БГУИР

Кафедра ЭВМ

Отчет по лабораторной работе №1

Тема: «Знакомство с CCS. Цифровой ввод-вывод»

Выполнил:

студент группы 350431 Козяков А.И.

Проверил:

к.т.н., доцент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Селезнёв И.Л.

Минск

2010

**Цель:**

Ознакомиться с интегрированной средой разработки Code Composer Studio. Ознакомиться с основными функциональными возможностями платы MSP-EXP430F5529. Изучить приемы работы с цифровыми выводами.

**Постановка задачи:**

Разработать программу для микроконтроллера макетной платы MSP-EXP430F5529. Алгоритм работы программы:

1. По отпусканию кнопки S2 переключать своё состояние между «выключено» и «включено».
   1. При переходе в состояние «выключено» программа должна перестать реагировать на кнопку S1, а светодиоды должны погаснуть.
   2. При переходе в состояние «включено» разрешается работа кнопки S1, а светодиоды должны зажечься и гореть непрерывно.
2. По отпусканию кнопки S1 должен переключаться режим работы светодиодов между «мерцанием» и «непрерывным горением».

При этом не допускается подвисание устройства при нажатии и удерживании кнопок S1 и S2, а так же не должны использоваться прерывания и аппаратные таймеры.

**Теоретические сведения:**

### Экспериментальная плата MSP-EXP430F5529 разработана на основе микроконтроллера MSP430F5529 компании Texas Instruments. Это серия процессоров для обработки смешанных сигналов со сверхнизким энергопотреблением.

8-разрядные порты P1, P2, P3,…,P8, PJ управляют выводами контроллера. Выводы программируются либо как I/O, либо как вход/выход периферии. Порты могут объединяться в пары: P1 и P2 = PA, P3 и P4 = PB, P5 и P6 = PC, P7 и P8 = PD. При работе с прерываниями порты в пары не объединяются. Для порта могут быть доступны регистры:

* PxIN – чтение данных с вывода;
* PxOUT – установка значения выхода;
* PxDIR – выбор направления: 0 – вход, 1 – выход;
* PxREN – разрешение подтягивающего резистора;
* PxDS – выбор допустимой силы вывода;
* PxSEL – выбор функции вывода: 0 – I/O, 1 – периферия;
* PxIV – генерирует значение для изменения счетчика команд, соответствующее прерыванию с максимальным приоритетом;
* PxIES – выбор направления перепада для генерации запроса на прерывание: 0 – по фронту, 1 – по спаду;
* PxIE – разрешение прерывания;
* PxIFG – флаг прерывания.



Рис. 1. Подключение пользовательских кнопок и светодиодов

Пользователю программно доступны две кнопки S1 и S2, подключенные соответственно к выводу 7 порта 1 и выводу 2 порта 2. В дальнейшем такое подключение будем обозначать как P1.7 и P2.2 соответственно. Также программно доступны 8 светодиодов, три из которых (LED1 – LED3) размещены рядом с кнопками и подключены соответственно к выводам P1.0, P8.1, P8.2. Еще 5 светодиодов (LED4 – LED8) размещаются в блоке сенсорных кнопок и подключены к выводам P1.1 – P1.5 соответственно. Логика управления выводом на примере порта 1 представлена на рисунке ниже. Для других портов схемотехника может несколько отличаться, в зависимости от особенностей подключаемой к выводу периферии микроконтроллера.

### 

### Рис. 2. Организация входа-выхода с триггером Шмидта на примере порта 1

### Алгоритм решения задачи

Разработанная программа работает по следующему алгоритму:

1. Инициализация
2. В бесконечном цикле:
   1. Обновить состояние кнопок
   2. Если произошло событие отпускания кнопки, то обработать его.
   3. Обновить состояние светодиодов.

Обновление состояния кнопок производится следующим образом:

1. Считать текущее значение порта с кнопкой
2. Если текущее состояние порта – высокий уровень, а предыдущее состояние порта – низкий уровень, то сохранить факт отпускания кнопки.
3. Перезаписать предыдущее состояние текущим.

Обработка событий кнопок:

1. Если была отпущена кнопка S1, то
   1. Если работа устройства разрешена, то
      1. Погасить светодиоды
      2. Запретить работу устройства
   2. Иначе
      1. Разрешить работу устройства
      2. Включить светодиоды.
2. Иначе если была отпущена кнопка S2 и работа устройства разрешена, то
   1. Если мигание включено, то
      1. Выключить мигание
      2. Зажечь светодиоды
   2. Иначе включить мигание

Обновление состояния светодиодов:

1. Если работа устройства разрешена, и мигание включено, то
   1. Увеличить на 1 счётчик задержки мигания светодиодов
   2. Если счётчик досчитал до заданного значения, то
      1. Переключить состояние светодиодов на противоположное
      2. Обнулить счётчик задержки мигания.

### Листинг программы

#include <msp430.h>

/\*

Программа для микроконтроллера макетной платы MSP-EXP430F5529

Алгоритм работы программы:

- по отпусканию кнопки S2 переключать своё состояние между «выключено» и «включено».

При переходе в состояние «выключено» программа должна перестать реагировать на кнопку S1, а светодиоды должны погаснуть.

При переходе в состояние «включено» разрешается работа кнопки S1, а светодиоды должны зажечься и гореть непрерывно.

- по отпусканию кнопки S1 должен переключаться режим работы светодиодов между «мерцанием» и «непрерывным горением»

Условия:

- не допускается подвисание устройства при нажатии и удерживании кнопок

- не должны использоваться прерывания и аппаратные таймеры

\*/

/\*

============= Инициализация перменных ========

\*/

static int buttonS2; // состоние кнопки S1

static int buttonS1; // состоние кнопки S2

static int isS2PushedNow = 0; // кнопка S2 нажата/не нажата

static int isS1PushedNow = 0; // кнопка S1 нажата/не нажата

static int isDeviceON = 0; // устройство включено/выключено

static int isBlinkingEnabled = 0; // мерцание разрешено/запрещено

static int delay; // значние задержки

static int isDiodsON = 0; // диоды включены/выключены

int IsCanBlinking(int, int, int);

/\*

=========================================

\*/

// Инициализация состояний диодов

static void InitLeds()

{

// установка направления портов на выход

P8DIR |= BIT1;

P1DIR |= BIT0;

P8DIR |= BIT2;

}

// Инициализация состояний кнопок S1 и S2

static void InitButtons()

{

// устанавливка направления на вход

P1DIR &= !(BIT7); // кнопка S1

P2DIR &= !(BIT2); // кнопка S2

//разрешение подтягивающего регистра

P1REN |= BIT7; // кнопка S1

P2REN |= BIT2; // кнопка S2

}

// Включение диодов

static void TurnLedsOn()

{

P1OUT |= BIT0;

P8OUT |= BIT1;

P8OUT |= BIT2;

}

// Выключение диодов

static void TurnLedsOff()

{

P1OUT &= ~(BIT0);

P8OUT &= ~(BIT1);

P8OUT &= ~(BIT2);

}

/\* Проверка на возможность мерцания диодов

Вход:

int s1State - состояние кнопки S1(1 - нажата/0 - не нажата)

int s2State - состояние кнопки S2(1 - нажата/0 - не нажата)

int isEnabled - состояние диодов(1 - включено/0 - выключено)

Выход:

1 - (мерцание разрещено) или 0 - (мерцание запрешено)

\*/

int IsCanBlinking(int s1State, int s2State, int isEnabled)

{

// если мерцание разрешено и устройство включено и диоды вкючены,

// то возвращаем 1(мерцания разрешено), если нет - 0(мерцание запрещено)

return (s1State > 0 && s2State > 0 && isEnabled == 1) ? 1 : 0;

}

int main()

{

WDTCTL = WDTPW | WDTHOLD; // Stop watchdog timer

// Инициализация состояния диодов

InitLeds();

// Инициализация состояния кнопок

InitButtons();

while (1)

{

// Читаем состояния кнопок

buttonS1 = (P1IN & BIT7);

buttonS2 = (P2IN & BIT2);

// Если можно мерцать диодами

if (IsCanBlinking(isBlinkingEnabled, isDeviceON, isDiodsON))

{

// Переключаем диоды в противоположное состояние

P8OUT ^= BIT1;

P1OUT ^= BIT0;

P8OUT ^= BIT2;

// Фиксируем, что диоды включены

isDiodsON = 0;

}

// Если мерцание разрешено и устройство включено

else if (isBlinkingEnabled > 0 && isDeviceON > 0)

{

TurnLedsOn(); // Включаем диоды

isDiodsON = 1; // Фиксируем, что диоды включены

}

// Если мерцание запрещено и устройство включено

else if (isBlinkingEnabled == 0 && isDeviceON > 0)

{

TurnLedsOn(); // Включаем диоды на случай, если они были выключены при мерцании

isDiodsON = 1; // Фиксируем, что диоды включены

}

else // Иначе - устройство выключено

{

TurnLedsOff(); // Выключем диоды

isDiodsON = 0; // Фиксируем, что диоды выключены

}

// Обработка состояния кнопки S2(кнопка включения устройства)

// Если было зафексировано нажатие кнопки и в данный момент она не нажата

if (isS2PushedNow == 1 && buttonS2 != 0)

{

// Обнуляем фиксацию нажатия кнопки

isS2PushedNow = 0;

// Если устройство было выключено

if (isDeviceON == 0)

{

isDeviceON = 1; // Фиксируем, что устройство включено

TurnLedsOn(); // Включаем диоды

isDiodsON = 1; // Фиксируем, что диоды включены

}

// Если устройство было включено

else if (isDeviceON > 0)

{

isDeviceON = 0; // Фиксируем, что устройство выключено

isBlinkingEnabled = 0; // Фиксируем, мерцание запрещено

TurnLedsOff(); // Выключаем диоды

}

}

// Иначе, если кнопка нажата в данный момент

else if (buttonS2 == 0)

{

// Фиксируем нажатие кнопки включения/выключения устройства

isS2PushedNow = 1;

}

// Обработка состояния кнопки S1(кнопка мерцания диодами)

// Если было зафексировано нажатие кнопки и в данный момент она не нажата

if (isS1PushedNow == 1 && buttonS1 != 0)

{

// Обнуляем фиксацию нажатия кнопки

isS1PushedNow = 0;

// Если мерцание было выключено

if (isBlinkingEnabled == 0)

{

isBlinkingEnabled = 1; // Фиксируем, что мерцание включено

}

// Если мерцание было включено

else if (isBlinkingEnabled > 0)

{

isBlinkingEnabled = 0; // Фиксируем, что мерцание выключено

}

}

// Иначе, если кнопка нажата в данный момент и устройство включено

else if (buttonS1 == 0 && isDeviceON > 0)

{

// Фиксируем нажатие кнопки включения/выключения мерцания

isS1PushedNow = 1;

}

// устанавливаем задержку до следующей итерации цикла

delay = 8000;

while (delay > 0)

{

delay--;

}

}

}

**Заключение**

В результате выполнения работы была написана требуемая программа и изучена работа с цифровыми портами ввода-вывода микроконтроллера MSP430F5529.