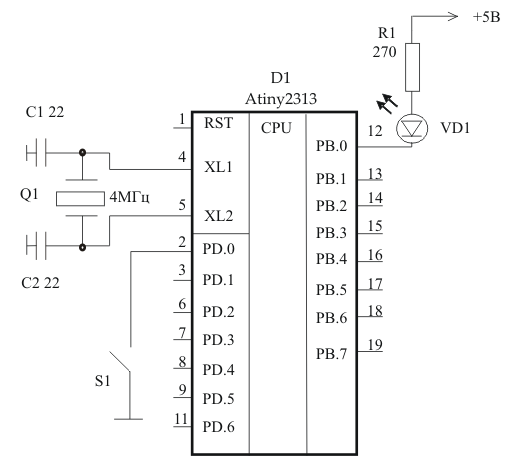
Цель работы

Изучить на примере светодиодного индикатора, работающего в режиме поочередного включения и отключения, программные методы борьбы с дребезгом контактов; организацию процедур ожидания события; команды вызова и возврата из подпрограммы rcall и ret, работы со стеком push и pop, декремента dec, перехода по условию «неравно» brne.

****

.include "tn2313def.inc" ; Присоединение файла описаний

.list ; Включение листинга

.def temp = r16 ; Определение главного рабочего регистра

.def loop = r17 ; Определение регистра организации цикла

;------------------------- Начало программного кода

.cseg ; Выбор сегмента программного кода

.org 0 ; Установка текущего адреса на ноль

;-------------------------- Инициализация стека

ldi temp, RAMEND ; Выбор адреса вершины стека

out SPL, temp ; Запись его в регистр стека

;-------------------------- Инициализация портов ВВ

ldi temp, 0 ; Записываем 0 в регистр temp

out DDRD, temp ;Записываем этот 0 в DDRD (порт PD на ввод)

ldi temp, 0xFF ;Записываем число $FF в регистр temp

out DDRB, temp ; Записываем temp в DDRB (порт PB на вывод)

out PORTB, temp ; Записываем temp в PORTB (потушить светодиод)

out PORTD, temp ; Записываем temp в PORTD (включ. внутр. резист.)

;--------------------------- Инициализация компаратора

ldi temp, 0x80 ; Выключение компаратора

out ACSR, temp

;----------------------------Начало основного цикла

;---------------------------- Цикл ожидания замыкания кнопки

main:in temp, PIND ; Читаем содержимое порта PD

sbrc temp, 0 ; Проверка младшего разряда

rjmp main ; Если не ноль, переходим в начало

rcall wait ;Вызов подпрограммы задержки

;-----------------------------Переключение светодиода

in temp, PINB ;Читаем содержимое порта PB

sbrc temp, 0 ;Проверка младшего разряда

rjmp m1

sbi PORTB, 0 ;Установка выхода PB.0 в единицу

rjmp m2

m1: cbi PORTB, 0 ;Сброс PB.0 в ноль

;-----------------------------Цикл ожидания отпускания кнопки

m2: in temp, PIND ;Читаем содержимое порта PD

sbrs temp, 0 ;Проверка младшего разряда

rjmp m2 ;Продолжить ожидание отпускания кнопки

rcall wait ;Вызов подпрограммы задержки

rjmp main ;К началу цикла

;-----------------------------Подпрограмма задержки

wait: push loop ;Сохраняем содержимое регистра loop

ldi loop, 200 ;Помещаем в loop константу задержки

;Цикл задержки:

wt1:

dec loop ;Уменьшаем значение регистра loop

brne wt1 ;Если не ноль, продолжаем цикл

pop loop ;Восстанавливаем значение регистра loop

ret ;Выход из подпрограммы

Описание программы

Антидребезговая задержка. Так как задержка нужна в двух разных местах программы, она оформлена в виде подпрограммы. Для формирования задержки используется один дополнительный регистр общего назначения.

Поэтому в начале нашей новой программы (строка 4) добавлена команда описания регистра. Регистру г17 присваивается имя loop. Регистр loop можно считать переменной с именем loop.

Запись значения в этот регистр эквивалентна присвоению значения переменной. Также можно интерпретировать и другие операции с регистром. Сложение содержимого двух регистров можно считать сложением переменных, вычитание – вычитанием переменных, и так далее.

Подпрограмма задержки расположена в строках 32—37. Первой строке подпрограммы присвоена метка wait. Именно по этой метке и будет вызываться подпрограмма. Опустим пока назначение команд push и pop (строки 32 и 36). Собственно процедура задержки расположена в строках 33—35. Формирование задержки производится путем многократного выполнения пустого цикла. Сначала в регистр loop записывается некоторое число (строка 33).

Затем начинается цикл, который постепенно уменьшает значение регистра loop до нуля (строки 34 и 35). Происходит это следующим образом. В строке 34 содержимое регистра уменьшается на единицу, а в строке 35 происходит проверка содержимого на нуль. Если содержимое регистра не равно нулю, то управление передается по метке wt l, и цикл повторяется. Когда же содержимое loop окажется равным нулю, очередного перехода не произойдет, и цикл задержки закончится.

Время задержки при этом определяется числом, загруженным в рабочий регистр, и временем выполнения команд, образующих программный цикл.

Предположим, что в управляющей программе необходимо реализовать временную задержку 153 мкс. Для ее получения требуется определить число Х, загружаемое в рабочий регистр. Определение числа Х выполняется на основе расчета времени выполнения команд, образующих данную подпрограмму (см. листинг 3.1). При этом необходимо учитывать, что команды push и pop, ldi и ret выполняются однократно, а число повторений команд dec и brne равно числу Х. Кроме того, обращение к подпрограмме временной задержки осуществляется по команде rcall wait, время исполнения которой также необходимо учитывать при подсчете временной задержки. В сводной таблице команд Ассемблера микроконтроллеров AVR указывается, за сколько тактов исполняется каждая команда: rcall – 3 такта, push – 2, ldi – 1, dec – 1, brne – 2 (команда brne выполняется за один такт, если не вызывает перехода, и за два такта, если вызывает переход), pop – 2, ret – 4 такта.

Тактовая частота кварцевого резонатора у нас равна 4 МГц. Длительность одного такта равна 0,25 мкс. Таким образом, подпрограмма выполняется за время (3 + 2 + 1 + (1+2)Х +2 +4) 0,25 мкс = (12 +3Х) 0,25 мкс= (3+0,75Х) мкс. . Из уравнения 3+0,75Х = 153 получим Х = 200

В данном случае при загрузке в регистр числа 200 требуемая временная задержка (153 мкс) реализуется точно. Если число Х получается дробным, то временную задержку можно реализовать лишь приблизительно. Для более точной подстройки в подпрограмму могут быть включены команды NOP (нет операции), время выполнения которой равно одному такту (в данном случае – 0,25 мкс).

Минимальная временная задержка, реализуемая подпрограммой, составляет 3,75 мкс (Х = 1). Временную задержку меньшей длительности программным путем можно реализовать, включая в программу цепочки команд NOP.

Максимальная задержка, которую можно сформировать при помощи 8-разрядного регистра и данной подпрограммы, равна (3+0,75\*255) = 194,25 мкс.

Для реализации задержки большей длительности можно рекомендовать увеличить тело цикла включением дополнительных команд или использовать метод вложенных циклов. Так, например, если в подпрограмму задержки перед командой dec вставить дополнительно три команды NOP, то максимальная задержка составит (3+1,5\*255) = 385,5 мкс (т.е. почти в 2 раза больше).

Задержку большей длительности можно получить также за счет предварительного деления частоты тактового генератора. Тогда длительность выполнения каждой команды нужно было бы умножать на коэффициент деления предварительного делителя.

Рассмотрим команды push и pop работы со стеком. Они предназначены для сохранения в стеке (push) и последующего восстановления (pop) содержимого регистра 1оор. В начале подпрограммы (строка 32) значение 1оор сохраняется, а перед выходом из подпрограммы (строка 36) – восстанавливается.

Подобный прием придает программе одно полезное свойство. После окончания работы подпрограммы значения всех регистров микроконтроллера остаются без изменений. В данном конкретном случае такое свойство ничего не дает, кроме дополнительной задержки. Однако в сложных программах, имеющих не одну, а несколько подпрограмм, одни и те же регистры удобно использовать в разных подпрограммах.

Те же самые регистры может использовать и основная программа. В этом случае описанное выше полезное свойство просто необходимо для правильной работы всей программы. Зная эту особенность, программисты стараются применять подобный прием в каждой подпрограмме, независимо от того, полезен он в данном конкретном случае или нет.

В соответствии с алгоритмом (рис.3.1) подпрограмма задержки в нашей программе вызывается два раза. Первый раз — после окончания цикла ожидания нажатия кнопки (строка 20). Второй раз — после окончания цикла ожидания отпускания кнопки (строка 30).

Вывод

В процессе выполнения лабораторной работы были изучены программные методы борьбы с дребезгом контактов; организацию процедур ожидания события на примере светодиодного индикатора, работающего в режиме поочередного включения и отключения; команды вызова и возврата из подпрограммы rcall и ret, работы со стеком push и pop, декремента dec, перехода по условию «неравно» brne.