310F7300F07031C2D
:100030001A281B288F0A0F08212086002C2039201F
:10004000092882076F340C345B345E343C347634D8
:1000500077344C347F347E3455308C008A308D00B8
:1000600003308E008C0B32288D0B32288E0B3228F9
:1000700008008B01030189000F088800831608150A
:10008000553089000AA3089008814081283120800AC
:1000900083120301890083160814831208088F0055
:0200A000080056
:02400E00F13F80
:024200000500B7
:00000001FF
```

Несложно убедиться в незначительности сделанных модификаций текста программы. Тем не менее, это новый качественный подход в работе устройства. Прокомментируем работу нашей программы.

В шапке программы мы прописали регистры для работы с ПЗУ. Между шапкой и текстом программы стоит пара команд предварительной записи. Для чего она нужна? После прошивки и первого включения МК содержимое регистров ПЗУ нам неизвестно; также и МК туда еще не успел сделать запись числа (чисел). Для того чтобы точно знать содержимое в момент первого включения (не путать с повторным включением) мы путем предварительной записи помещаем в нулевой регистр ПЗУ число 5. Глядя на этот вопрос с другой стороны, мы можем в начале программы организовать проверку содержимого нулевого регистра ПЗУ и если оно не входит в диапазон 0…9 прописать туда из программы нужное число, например 5. Мы решили не усложнять алгоритм программы, т.к. после повторного включения в регистре ПЗУ будет находиться "правильное" число.

Итак, предварительная запись числа (чисел) в ячейку ПЗУ – это ни что иное, как процедура конкретного определения содержимого ячеек ПЗУ. Запись чисел в ПЗУ делается в момент прошивки МК из программы IC-Prog. Строчки предварительной записи не входят в текст программы, они указывают – что и куда записать. Тем не менее, данные, помещенные в ПЗУ в ходе предварительной записи, могут сразу использоваться для расчетов и только потом из программы изменяться.

После открытия фала прошивки (пункт 10) в окне данных EEPROM мы видим предварительно записанную нами пятерку в ОЗУ (в шестнадцатеричном формате):




Необходимо отметить, что прошиваемые данные EEPROM (данные ПЗУ) из оболочки IC-Prog могут быть изменены вручную. Попробуйте вставить любое число в нулевую ячейку EEPROM из диапазона 00…09, прошить МК и сделать первое включение. Если вы поняли зависимость ваших действий, следовательно, вы овладели новыми знаниями, а именно как без строчек предварительной записи заполнить ПЗУ из IC-Prog!

Программный код также можно менять из IC-Prog, но для этого необходимы знания машинного кода. Увы, у нас таких знаний нет.
Предлагаю вам выключить устройство в тот момент, когда на индикаторе у нас отображен символ девяти "9".

А теперь МК вставим в программатор и считаем содержимое нашего МК из IC-Prog. Нажимаем кнопку  "Читать микросхему". Теперь смотрим в окно области данных EEPROM и обнаруживаем запись числа 09.

Все остальные действия в программе по работе с ПЗУ ранее прокомментированы.

Далее рассмотрим программу, которая после 5-го включения блокирует работу прибора (из примера 7) и выдает короткие звуковые сигналы.

```
            LIST        P=PIC16F84A
            __CONFIG    H3FF1
W           EQU         0
F           EQU         1
PC          EQU         H0002
STATUS      EQU         H0003
PORTA       EQU         H0005
PORTB       EQU         H0006
TRISA       EQU         H0005
TRISB       EQU         H0006
INTCON      EQU         H000B
C           EQU         0
Z           EQU         2
Reg_1       EQU         H000C
Reg_2       EQU         H000D
Reg_3       EQU         H000E
Reg_4       EQU         H000F     ; регистр под результат
; Определение регистров для работы с памятью (операции чтения/записи)
EEData      equ         08h         ; EEPROM - данные
EECon1      equ         08h         ; EECON1 - банк1.
EEAdr       equ         09h         ; EEPROM - адрес
EECon2      equ         09h         ; EECON2 - банк2.
            org         2100h       ; Обращение к EEPROM памяти данных.
            DE          5           ; предварительная запись
            org         0           ; начало программы
; подготовительные моменты
            bsf         STATUS,5    ; переход в Банк 1
            movlw       b00011111
            movwf       TRISA
            clrf        TRISB
            bcf         STATUS,5    ; переход назад в Банк 0
; чтение значения из памяти
            call        Read
; проверка числа кол-ва включений на равенство нулю
            bcf         STATUS,Z    ; опустим флаг Z в ноль
            movf        Reg_4,F     ; копировать из Reg_4 в Reg_4
            btfsc       STATUS,Z    ; делаем бит-проверку Z-флага
; если Z=1, то выполняется следующая инструкция, иначе – пропускается
            goto        m6
            goto        m7
; зацикливание для случая ограничения работы (звуковые сигналы)
m6          clrf        PORTB
            call        Pause
            bsf         PORTB,7
            call        Pause
            goto        m6
; уменьшение на единицу кол-ва включений
m7          decf        Reg_4,F
            call        Write
; выход в режим нормальной работы
            movlw       b01101111
            movwf       PORTB
            clrf        Reg_4
; отслеживание нажатий кнопок
m3          btfss       PORTA,2     ; бит-проверка ножки RA2 - уменьшение
            goto        m1
            btfss       PORTA,3     ; бит-проверка ножки RA3 - увеличение
            goto        m2
            goto        m3          ; зацикливание проверки
; проверка на ноль (на крайнее значение) и уменьшение значения регистра
m1          bcf         STATUS,Z    ; опустим флаг Z в ноль
            movf        Reg_4,F     ; копировать из Reg_4 в Reg_4
            btfsc       STATUS,Z    ; делаем бит-проверку Z-флага
; если Z=1, то выполняется следующая инструкция, иначе - пропускается
            goto        m4          ; переходим на отрисовку значения
            decf        Reg_4,F     ; уменьшить значение на 1 и сохранить
            goto        m4
; проверка на 9 (на др. крайнее значение) и увеличение значения регистра
m2          bcf         STATUS,C    ; опускаем флаг C в ноль
            movlw       .247        ; (255-9)+1 = 247 -> W
            addwf       Reg_4,W     ; (Reg_4)+W
            btfss       STATUS,C ; делаем бит-проверку C-флага
; если бит C=0, то выполняется следующая инструкция
; если бит C=1, то следующая инструкция пропускается
```

```
            goto        m5
            goto        m4
m5          incf        Reg_4,F     ; увеличить значение на 1 и сохранить
m4          movf        Reg_4,W
            call        TABLE
            movwf       PORTB
            call        Pause
            call        Write       ; вставка записи
            goto        m3
;====================================
TABLE       addwf       PC,F        ; Содержимое счетчика команд PC = PC + W
            retlw       b01101111 ; 0
            retlw       b00001100 ; 1
            retlw       b01011011 ; 2
            retlw       b01011110 ; 3
            retlw       b00111100 ; 4
            retlw       b01110110 ; 5
            retlw       b01110111 ; 6
            retlw       b01001100 ; 7
            retlw       b01111111 ; 8
            retlw       b01111110 ; 9
;====================================
;delay = 500000 machine cycles
Pause       movlw       .85
            movwf       Reg_1
            movlw       .138
            movwf       Reg_2
            movlw       .3
            movwf       Reg_3
wr          decfsz      Reg_1, F
            goto        wr
            decfsz      Reg_2, F
            goto        wr
            decfsz      Reg_3, F
            goto        wr
            return
;====================================
; запись данных в энергонезависимую память EEPROM (ПЗУ)
Write       clrf        INTCON      ; Глобальный запрет прерываний
            clrw                    ; обнулить W, т.е. установить адрес "0"
            movwf       EEAdr       ; Скопировать W в регистр EEAdr
            movf        Reg_4,W     ; Скопировать данные из Reg_4 в регистр W
            movwf       EEData      ; Скопировать W в EEPROM
            bsf         STATUS,5    ; Переход в первый банк.
            bsf         EECon1,2    ; Разрешить запись.
            movlw       055h        ; Обязательная
            movwf       EECon2      ; процедура
            movlw       0AAh        ; при записи.
            movwf       EECon2      ; ----"----
            bsf         EECon1,1    ; ----"----
            bcf         EECon1,4    ; Сбросить флаг прерывания по окончании
            bcf         STATUS,5    ; Переход в нулевой банк.
            return
;====================================
; чтение данных из энергонезависимой памяти EEPROM (ПЗУ)
Read        bcf         STATUS,5    ; Переход в нулевой банк.
            clrw                    ; обнулить W, т.е. установить адрес "0"
            movwf       EEAdr       ; Скопировать W в регистр EEAdr
            bsf         STATUS,5    ; Переход в первый банк.
            bsf         EECon1,0    ; Инициализировать чтение.
            bcf         STATUS,5    ; Переход в нулевой банк.
            movf        EEData,W    ; Скопировать в W из EEPROM
            movwf       Reg_4       ; Скопировать из W в регистр Reg_4
            return
;====================================
            end                     ; конец программы
```

Далее текст прошивки:

```
:020000040000FA
:1000000083161F3085008601831254200311 8F0848
:1000100003190B2810288601382086173820 0B2852
:10002000 8F0345206F3086008F01051D1A28851D1E
:1000300020281528031 18F08031927288F03272844
:10004000031 0F7300F07031C262827288F0A0F08F4
:10005000 2D2086003820452015288207 6F340C3467
:100060005B345E343C34763477344C347F347E34C5
:1000700055308C008A308D0003308E008C0B3E286A
:10008000 8D0B3E288E0B3E2808008B010301890052
:10009000 0F088800831608155530890 0AA3089009A
:1000A000881408128312080083120301890083164 2
:0A00B000081483120808 8F000800EE
:02400E00F13F80
:024200000500B7
:00000001FF
```

Выделенное жирным шрифтом в тексте программы представляет суть поставленной задачи. Специальные комментарии не требуются, т.к. основные пояснения сделаны в тексте программы. Для правильной работы устройства с алгоритмом блокировки требуется между включениями выдерживать паузу 2…5 сек для полной разрядки цепей питания.

Для самостоятельной работы предлагаю объединить в отдельной программе два свойства: сохранение значения и ограничение работоспособности. Разумеется, для этого вам потребуется два регистра ПЗУ и, как следствие, более интеллектуальный выбор адреса ПЗУ в сегментах чтения и записи. Не упускайте возможности провести гимнастику ума.

<<< назад далее >>>