Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Ульяновский государственный технический университет»

Кафедра «Вычислительная техника»

Дисциплина «Автоматизация проектирования микропроцессорных систем»

Лабораторная работа №3

Вариант № 17

Выполнил студент

группы ИВТАПбд-41:

Сулейманов М.З.

Проверил:

Игонин А.Г.

Ульяновск, 2024

**Оглавление**

[1. Задание 3](#_Toc186075891)

[2. Краткие теоретические сведения 3](#_Toc186075892)

[3. Основные сведения об устройстве 4](#_Toc186075893)

[4. Сведения о среде моделирования 4](#_Toc186075894)

[5. Порядок выполнения работы 5](#_Toc186075895)

[6. Вывод 9](#_Toc186075896)

[7. Список литературы 9](#_Toc186075897)

# Задание

Выполнить лабораторную работу № 2 с применением микроконтроллера. По заданию, микроконтроллер AVR.

На вход схемы поступают 8 сигналов x1...x8

В выход схемы поступают 4 сигнала y1..y4

В системе имеется 4 таймера

T1 = 3c, T2 = 5с, T3 = 7c, T4 = 11c

y1 =( x1&x2'&x3 | x4&x5'&x6)&T2

y2 = x3&T1 | x5'&x6T2 | x3'&x7T3 | x4'&x8T4

y3 = (x1|x3'|x5)T4

y4 = x1&x2&x3&x4&x5&x6&x7&x8 | x1'&x2'&x3'&x4'&x5'&x6'&x7'&x8'

# Краткие теоретические сведения

Микропроцессорные системы — частный вид электронный системы, которые обладают узлом, блоком, прибором ил комплексом, которые производят обработку информации.

Сигнал — это изменение во времени некоторой физической величины

Таймер микроконтроллера – это цифровой счётчик, осуществляющий подсчёт количества подаваемых на него импульсов. Источником импульсов для таймера-счётчика могут служить: как тактовые импульсы от внутреннего генератора МК, так и импульсы, подаваемые непосредственно на вход таймера с внешнего источника.

Микроконтроллер — микросхема для программного управления электронными устройствами.

# Основные сведения об устройстве

Микроконтроллер ATmega16 - это 8-ми разрядный микроконтроллер с 16 Кб внутрисистемно программируемой Flesh-памяти. Схема и расположение выводов представлены на рис.1

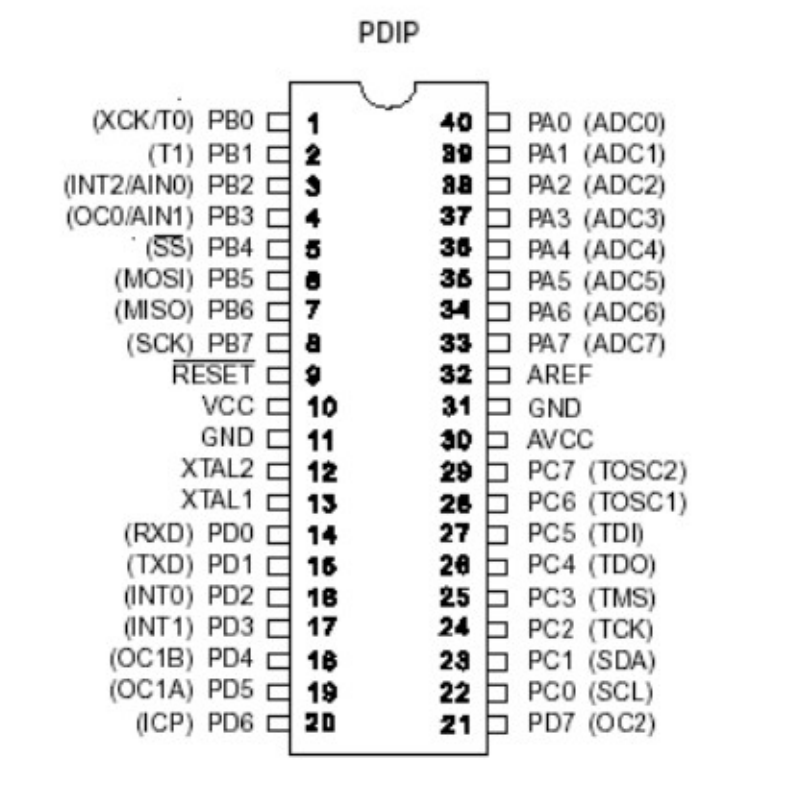


Рис. 1. Схема ATmega16.

# Сведения о среде моделирования

Proteus – это пакет программ для автоматизированного проектирования (САПР) электронных схем. Представляет возможности ввода схемы в графическом редакторе, моделирования ее работы и разработки печатной платы. Возможность моделирования работы микроконтроллеров (PIC, AVR и т. д.)

Microchip Studio - (ранее Atmel Studio и AVR Studio) — основанная на Visual Studio бесплатная проприетарная интегрированная среда разработки (IDE) для разработки приложений для 8- и 32-битных микроконтроллеров семейства AVR и 32-битных микроконтроллеров семейства ARM от компании Atmel, работающая в операционных системах Windows. Atmel Studio содержит компилятор GNU C/C++ и эмулятор, позволяющий отладить выполнение программы без загрузки в микроконтроллер.

# Порядок выполнения работы

Для выполнения данной работы необходимо создать проект, в котором будет размещаться схема. После создания, необходимо добавить в список те элементы схемы, которые будут отображаться на схеме. В меню, во вкладке «Component Mode» необходимо найти следующие элементы:

– LOGICSTATE – элемент для установки логического состояния 1 или 0

– LED-BLUE – светодиод для отображения результатов функций

– ATMEGA16 — микроконтроллер

– RES – резистор, который соединяется с выводом микроконтроллера и светодиодом. Для резисторов надо поставить сопротивление 220 Ом.

Во вкладке «Terminals Mode» надо добавить элемент GROUND — для заземления светодиода.

По умолчанию питание микроконтроллера не отображается в Proteus, поэтому добавлять лишнее не имеет необходимости.

После добавления всех элементов на схему, необходимо их соединить проводами. (Рис.2.)

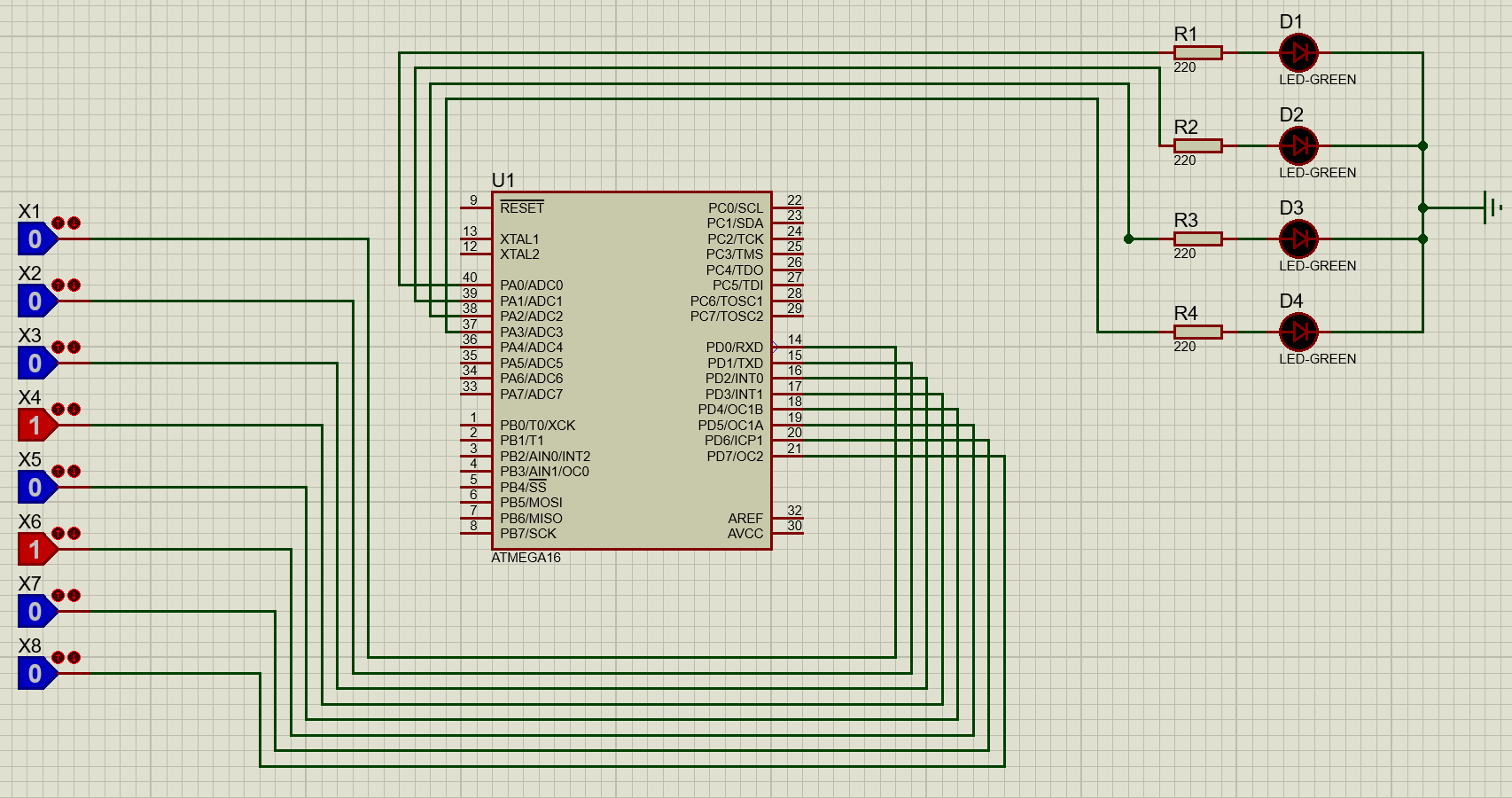


Рис. 2. Схема в Proteus

Для того, чтобы запрограммировать микроконтроллер, в Microchip Studio необходимо создать проект и выбрать язык Си/C++. В данном проекте необходимо определить частоту микроконтроллера и библиотеки, необходимые для работы с ним. (Листинг 1)

*Листинг 1. Библиотеки для работы*

|  |
| --- |
| define F\_CPU 1000000UL  #include <avr/io.h>  #include <util/delay.h>  #include <avr/interrupt.h> |

Для работы с таймером используется встроенный 8-ми битный таймер T1. Для данного таймера необходимо определить регистры: конфигурационный регистр TCCR1 и его биты, счётный регистр TCNT1 и регистр сравнения OCR1, а также регистр флагов разрешения прерываний TIMSK. (Листинг 2)

*Листинг 2. Определение регистров*

|  |
| --- |
| void initTimer() {  // CTC  TCCR1B |= (1 << WGM12);  // Enabling interrupt mask register to interrupt on OCIE1A  TIMSK |= (1 << OCIE1A);  OCR1A = 15625;  // Setting divider  TCCR1B |= (1 << CS10)|(1 << CS11);    sei();  } |

Следующая функция – это обработчик прерывания от таймера Т1, т. е. перечень действий, которые надо совершить в момент его возникновения. (Листинг 3)

*Листинг 3. Обработчик прерывания*

|  |
| --- |
| ISR (TIMER1\_COMPA\_vect) {  interruptCount++;    if (interruptCount == (t1Period \* k - 1)){  interruptFlags[0] = 1;  }  else if ((interruptCount + 1) % (t1Period \* k) == 0){  interruptFlags[0] = 1;  }  else if ((interruptCount) % (t1Period \* k) == 0){  interruptFlags[0] = 0;  }    if (interruptCount == (t2Period \* k - 1)){  interruptFlags[1] = 1;  }  else if ((interruptCount + 1) % (t2Period \* k) == 0){  interruptFlags[1] = 1;  }  else if ((interruptCount) % (t2Period \* k) == 0){  interruptFlags[1] = 0;  }      if (interruptCount == (t3Period \* k - 1)){  interruptFlags[2] = 1;  }  else if ((interruptCount + 1) % (t3Period \* k) == 0){  interruptFlags[2] = 1;  }  else if ((interruptCount) % (t3Period \* k) == 0){  interruptFlags[2] = 0;  }    if (interruptCount == (t4Period \* k - 1)){  interruptFlags[3] = 1;  }  else if ((interruptCount + 1) % (t4Period \* k) == 0){  interruptFlags[3] = 1;  }  else if ((interruptCount) % (t4Period \* k) == 0){  interruptFlags[3] = 0;  }      } |

Прерывание устанавливается в каждый момент времени, равный заданному времени для таймеров, а также для соответствующего периода.

Необходимо определить функцию инициализации таймера и функцию разрешения прерываний. Там же надо определить какие регистры будут принимать сигнал, а какие выдавать его. (Листинг 4)

*Листинг 4. Инициализация таймеров и портов*

|  |
| --- |
| int inputs[8];  int outputs[4];  void readInputs() {  for (int bit = 0; bit < 8; bit++) {  inputs[bit] = (PIND & (1 << bit)) >> bit;  }  } |

В основном цикле происходит чтение сигналов, определение сигналов для светодиодов и запись в порт. (Листинг 5)

*Листинг 5. Основный цикл*

|  |
| --- |
| void calculateOutputs(int\* x) {    int y1 = ((x[0] && !x[1] && x[2]) || (x[3] && !x[4] && x[5])) && interruptFlags[1];    int y2 = (x[2] && interruptFlags[0])|| (!x[4] && x[5] && interruptFlags[1]) || (!x[2] && x[6] && interruptFlags[2]) || (!x[3] && x[7] && interruptFlags[3]);    int y3 = (x[0] || !x[2] || x[4]) && interruptFlags[3];    int y4 = (x[0] && x[1] && x[2] && x[3] && x[4] && x[5] && x[6] && x[7]) || (!x[0] && !x[1] && !x[2] && !x[3] && !x[4] && !x[5] && !x[6] && !x[7]);      outputs[0] = y1;  outputs[1] = y2;  outputs[2] = y3;  outputs[3] = y4;  }  int main(void) {  initPorts();  initTimer();    while(1) {  readInputs();  calculateOutputs(inputs);    for (int i = 0; i < 4; i++) {  PORTA &= ~(1 << i);  PORTA |= outputs[i] << i;  }  }  } |

После сборки программы, нужно в Proteus прикрепить данную программу с расширением .hex.

Для проверки работы были поданы сигналы 1 0 1 0 1 0 0 0. Данная последовательность выдает сигналы y1 = 1, y3 = 1, y5 = 0. При секундах, деленных на 5 и 9 светодиод зажигается (Рис.3).

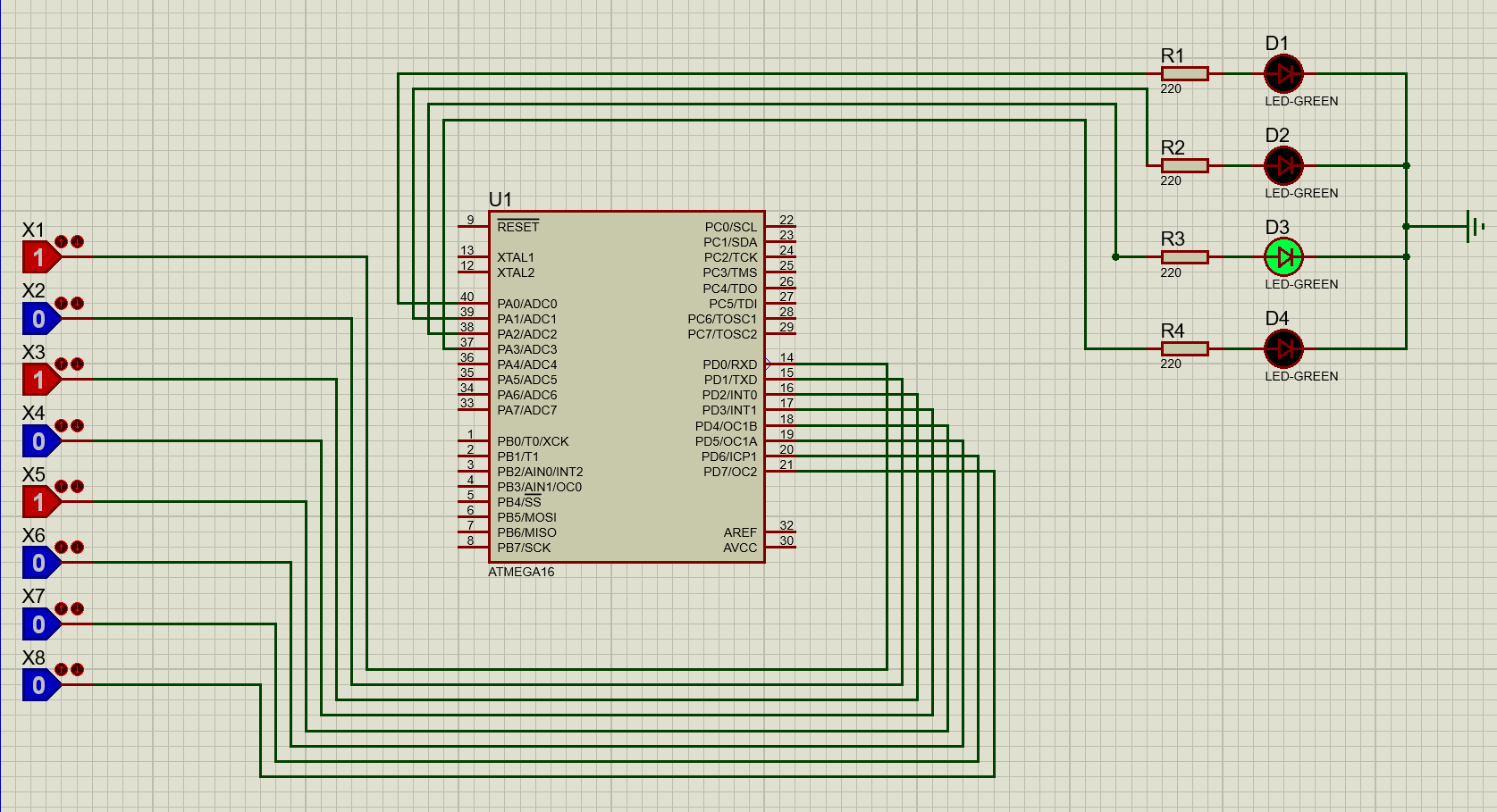


Рис. 3. Тестирование схемы

# Вывод

В ходе выполнения лабораторной работы, была разработана модель схемы, которая выполняет соответствующие функции с применением микроконтроллера. Была изучена работа микроконтроллера Atmega16 и работа с обработчиками прерываний. Также была изучена программа Microchip Studio и как с ней работать.

# Список литературы

1. Разработка и отладка микропроцессорных устройств в виртуальной среде моделирования Proteus [Электронный ресурс]: метод. указания / сост. В. Г. Иоффе. URL: http://repo.ssau.ru/bitstream/Metodicheskie-materialy/Razrabotka-iotladka-mikroprocessornyh-ustroistv-v-virtualnoi-srede-modelirovaniyaProteus-Elektronnyi-resurs-metod-ukazaniya-70958/1/Иоффе%20В.Г.%20Разработка%20и%20отладка%20микропроцессорных%20устройств.pdf

2. Филатов М. Синтез цифровых устройств комбинационного типа в Proteus 8.1 | Компоненты и технологии [Электронный ресурс] - URL : https://kite.ru/proteus-8-1-2-2/

3. Базовые элементы в Proteus 8 | RXRX [Электронный ресурс] - URL : https://rxtx.su/proteus/bazovye-elementy-v-proteus-8/

4. Встроенный таймеры и счетчики AVR микроконтроллеров |VPAYAEM.RU [Электронный ресурс] URL : https://vpayaem.ru/Atmega8\_timer.html

5. Павел Бобков. Учебный курс AVR. Таймер - счетчик Т0. Регистры. Ч1 [Электронный ресурс] — URL: https://chipenable.ru/index.php/programming-avr/171-avr-timer-t0-ch1.html

6. ATmega16, ATmega16L [Электронный ресурс] - URL: http://www.gaw.ru/html.cgi/txt/ic/Atmel/micros/avr/atmega16.htm?ysclid=m2rc6fuz7m3312980248. Приложение

|  |
| --- |
| #define F\_CPU 1000000UL  #include <avr/io.h>  #include <util/delay.h>  #include <avr/interrupt.h>  #define k 1  void initPorts() {  // Setting port to read  DDRD = 0x00;  PORTD = 0x00;    // Setting port to write  DDRA = 0xFF;  PORTA = 0xFF;  }  void initTimer() {  // CTC  TCCR1B |= (1 << WGM12);  // Enabling interrupt mask register to interrupt on OCIE1A  TIMSK |= (1 << OCIE1A);  OCR1A = 15625;  // Setting divider  TCCR1B |= (1 << CS10)|(1 << CS11);    sei();  }  int interruptCount = -1;  int interruptFlags[4] = {0, 0, 0, 0};  // Periods in seconds  const int t1Period = 3;  const int t2Period = 5;  const int t3Period = 7;  const int t4Period = 11;  // Handling timer interrupt  ISR (TIMER1\_COMPA\_vect) {  interruptCount++;    if (interruptCount == (t1Period \* k - 1)){  interruptFlags[0] = 1;  }  else if ((interruptCount + 1) % (t1Period \* k) == 0){  interruptFlags[0] = 1;  }  else if ((interruptCount) % (t1Period \* k) == 0){  interruptFlags[0] = 0;  }    if (interruptCount == (t2Period \* k - 1)){  interruptFlags[1] = 1;  }  else if ((interruptCount + 1) % (t2Period \* k) == 0){  interruptFlags[1] = 1;  }  else if ((interruptCount) % (t2Period \* k) == 0){  interruptFlags[1] = 0;  }      if (interruptCount == (t3Period \* k - 1)){  interruptFlags[2] = 1;  }  else if ((interruptCount + 1) % (t3Period \* k) == 0){  interruptFlags[2] = 1;  }  else if ((interruptCount) % (t3Period \* k) == 0){  interruptFlags[2] = 0;  }    if (interruptCount == (t4Period \* k - 1)){  interruptFlags[3] = 1;  }  else if ((interruptCount + 1) % (t4Period \* k) == 0){  interruptFlags[3] = 1;  }  else if ((interruptCount) % (t4Period \* k) == 0){  interruptFlags[3] = 0;  }      }  int inputs[8];  int outputs[4];  void readInputs() {  for (int bit = 0; bit < 8; bit++) {  inputs[bit] = (PIND & (1 << bit)) >> bit;  }  }  void calculateOutputs(int\* x) {    int y1 = ((x[0] && !x[1] && x[2]) || (x[3] && !x[4] && x[5])) && interruptFlags[1];    int y2 = (x[2] && interruptFlags[0])|| (!x[4] && x[5] && interruptFlags[1]) || (!x[2] && x[6] && interruptFlags[2]) || (!x[3] && x[7] && interruptFlags[3]);    int y3 = (x[0] || !x[2] || x[4]) && interruptFlags[3];    int y4 = (x[0] && x[1] && x[2] && x[3] && x[4] && x[5] && x[6] && x[7]) || (!x[0] && !x[1] && !x[2] && !x[3] && !x[4] && !x[5] && !x[6] && !x[7]);      outputs[0] = y1;  outputs[1] = y2;  outputs[2] = y3;  outputs[3] = y4;  }  int main(void) {  initPorts();  initTimer();    while(1) {  readInputs();  calculateOutputs(inputs);    for (int i = 0; i < 4; i++) {  PORTA &= ~(1 << i);  PORTA |= outputs[i] << i;  }  }  } |