Перенос программы сенсорной кнопки с ассемблера на Си. Реализация контроля нескольких сенсорных кнопок.

Данный проект является продолжением темы использование Microchip PICkit 2 Debug Express для создания емкостного сенсорного переключателя на основе технологии mTouch™

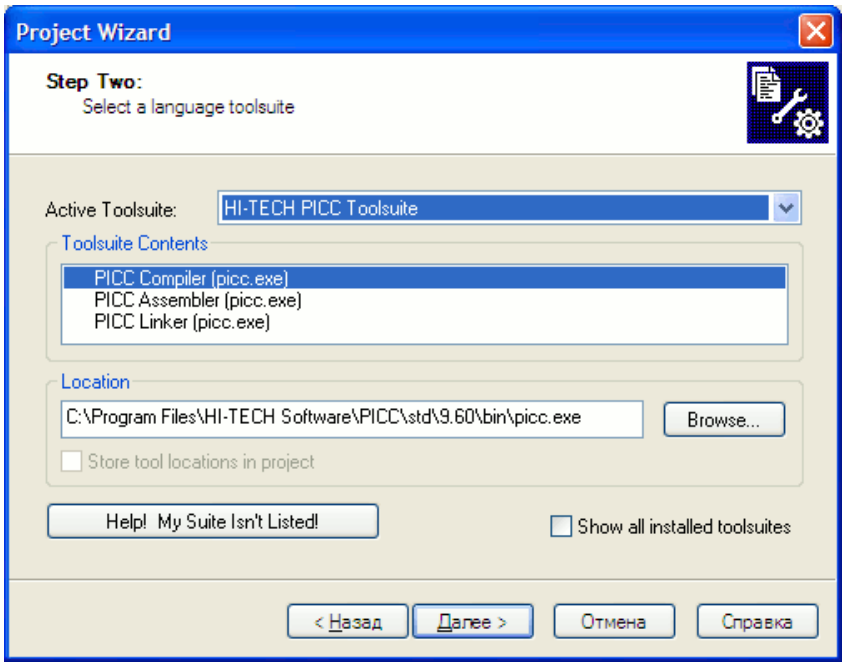
Краткий How-To по созданию проекта на Си

Необходимый инструментарий:

1. установленная среда разработки MPLAB IDE
2. установленная версия компилятора HI-TECH PICC. Можно использовать бесплатную версию PICC-Lite™ или HI-TECH C PRO for the PIC10/12/16 MCU Family (Lite mode). см. http://www.htsoft.com [http://www.htsoft.com]
3. Комплект PICkit2 Debug Express демонстрационная плата с PIC16F887

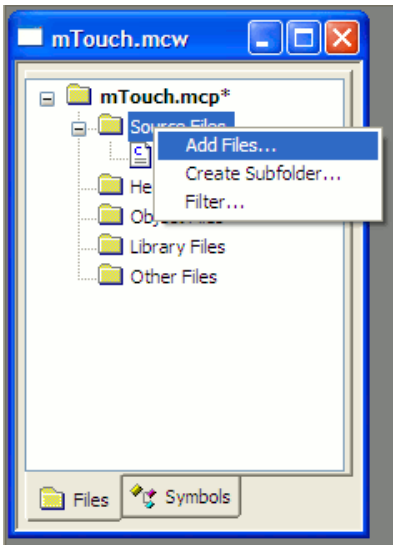
Создание проекта

Создание проекта практически ничем не отличается о создания проекта на ассемблере [http://pickit2.ru/doku.php/руководство.пользователя.pickit2#руководство.пользователя.debug.express] (см. п.4.5.3 - Создание нового проекта в MPLAB IDE), единственное что в качестве инструмента нужно выбрать не Microchip MPASM Toolsuite, а HI TECH Toolsuite



Создадим новый файл (File→New или Ctrl+N), сохраим его под именем mTouch.c

Добавим файл в проект (щелкаемся правой кнопкой Source Files, выбираем Add Files - добавить файлы)



Структура файла на языке ассемблер и Си

Примерная структура листинга файла на языке ассемблер приведена ниже

```
;************************************************
;* Директивы ассемблера настройки проекта       *
;************************************************
  processor 16f886
  ....

;************************************************
;* Подключение файла описания контроллера       *
;************************************************
include "p16F887.inc"
  ...

;************************************************
;* Определение конфигурационных битов контроллера
;************************************************
__CONFIG    _CONFIG1, _LVP_OFF & _FCMEN_OFF & _IESO_OFF & _BOR_OFF & _CPD_OFF & _CP_OFF & _MCLRE_OFF & _PWRTE_ON & _WDT_OFF & _INTRC_OSC_NOCLKOUT
__CONFIG    _CONFIG2, _WRT_OFF & _BOR21V

;************************************************
;* Объявление переменных, констант              *
;************************************************

  include "variables.inc"
  ....
  avdepth equ .6
  ....

;************************************************
;* Начало программы, Адрес сброса               *
;************************************************
  org     .0      ; Адрес сброса
reset
  call    init  ; вызов подпрограммы инициализации
  goto    main  ; переходим к основному циклу программы

;************************************************
;* обработка прерываний                         *
;************************************************
  org     .4            ; Адрес вектора прерывания
interrupt
...
...
  retfie

;************************************************
;* подпрограммы
;************************************************
...
...

;************************************************
;* основной цикл программы                      *
;************************************************
main
  ....
  ....
  end   ; Директива окончания листинга
```

В Си структура листинга остается почти такой же:

```
;***********************************************
;* Подключение файла описания контроллера
;* и других необходимых файлов
;***********************************************
#include <pic.h>

;***********************************************
;* Определение конфигурационных битов контроллера
;***********************************************
__CONFIG(LVPDIS & FCMDIS & IESODIS & BOREN & DUNPROTECT & UNPROTECT & MCLRDIS & PWRTEN & WDTDIS & INTIO & BORV21);


;***********************************************
;* Объявление переменных, констант              *
;***********************************************
...

;***********************************************
;* подпрограммы
;***********************************************
...

;***********************************************
;* подпрограммы обработки прерываний
;***********************************************
static void interrupt
isr(void) {  // функция обработчика прерывания - имя любое
if (T0IF) {
    ...
  }
}

;***********************************************
;* основной функция программы                   *
;***********************************************
void main(void) {
init();    // подпрограмма инициализации

for(;;) {  // Основной бесконечный цикл
    ...
}
}
```

В Си программе обязательно должна быть функция **main**

Перед первым упоминанием функции в листинге, эта функция должна быть определена. При этом все функции не обязательно располагать ближе к началу файла, достаточно объявить что такая функция будет определена в другом месте (ближе к концу файла или вообще в другом файле). Например

```
extern void delay (unsigned char time);
```

Итак, функция main

Как и в ассемблерной программе вначале запускаем подпрограмму инициализации init(), в которой инициализируем периферию микроконтроллера. Можно было бы не выделять в отдельную функцию, но это бы захламило листинг программы. Далее запускаем задержку на 100мс. После задержки обнуляем таймера 1 и 0, разрешаем прерывания. Далее идет бесконечный цикл программы. В данном случае программа опрашивает флаг нажатия кнопки, если кнопка нажата - зажигаем светодиод, иначе - гасим.

```
void main(void) {
    init();
    delay(100); // ; wait 100mSec

    SET_GETFIRST;

    TMR1L = 0;
    TMR1H = 0;  //      ; clear the timer value
    TMR0 = 0;           //      ; clear timer 0
    T0IE = 1;           //      ; turn on timer tick
    TMR1ON = 1; //      ; turn on timer 1
    GIE = 1;

    for(;;) {
       if(KEYDOWN) {
         PORTD = 1;
       } else {
         PORTD = 0;
       }
   }
} // end void main(void)
```

Подпрограмма инициализации

Нечего особенного, все как в ассемблерной инициализации, за исключением тогоЮ что в Си не нужно выбирать банки памяти, в которых находятся те или иные регистры. Компилятор это сделает за вас. Функция init не возвращает и не принимает никаких данных, поэтому пишем void init(void)

```
void init(void) {
  OSCCON = 0b01110000;
```

```
  OSCTUNE = 0;
  TRISA = 0b00010101;
  TRISC = 0b00000011;
  TRISD = 0;
  PORTA = 0;
  T1CON = 0b00000010;

 // инициализация mTouch
  CM1CON0 = 0b10010100;
  CM2CON0 = 0b10100000;
  CM2CON1 = 0b00110010; // компараторы поключены к CVref
  SRCON = 0b11110000;
  VRCON = 0b10000111; // включить источник опорного напряжения
  ANSEL = 0b00000101; // an0, an2 - аналоговые входы
  ANSELH = 0;
  /////////////////
  OPTION = 0b10000111; // подтяжка выключена, считаем имульсы по спадам, предделитель к t0, 1:256
  sysflag = 0; // очищаем регистр флагов (нажатия кнопки и первого измерения
}
```

Функция задержки delay

Обычно компиляторы имеют готовые библиотечные функции, в том числе и функции задержки. Но для учебного примера создадим свою функцию. Функция задержки delay получает в качестве параметра число циклов (от 0 до 255), поэтому функция определяется как void delay(unsigned char)

```
void delay (unsigned char time) {
  unsigned char temp1, temp2;

  temp1 = time;
  while(temp1--) {
     temp2 = 249;
     while(temp2--) {
        NOP();
        NOP();
        NOP();
        NOP();
     }
   }
}
```

Прерывания

В прерывании фиксируем текущее значение частоты, усредняем с предыдущими измерениями.

Если текущее значение частоты больше чем среднее значение, то это означает что емкость сенсора уменьшилась, значит кнопка не «нажата».

Если текущее значение частоты меньше чем среднее значение на величину порога, то это означает что частота резко уменьшиась из-за резкого увеличения емкости сенсора. Следовательно кнопка «нажата».

В каждом цикле прерываний измеренные значения частоты усредняются. Усреднение (плавающее значение базы, большое отклонение от которой принимаем за прикосновение) нужно для компенсации ухода частоты при изменении среды - влажности, температуры и т.п.

```
static void interrupt // Объявление что дальше будет функция обработки прерываний
isr(void) {           // имя функции обработки прерываний не важно
  if (T0IF) {
     TMR1ON = 0;      // остановить timer1

     if(GETFIRST) {   // Если контроллер только запустился - среднее значение равно первому измеренному
        LOW(average) = TMR1L;
        HIGH(average) = TMR1H;
        RESET_GETFIRST;
     }

     LOW(freq) = TMR1L;  // берем измеренное значение
     HIGH(freq) = TMR1H;
     TMR1L = 0;    // обнуляем таймер
     TMR1H = 0;

     //усреднение измерений
     // ;    (n-1) x oldvalue + newvalue
     // ; = ---------------------------------
     // ;               (n)
     avtemp = (unsigned long)average*avdepth + freq - average;
     average = avtemp/avdepth;

     if(average < freq) {  // если текущее значение больше среднего
        average = freq;    // значит кнопка не нажата
        RESET_KEYDOWN;
     } else {              // иначе
        difference = average - freq;
        if(difference > trip) { // если разница со средним больше порога
           SET_KEYDOWN;         // то кнопка нажата
        } else {                // Иначе
           RESET_KEYDOWN;       // кнопка не нажата
        }
     }
```

```
    TMR1ON = 1; //    ; запустить таймер
    T0IF = 0; //
  }
}
```

Видно, что Си программа гораздо более читабельна и текст программы занимает меньше места.

Исходные коды проекта сенсорной кнопки на Си

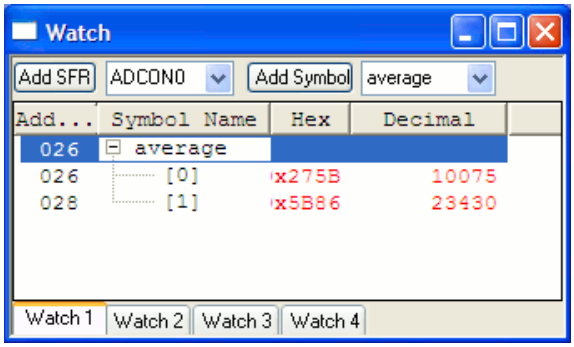
Реализация контроля нескольких сенсорных кнопок

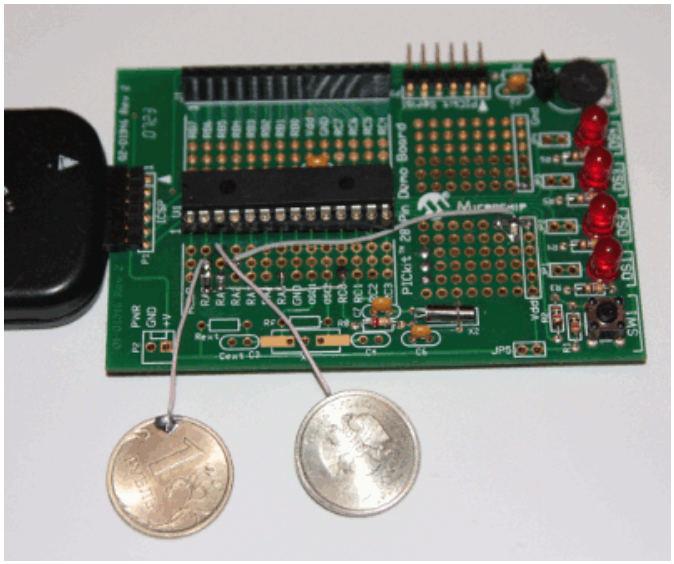
Теперь, когда программа на Си написана и отлажена, можно легко увеличть количество сенсорных кнопок. Контроллеры семейства PIC16F88x (PIC!6F886, PIC16F887 и другие) имеют на входе компараторов мультиплексор на 4 внешних входа, поэтому легко добавить обработку дополнительных сенсорных кнопок (до 4-х), причем потребуется минимум внешних элементов (по одному резистору на каждую кнопку).

Обратите внимание
Соседние кнопки будут оказывать влияние друг на друга из-за наличия паразитных емкостей между проводниками на печатной плате. Поэтому возможно понадобится корректировка порога срабатывания. Однако для упрощения себе жизни внимательно отнеситесь к разработке печатной платы и минимизации влияния сенсорных каналов друг на друга. Подробнее см. на http://microchip.com/mTouch [http://microchip.com/mTouch]

Для корректировки порога чувствительности кнопок и предотвращения взаимного влияния пригодится PICkit2 в режиме отладчика. С помощью PICkit2 можно посмотреть состояние переменных в работающем микроконтроллере.

Отметьте значения среднего значения частоты в каналах для «нажатых» и «отпущенных» кнопок. Отметьте взаимное влияние кнопок друг на друга, исходя из этого задайте порог срабатывания кнопок.





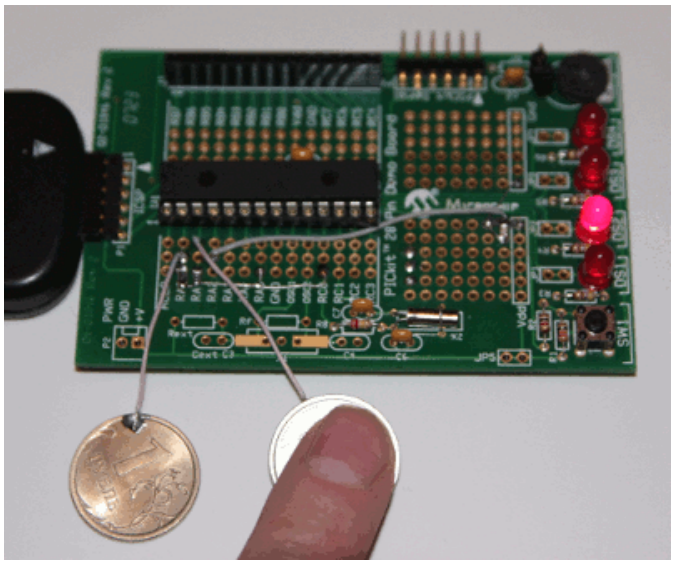
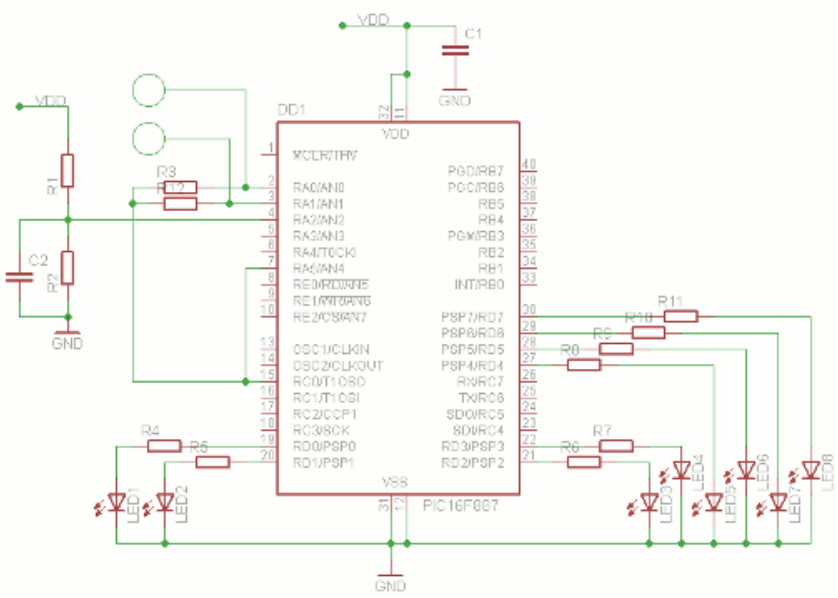


Схема с 2-я сенсорными кнопками



Исходные коды программы под 2 сенсорные кнопки.

Уменьшение потребления схемы и другие модификации программы и схемы

В рассмотренном примере мы в качестве верхнего порога релаксационного генератора использовали программируемый источник опорного напряжения, а нижний порог задан внешним делителем. Многие микроконтроллеры PIC имеют несколько источников опорного напряжения - программируемый источник опорного напряжения и фиксированный источник опорного напряжения. В PIC16F886 (PIC16F887 и др.) есть фиксированный источник опорного напряжения на 0.6В, который можно сконфигурировать на подключение к компараторам. Поэтому мы можем легко исключить из схемы внешний делитель напряжения и использовать внутренний источник опорного напряжения.

Исходные коды программы под 2 сенсорные кнопки и внутренний источник напряжения 0.6В.