Обработка логических сигналов на входах микроконтроллера.

В современных проектах с применением микроконтроллеров в управлении объектами приходиться сталкиваться с обработкой логических сигналов, которые могут содержать большое количество помех. Эти помехи могут быть вызваны как дребезгом контактов механических устройств, так и электромагнитным воздействием, которое может, присутствуют в окружающей среде. Особенно это актуально для устройств, предназначенных для работы на промышленных объектах с высоким уровнем электромагнитным помех.

Классически для решения подобных задач применяют принцип опроса порта с последующим ожиданием и повторным опросом. Но такое решение вносит в основной рабочий цикл программы значительные задержки, что часто не желательно для большинства проектов.

Для решения этой проблемы был разработан алгоритм обработки входных сигналов, который позволяет отфильтровывать сигналы с длительностью ниже заданного значения и при этом вносить минимальную задержку в основной цикл программы. Такой алгоритм может с успехом применяется не только для контроля сигналов удаленных датчиков, но и для опроса кнопок управления устройством.

Этот алгоритм представляет собой компаратор длительности сигнала, который позволяет изменять значение флага состояния порта, только при получении длительности сигнала превышает установленного значения.

На рисунке 1 приведен пример работы такого алгоритма. Как только длительность сигнала превышает заданное значение, флаг порта переключается.

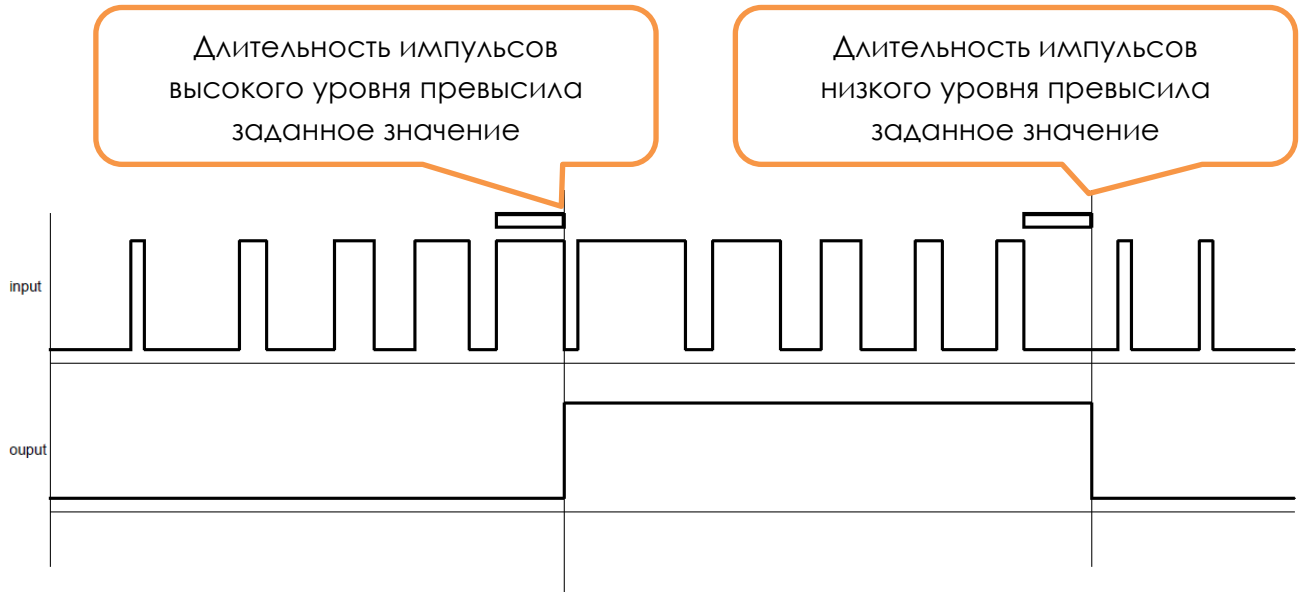


Рисунок 1

Из рисунка видно, что флаг порта переключают сигналы, длительность которых превышает заданное значение. Можно (при необходимости) отдельно задавать длительность переключения в состояние «1» и длительность переключения в состояние «0»

Для реализации этого алгоритма необходимо выделить буфер порта. Этот буфер необходим, что бы дать возможность другим функциям программы независимо использовать

порт, например, для вывода данных на светодиодный индикатор. В этот буфер с определённым периодом копируются входные данные. Для каждой линии ввода выделяется бит предыдущего состояния и переменная таймера задержки.

Логика работы функции следующая, она проверяет состояние бита буфера и в зависимости от его текущего состояния сравнивает с состоянием, полученным при предыдущей проверке. Если они не совпадают, то она инициализирует (загружает) таймер задержки и устанавливает бит прошлого состояния равный текущему состоянию.

Если текущее состояния, анализируемого бита порта, совпадает с прошлым состоянием, то функция тестирует таймер задержки на ноль и если он равен нулю, устанавливает флаг.
Аналогичный алгоритм и для сброса флага.

В ниже приведенной программе тестируется порт B. Подразумевается, что включены подтягивающие резисторы.

```
if(rPORTB & 0b10000000)        // тестируем необходимый бит порта
{// если на входе ноль
   if(bps==1)                  // проверяем флаг предыдущего состояния
   { // они не равны
         bps=0;                // обнуляем флаг
         timerD=ZADINPU;       // инициализируем таймер задержки
   }
   if(timerD ==0)              //проверяем таймер задержки == 0
   {// если – Да
         Flag=1;               // устанавливаем Флаг
   }
}
else
{// если на входе единица
   if(bps ==0)                 // проверяем бит предыдущего состояния
   {// они не равны
         bps =1;               // устанавливаем флаг
         timerD =ZADINPU;      // инициализируем таймер задержки
   }
   if(timerD ==0)              //проверяем таймер задержки == 0
   {// если – Да
         Flag =0;              // сбрасываем Флаг
   }
}
```

Аналогично можно тестировать кнопки клавиатуры управления устройством, только для этого нет необходимости тестирования на сброс флага.

```
if(rPORTB & 0b10000000)     // тестируем необходимый бит порта
{     // если на входе ноль
      if(bps==1)            // проверяем флаг предыдущего состояния
      { // они не равны
            bps=0;          // обнуляем флаг
            timerD=ZADINPU; // инициализируем таймер задержки
      }
      if(timerD ==0)        //проверяем таймер задержки == 0
      {     // если – Да
            Flag=1;         // устанавливаем Флаг
```

```
            }
        }
        else
        {       // если на входе единица
            bps =1;             // устанавливаем бит
            Flag =0;            // сбрасываем Флаг
        }
```

Таймер обработки задержки необходимо поместить в прерывания, где будет формироваться необходимая длительность. Сам отсчет времени организуется следующим образом.

```
//------------------------период прерываний 10 Гц.------------------------
        if(timerD!=0) timerD--;
```

Для чтения порта в прерываниях устанавливаем блок для чтения входных сигналов:

```
//-------------------------------------------------
   //чтение состояния кнопок клавиатуры и входов датчиков
   TRISB=0xFF;                  // отключение выходных сигналов настройка на прием данных
   _delay(1);                   // задержка для нормализации уровней сигналов
                                // для тактовой частоты выше 8 мГц, необходимо увеличь
   rPORTB=PORTB;                // копирования данных порта в буфер
   TRISB=0;                     // настройка порта на вывод данных
//-------------------------------------------------
```

Логика работы копирования данных в буфер порт может быть иной и зависит от конкретного расположения портов ввода – вывода.

Демонстрационная программа, использующая настоящий алгоритм.

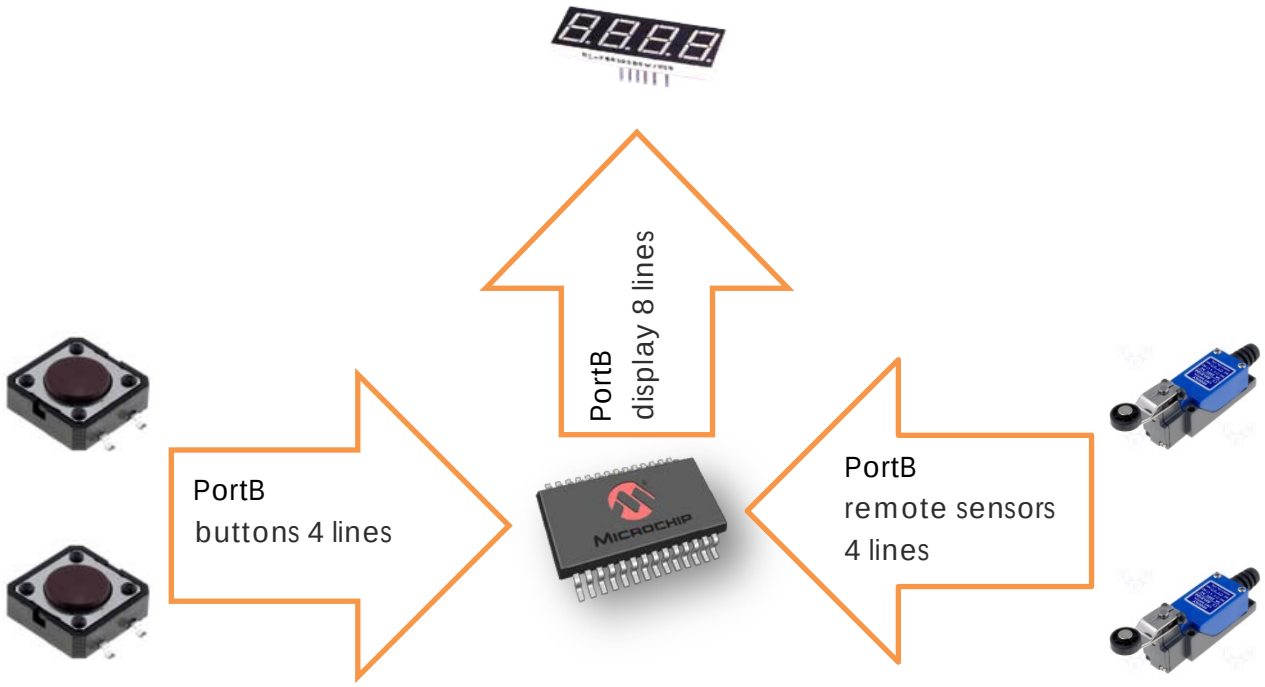


Рисунок 2

Для демонстрации используется схема реального изделия. В этой схеме PORTB контроллера используется для динамической индикации на светодиодный дисплей и этот же порт используется для получения информации с четырех кнопок клавиатуры и с четырех удаленных датчиков. Для обработки дребезга контактов тактовых кнопок используется задержка равная 0,1 Сек. А для фильтрования помех с удаленных датчиков длительность в 1 секунду, как на установку флага, так и на сброс.

Тактовая частота контроллера 8 мГц. Опрос PORT B помещен в прерывания.

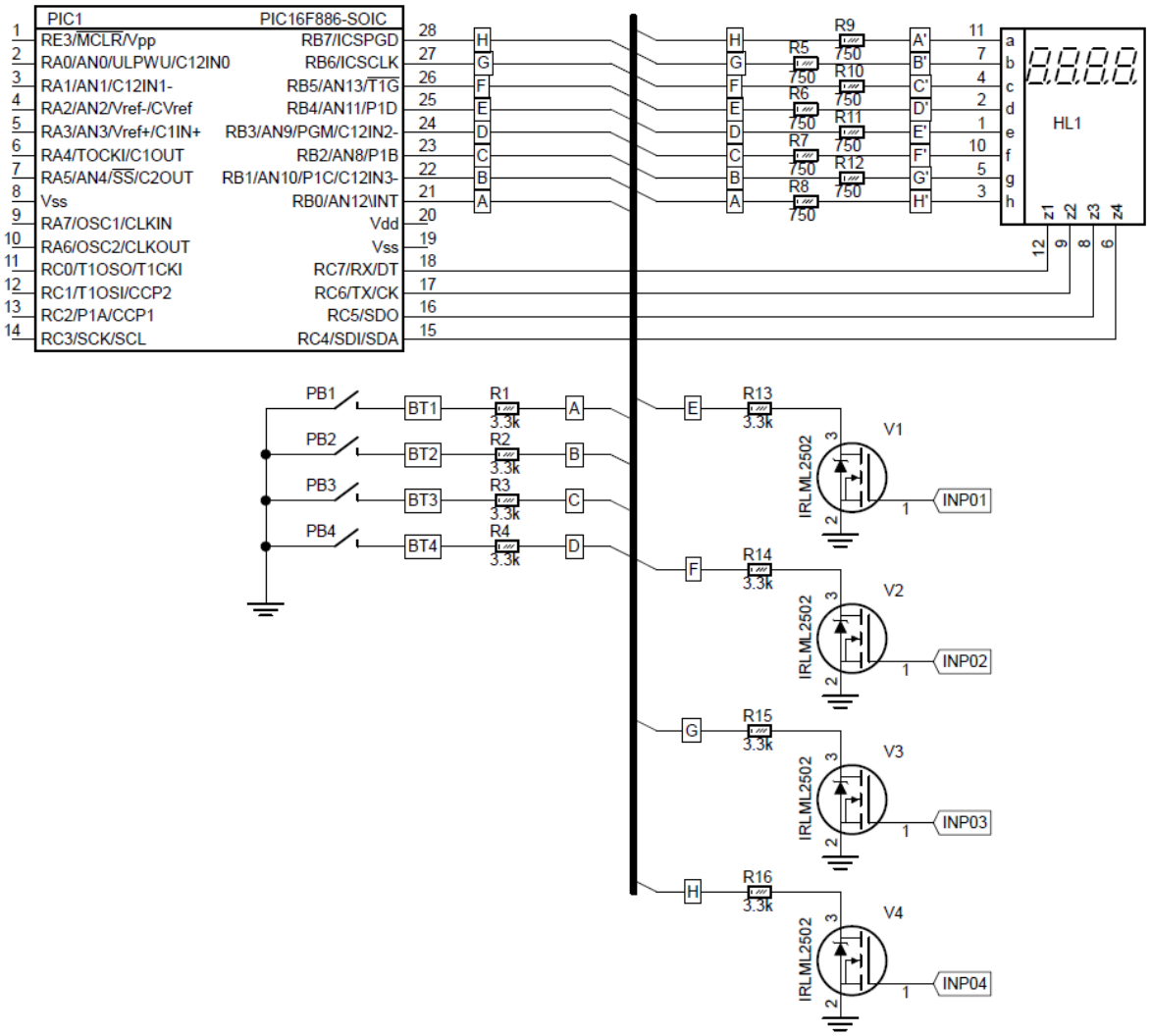
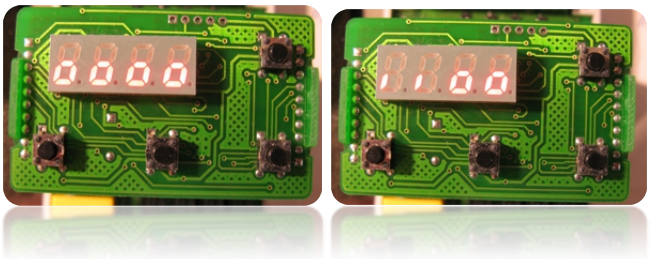


Рисунок 3

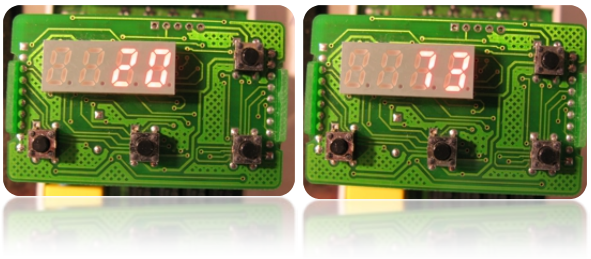
В демонстрационной программе для индикации нажатия кнопки используется принцип подсчета импульсов. В ней реализован алгоритм отдиночного нажатия и последующего автонабора с увеличением скорости при удержании кнопки.

Схема для тестирования приведена на рисунке 3. Значение резисторов приведенно для питания 5 вольт. Для питания 3,3 вольта необходимо резисторы в цепи сетодиодного индикатора уменьшить до 220 ом, а резисторы в цепи кнопок и полеыфх транзисторов увеличить до 6,8-10 кОм.

Для демонстрации сигналов с удаленных датчиков используется индикация «ноликов» в нижних сегментах индикатора.



Для демонстрации подсчета импульсов цифровое значение счетчика.



Смотрите видео …

Демо программа …

The version v2.0.
Email: gena.chernov@gmail.com
http://svetomuzyka.narod.ru/ http://invent-systems.narod.ru/