Санкт-Петербургский государственный университет информационных технологий, механики и оптики

Кафедра “Компьютерные технологии”

Б.М. Ярцев, А.А. Шалыто

Проект “Бодрое утро”

Электронные часы с будильником

Проектная документация

Проект создан в рамках

“Движения за открытую проектную документацию”

http://is.ifmo.ru

Санкт-Петербург

2005

Оглавление

Введение

*Отец рассказывал мне, что в возрасте 7-8 лет
он обнаружил под рисунком в книге Жюль Верна
подпись: «Паганель бодро совал».*

*Несколько лет книга распространяла сладостный
аромат тайны. Каково же было разочарование, когда
выяснилось, что в действительности под рисунком
написано: «Паганель бодрствовал».*

*Виктор Ильич Варшавский.
Из воспоминаний "Поток сознания".*

Данный проект основан на курсовой работе по дисциплине "Контроллеры и управляющие системы". Было предложено проектировать управляющие системы для различных устройств, таких как телефон, домофон, микроволновая печь и. т. д. Предполагалось, что студенты спроектируют управляющую систему, а затем реализуют ее функциональность полностью или частично на микроконтроллерном стенде *SDK1.1*, изготавливаемом компанией *LMT* (http://lmt.ifmo.ru), который используется для обучения и отладки различных управляющих систем. Подробнее о нем можно узнать на сайте http://lmt.ifmo.ru/sdk1_1.html.

В качестве устройства для проектирования были выбраны часы-будильник. Этот выбор был сделан исходя из двух принципов. Во-первых, функциональность часов-будильника может быть практически полностью реализована на стенде *SDK1.1*, что крайне сложно сделать, например, для домофона или системы видеонаблюдения. Во-вторых, требования, предъявляемые к часам достаточно хорошо известны, что упрощает написание технического задания. Вначале, функциональность часов была реализована на программном эмуляторе, и только после этого – на микроконтроллерном стенде. При реализации на эмуляторе как система управления, так и программный эмулятор были написаны на языке *Java*. На этом эмуляторе была выполнена предварительная отладка системы управления.

Проект называется "Бодрое утро".

При проектировании системы управления использовалась технология автоматного подхода [1]. Подробнее об этой технологии можно узнать на сайте http://is.ifmo.ru. Данный проект демонстрирует преимущества автоматного подхода при реализации задач рассматриваемого класса на микроконтроллерах [2].

1. Постановка задачи

Спроектировать электронные часы-будильник в соответствии с описанием, приведенным ниже.

1.1. Внешний вид

На рис. 1 представлен вид часов спереди.

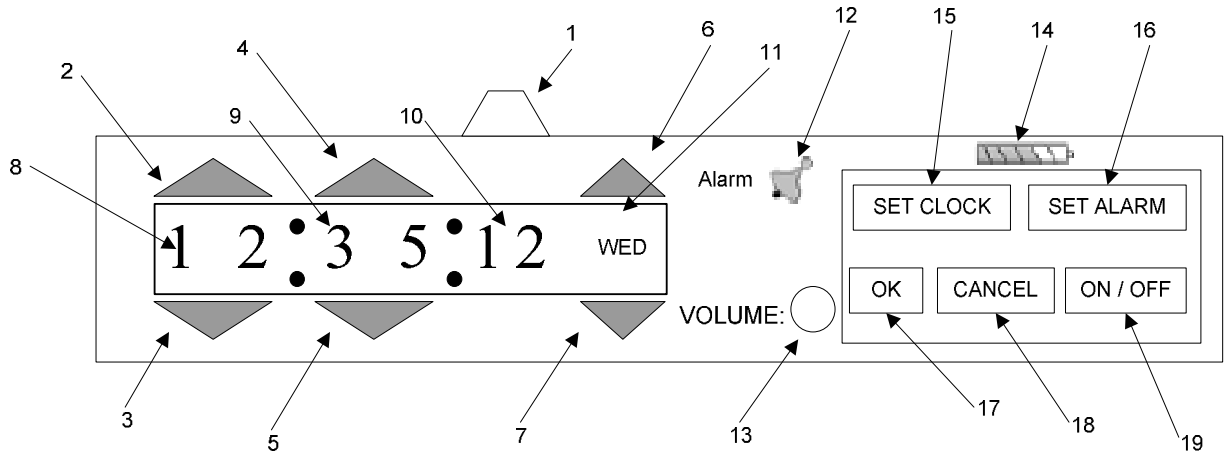

Рис. 1. Часы “Бодрое утро”. Вид спереди

На верхней панели часов находится кнопка выключения будильника 1. На передней панели часов предусмотрены кнопки редактирования времени часов и будильника 2 – 7. Дисплей позволяет отображать часы 8, минуты 9, секунды 10 и день недели 11. Световой индикатор 12 загорается при включении будильника. Аналоговый регулятор громкости будильника 13. Индикатор заряда батарей 14. Кнопка установки часов 15, кнопка редактирования будильника 16, а также кнопки, используемые для редактирования 17 – 19.

На рис. 2 представлен вид часов сзади.

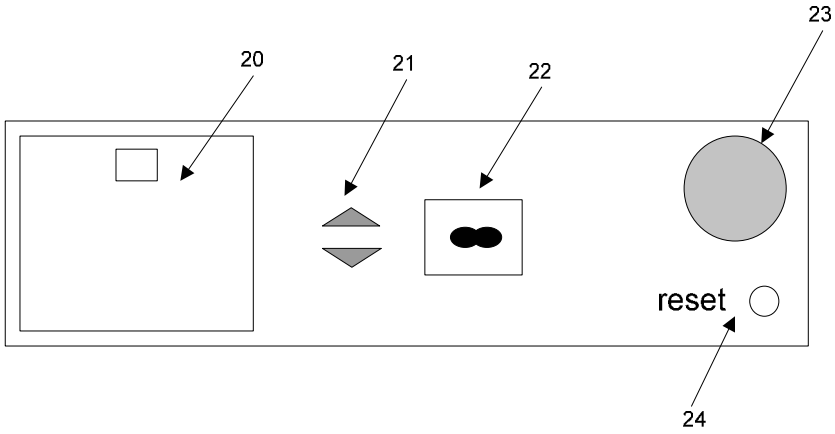

Рис. 2. Часы “Бодрое утро”. Вид сзади

На задней панели находится крышка, под которой расположены батареи 20, две кнопки для подстройки хода часов 21, вход для сетевого шнура 22, динамик 23 и кнопка сброса всех параметров 24.

1.2. Функциональность

1.2.1. Технические характеристики

Часы “Бодрое утро” имеют пять основных режимов работы: обычный режим, режим установки времени часов, режим установки времени будильника, режим корректировки времени и энергосберегающий режим. А также еще в каждом из этих режимов, возможен режим побудки (часы издают громкий звук). В обычном режиме на дисплей выводится текущее время. В режиме установки времени часов пользователь может настроить часы. В режиме установки времени будильника пользователь может настроить будильник. В режиме подстройки часов можно подстроить часы. Режим энергосбережения включается, когда заканчивается питание батарей. В этом режиме отключается звук. Часы могут работать как от батарей, так и от сети (220 v). Предусмотрена подстройка хода часов при помощи кнопок на задней панели 22. В часы встроена небольшая батарея, способная поддерживать ход часов и настройки будильника при отключении его от питания в течение долгого периода времени. При этом на дисплей ничего не выводится, а будильник не работает. Громкость будильника можно отрегулировать на передней панели при помощи аналогового регулятора 13.

1.2.2. Обычный режим

В обычном режиме на дисплей выводится текущее время 8 – 10 и день недели 11. Если будильник с ближайшим установленным временем активирован, то индикатор 12 горит, в противном случае этот индикатор не горит. Нажатие кнопки 15 переводит часы в режим установки времени часов, а нажатие кнопки 16 – в режим установки времени будильника. Кнопки 2 – 7 и 17 – 19 в обычном режиме не выполняют никаких действий. При нажатии на одну из кнопок подстройки времени, часы переходят в режим корректировки времени.

1.2.3. Режим установки времени часов

В этом режиме можно установить текущее время и день при помощи кнопок 2 – 7. При переходе в этот режим часы, минуты и день недели на дисплее принимают значение текущего времени и в дальнейшем не меняют своих значений. Секунды 10 становятся незажженными. При нажатии на кнопку 2 час увеличивается на единицу. При нажатии на кнопку 3 текущий час уменьшается на единицу. При нажатии на кнопки 4 или 5 текущая минута соответственно увеличивается или уменьшается на единицу. Кнопки 6 и 7 сменяют текущий день на следующий или предыдущий. При нажатии на кнопку 17 происходит возврат в обычный режим, установление отредактированного времени и вывод его на дисплей. Секунды инициализируются нулем. При нажатии на кнопку 18 происходит возврат в обычный режим без установки отредактированного времени. Остальные кнопки в этом режиме не выполняют никаких действий.

1.2.4. Режим установки времени будильника

В этот режим часы переходят при нажатии кнопки 16 в обычном режиме. Он предназначен для настройки будильника. Часы поддерживают возможность установки отдельного времени будильника на каждый из семи дней недели. День недели, на который

устанавливается будильник, высвечивается на дисплее 11. Секунды 10 в этом режиме незажжены и не используются. При переходе в этот режим, в качестве дня на который устанавливается будильник, выбирается завтрашний день (если вы выставляете, например, будильник в среду, то Вам будет предложено выставить будильник на четверг). Если будильник на этот день активирован, то индикатор 12 горит, в противном случае он не горит. Нажатие кнопки 19 переводит будильник на редактируемый день из активированного в неактивированное состояние и обратно. Время будильника выставляется при помощи кнопок 2 – 5. При помощи кнопок 6, 7 можно выбирать день недели, на который устанавливается будильник. При выборе другого дня, на который устанавливается будильник, на дисплей выводится время будильника, которое было установлено на этот день, индикатор 12 загорается или гаснет в зависимости от того активирован будильник на этот день или нет. По кнопке 17 производится сохранение установленного времени будильников и возврат в обычный режим. По кнопке 18 производится возврат в обычный режим. При этом установленное время не сохраняется.

В начале будильник на все дни деактивирован, время срабатывания будильника установлено на 10 часов утра.

1.2.5. Режим корректировки времени

В часы встроена поддержка корректировки хода на плюс или минус 30 с в течение месяца. В этот режим часы переходят при нажатии одной из кнопок 21. Час 8 незажжен, на индикаторе минут 9 горит либо “–“, либо ничего не горит. На индикаторе секунд горит текущее смещение хода часов. Если на индикаторе 9 – “–“, то смещение отрицательное, если этот индикатор не горит, то оно положительное. При помощи кнопок 22 можно редактировать смещение. При нажатии кнопки 17 производится сохранение установленного смещения и возврат в обычный режим. При нажатии кнопки 18 производится возврат в обычный режим без сохранения.

1.2.6. Режим энергосбережения

В этот режим часы переходят, когда в батареях осталось мало энергии. В этом режиме функциональность аналогична функциональности уже описанных четырех режимов за исключением того, что не работает звук.

1.2.7. Режим побудки

В этом режиме часы издают громкий звук. В этот режим происходит переход при выполнении двух условий – время на часах совпало со временем будильника на этот день будильник на этот день активирован. Звуковой сигнал играет в течение одной минуты. Его можно отключить при помощи кнопки (1) на верхней панели. Громкость сигнала можно установить при помощи регулятора (13). Переход в этот режим невозможен, если часы находятся в энергосберегающем режиме.

2. Архитектура проекта

2.1. Структурная схема

Структурная схема системы приведена на рис. 4.

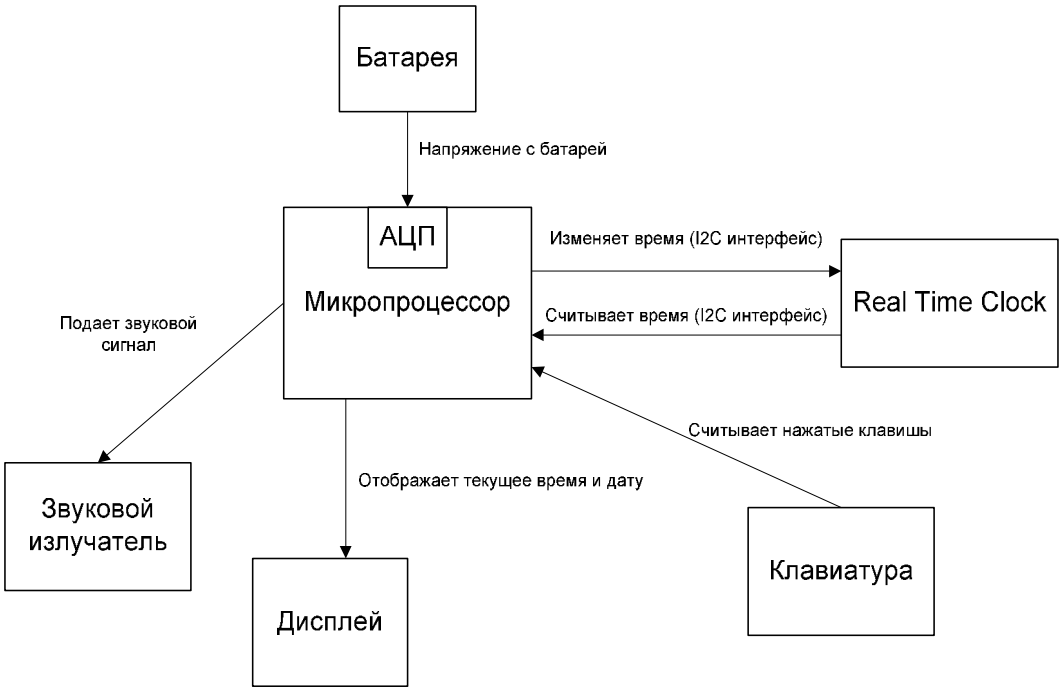

Рис. 3. Структурная схема

2.2. Управляющая система

Управляющая система спроектирована при помощи технологии автоматного программирования (http://is.ifmo.ru/). Она состоит из двух конечных автоматов `A0` и `A1`.

2.2.1. Автомат *A0*

Словесное описание

Автомат `A0` отвечает за обеспечение звука и переход при разряде батарей в энергосберегающий режим. Начальное состояние – “Инициализация”.

Схема связей автомата *A0*

Схема связей этого автомата приведена на рис. 4.

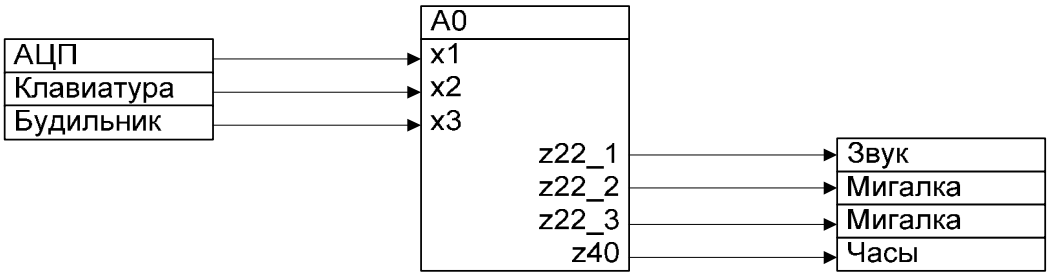

Рис. 4. Схема связей автомата *A0*

Описание входных переменных

`x1` – батареи разряжены;

`x2` – нажата кнопка. Принимает значение `STOP_ALARM_BTN`, формируемое кнопкой выключения звука;

`x3` – необходимо обеспечить звук;

Список выходных воздействий

`z22_1` – выдать короткий звуковой сигнал;

`z22_2` – мигнуть светодиодами;

`z22_3` – отключить светодиоды;

`z40` – инициализировать дисплей, секундный таймер, таймер бездействия, клавиатуру и часы реального времени.

Граф переходов автомата *A0*

Граф переходов этого автомата приведен на рис. 5.

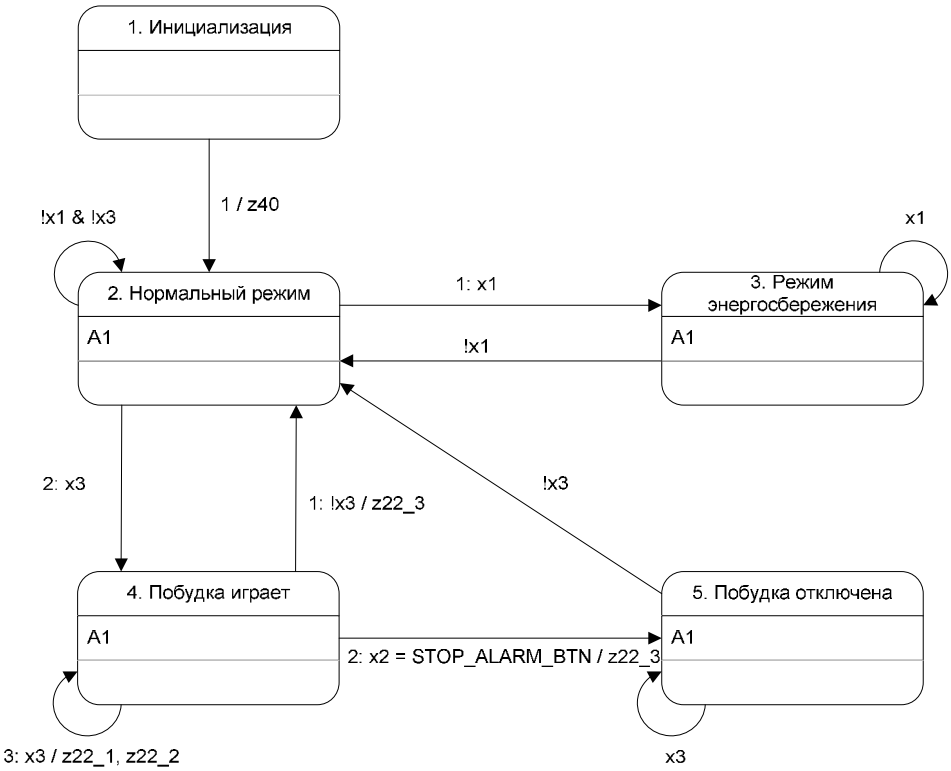

Рис. 5. Граф переходов автомата *A0*

2.2.2. Автомат *A1*

Словесное описание

Автомат `A1` отвечает за установку и подстройку времени, установку будильника, а также за отображение различной информации на дисплее. Начальное состояние – “Обычный режим”.

Схема связей автомата *A1*

Схема связей этого автомата приведена на рис. 6.

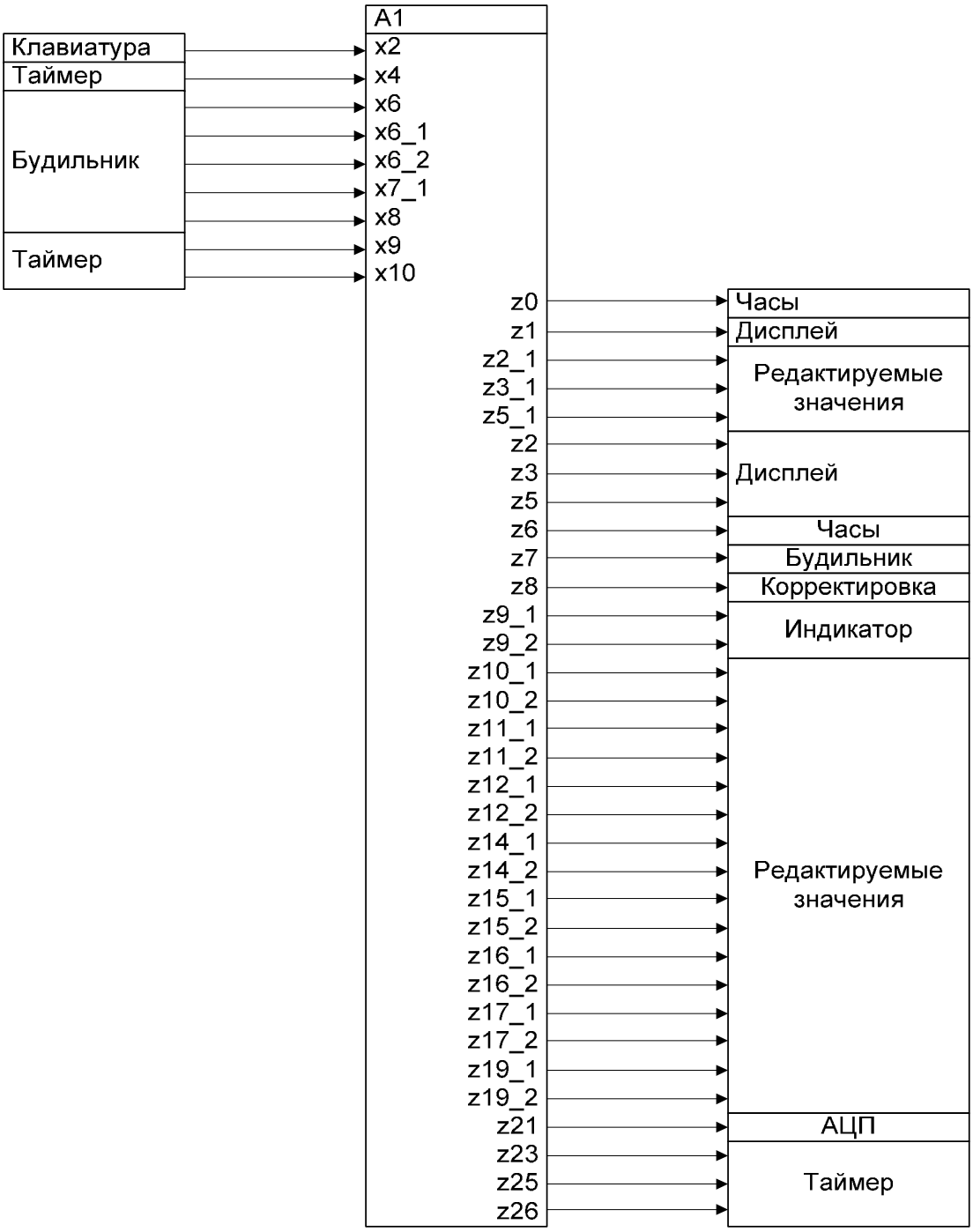

Рис. 6. Схема связей автомата *A1*

Список входных переменных

`x2` – нажата кнопка. Эта переменная принимает следующие значения:

- OK_BTN – от кнопки “OK”;
- CANCEL_BTN – от кнопки “CANCEL”;
- ONOFF_BTN – от кнопки “ON / OFF”;
- SET_CLOCK_BTN – от кнопки “SET CLOCK”;
- SET_ALARM_BTN – от кнопки “SET ALARM”;
- HOUR_P_BTN – от кнопки увеличения часа;

- HOUR_M_BTN – от кнопки уменьшения часа;
- MIN_P_BTN – от кнопки увеличения минуты;
- MIN_M_BTN – от кнопки уменьшения минуты;
- DAY_P_BTN – от кнопки увеличения дня;
- DAY_M_BTN – от кнопки уменьшения дня;
- ADJ_P_BTN – от кнопки увеличения величины подстройки;
- ADJ_M_BTN – от кнопки уменьшения величины подстройки;
- STOP_ALARM_BTN – от кнопки выключения побудки;

x4 – секундный таймер превысил значение одной секунды;

x6 – будильник на редактируемый день активирован;

x6_1– будильник на следующий за редактируемым днем активирован;

x6_2 – будильник на предыдущий за редактируемым днем будильник активирован;

x7_1 – будильник на следующий день активирован;

x8 – будильник на ближайшее время активирован;

x9 – таймер бездействия в каком-либо режиме превысил значение 20 с;

x10 – таймер обновления заряда батарей превысил значение 30 с.

Список выходных воздействий

z0 – считать текущее значение времени из часов реального времени;

z1 – вывести на дисплей текущее время;

z2_1 – подготовить редактируемое время часов;

z3_1 – подготовить редактируемое время будильника на неделю. Сделать текущим редактируемым временем следующий день;

z5_1 – подготовить редактируемую корректировку хода;

z2 – вывести на дисплей редактируемое время без секунд;

z3 – вывести на дисплей редактируемое время будильника на редактируемый день без секунд;

z5 – вывести на дисплей редактируемое значение корректировки хода;

z6 – сохранить значение времени;

z7 – сохранить значения будильников;

z8 – сохранить значение корректировки хода;

z9_1 – активировать индикатор будильника;

z9_2 – деактивировать индикатор будильника;

z10_1 – увеличить число часов в редактируемом времени на единицу;

z10_2 – уменьшить число часов в редактируемом времени на единицу;

z11_1 – увеличить число минут в редактируемом времени на единицу;

z11_2 – уменьшить число минут в редактируемом времени на единицу;

z12_1 – увеличить число дней в редактируемом времени на единицу;

z12_2 – уменьшить число дней в редактируемом времени на единицу;

z14_1 – сделать редактируемым следующий день;

z14_2 – сделать редактируемым предыдущий день;

z15_1 – увеличить число часов в редактируемом времени будильника на единицу;

z15_2 – уменьшить число часов в редактируемом времени будильника на единицу;

z16_1 – увеличить число минут в редактируемом времени на единицу;

z16_2 – уменьшить число минут в редактируемом времени на единицу;

z17_1 – активировать будильник на текущий день;

z17_2 – деактивировать будильник на текущий день;

z19_1 – увеличить корректирующее значение времени на одну секунду;

z19_2 – уменьшить корректирующее значение времени на одну секунду;

z21 – обновить индикатор состояния батарей в соответствии с зарядом на батареях;

z23 – сбросить секундный таймер;

z25 – сбросить таймер бездействия;

z26 – сбросить таймер обновления заряда батарей.

Граф переходов автомата *A1*

Граф переходов этого автомата приведен на рис. 7.

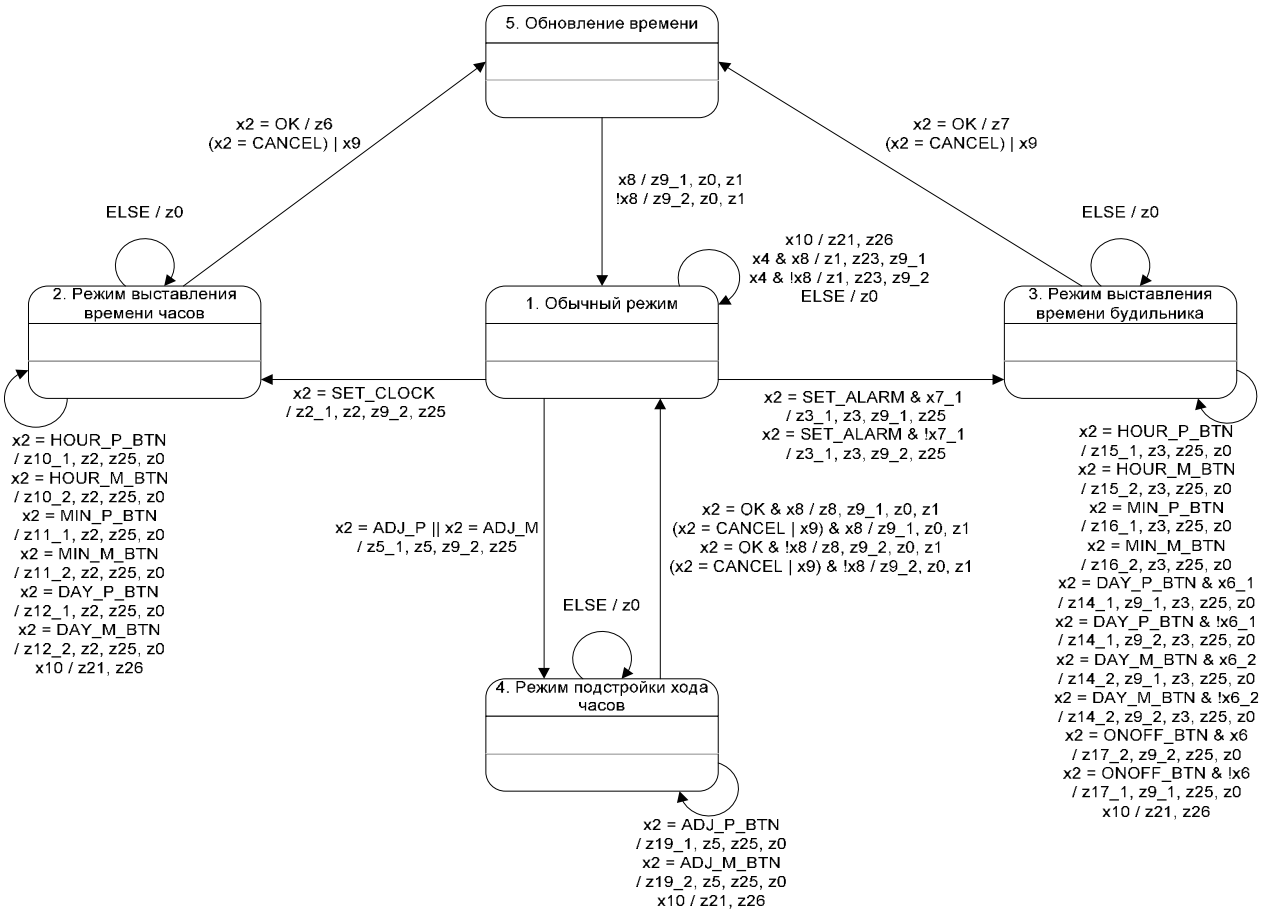

Рис. 7. Граф переходов автомата *A1*

3. Реализация

Функциональность часов-будильника была реализована на программном эмуляторе, написанном на языке *Java,* а затем на микроконтроллерном стенде *SDK1.1*.

3.1. Реализация на языке Java

Вначале была выполнена реализация на программном эмуляторе, написанном на языке *Java*. Для запуска программного эмулятора с управляющей программой необходимо *JRE 5.0* (*Java Runtime Environment 5.0*). Для компиляции используется *JDK 5.0*.

При разработке эмулятора была написана программа, эмулирующая интерфейс часов. После этого по схемам связей и графам переходов была реализована управляющая программа. Эта управляющая программа полностью реализует всю заявленную функциональность. Исключение состоит в том, что значение корректировки времени не оказывает никакого влияния на ход часов. На рис. 8 показан внешний вид программного эмулятора. В качестве сигнала звучит короткий музыкальный фрагмент (соло из песни "Мамба" музыкально-инструментального ансамбля "Ленинград").

Рис. 8. Программный эмулятор. Внешний вид

Кнопки на стенде соответствуют кнопкам часов. Бегунок справа позволяет выставить заряд батарей. Величина заряда отображается при помощи полоски зеленого цвета.

По нажатию кнопки “RESET” происходит перезагрузка часов и инициализация всех параметров (время, корректировка, будильник) значениями по умолчанию.

Когда играет побудка, то в правом нижнем углу начинает мигать светодиод (рис. 9).

Рис. 9. Программный эмулятор. Играет побудка

Функциональность программного эмулятора совпадает с функциональностью управляющей программы, реализованной на стенде.

3.2. Шаблон проектирования для языка Java

Управляющая система реализована как внутренний класс программного эмулятора.

Каждой входной переменной соответствует метод класса управляющей системы с именем, совпадающим с именем входной переменной, которая возвращает либо булевское, либо целое (переменная `x2`) значение. Например,

```
//Функция, соответствующая входной переменной x6_2
private boolean x6_2() {
     int prevDay = editedAlarmDay + 6;
     prevDay %= 7;
     return editedActive[prevDay];
}
```

Каждому выходному воздействию соответствует метод с именем, совпадающим с именем выходного воздействия, которое возвращает пустое значение.

```
//функция, соответствующая выходному воздействию z1
private void z1() {
     clockPanel.updateTime(currentTime);
```

```
}
```

Идентификаторы состояний автоматов реализованы при помощи констант. Например,

```
//Состояние "Обычный режим" автомата 1
public static final int A1_NORMAL = 1;
//Состояние "Редактирование времени" автомата 1
public static final int A1_TIME_EDIT = 2;
```

Автоматные переменные, хранящие текущее состояние автомата реализованы при помощи целочисленных переменных. Например,

```
//Автоматная переменная автомата A1, инициализированная
//состоянием "Обычный режим"
private int a0state;
```

Изоморфная реализация автоматных функций выполняется по шаблону приведенному на рис 16.

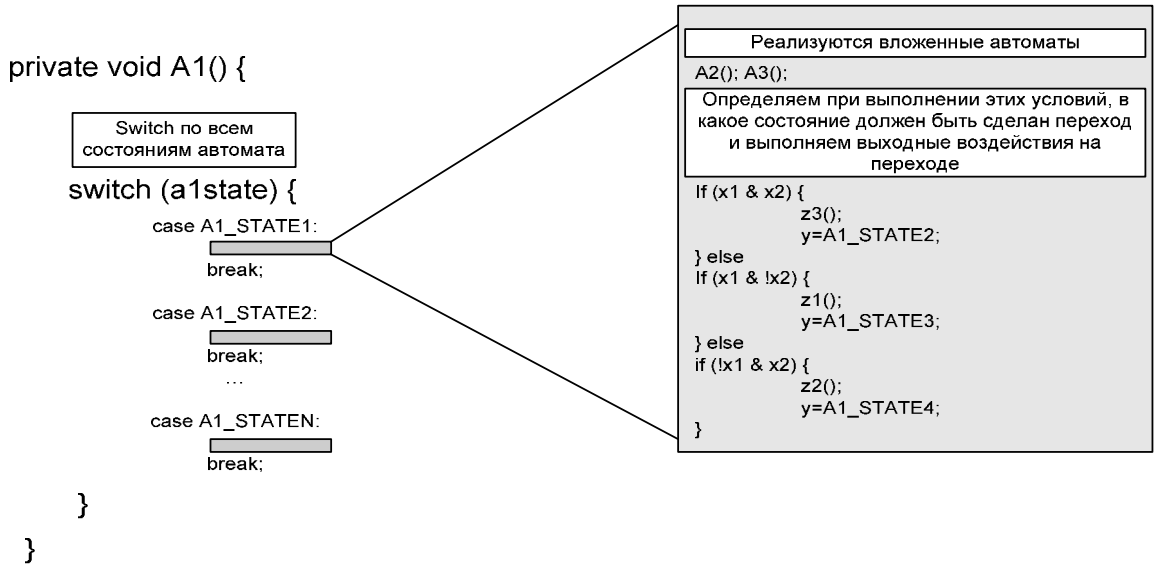

Рис. 10. Построение автоматной функции по графу переходов

Следующая функция постоянно вызывается в бесконечном цикле. В ней вызывается автоматная функция. Также как и в случае с микроконтроллерным стендом значения функций входных переменных сначала считываются в соответствующие переменные:

```
public void tick() {
     x1 = x1();
     x2 = x2();
     x3 = x3();
     x4 = x4();
     x6 = x6();
     x6_1 = x6_1();
     x6_2 = x6_2();
     x7_1 = x7_1();
     x8 = x8();
     x9 = x9();
     x10 = x10();
     a0();
```

```
}
```

Как выглядит основная функция `main`? Из нее в бесконечном цикле вызывается функция `clock.autoTick()`, а уже из этой функции, в свою очередь, вызывается непосредственно функция `tick()`.

```
public static void main(String[] args) {
     ClockWindow clock = new ClockWindow();
     while(true) {
          try {
               Thread.sleep(10);
               clock.autoTick();
          } catch (InterruptedException e) {
               e.printStackTrace();
          }
     }
}

public void autoTick() {
     auto.tick();
}
```

Исходный код программного эмулятора приведен в Приложении 1.

3.3. Реализация на стенде SDK 1.1

В данном проекте не приводится подробное описание учебного стенда *SDK1.1*. Подробное описание доступно на сайте компании *LMT* (http://lmt.ifmo.ru/sdk1_1.html).

Компиляция исходного кода управляющей программы для микроконтроллерного стенда производится при помощи пакета *uVision2/ARM Developers Kit* фирмы *Keil Software*. Процесс компиляции происходит следующим образом – вначале программа, написанная на языке *C* (Приложение 2) при помощи компилятора *C51* преобразуется в ассемблер для процессора *ADUC 812*, используемого в стенде. После этого этот код компилируется в объектный файл при помощи компилятора *A51* и линкуется с необходимой библиотекой `startup` при помощи линкера *BL51*. Далее полученный объектный файл преобразуется в *HEX*-формат при помощи конвертера *OH51* для дальнейшей загрузки на микроконтроллерный стенд. *BAT*-файлы, автоматизирующие компиляцию и загрузку, а также исходный код самой управляющей программы, размещены по адресу http://is.ifmo.ru.

На стенде *SDK1.1* удалось реализовать практически всю описанную функциональность. Не была реализована только работа с батареями (по причине их отсутствия). Поэтому в выходном воздействии `z21()` не выполняется никакого кода, а входная переменная `x1` всегда принимает значение `false`. Значение корректировки времени сохраняется в памяти микроконтроллера. Это значение можно редактировать, но никакого влияния на ход встроенных часов реального времени не оказывает.

Для микроконтроллерного стенда SDK 1.1 не существует встроенного в него отладчика. Поэтому была использована одна из **особенностей автоматного подхода** – наблюдаемость за состояниями. На дисплей микроконтроллера выводится

идентификаторы текущих состояний автоматов. Это заметно облегчает работу и тестирование управляющей программы. Программист может отследить все изменения состояний, а затем на основе этой информации исправить ошибки логики как на графе переходов, так и в коде входных переменных и выходных воздействий.

3.4. Интерфейс стенда SDK 1.1

Стенд *SDK1.1* с загруженной управляющей программой часов показан на рис. 11.

Рис. 11. Стенд SDK1.1 с загруженной управляющей программой. Обычный режим

В верхней части стенда размещен ЖК-дисплей, на который выводится вся информация. В данном случае часы находятся в обычном режиме. Три числа на первой строчке, разделенные двоеточием, показывают текущее время в формате ЧЧ:ММ:СС. Далее следует трехбуквенный идентификатор текущего дня недели: MON, TUE, WED, THU, FRI, SAT или SUN. Идентификатор ON высвечивается на второй строчке ЖК-дисплея, если будильник на ближайшее время активирован (`x8 == true`), в противном случае высвечивается OFF. Две цифры на второй строчке используются для отладки – **они соответствуют состояниям автоматов `A0` и `A1`**. В данный момент состояние автомата `A0` – “Нормальный режим”, а автомата `A1` – “Обычный режим”.

Кнопки часов соответствуют кнопкам на стенде:

- кнопка “OK” соответствует кнопке “9” стенда;
- кнопка “CANCEL” – кнопке “C”;
- кнопка “ON / OFF” – кнопке “D”;

- кнопка “SET CLOCK” – кнопке “7”;
- кнопка “SET ALARM” – кнопке “8”;
- кнопка увеличения часа – кнопке “1”;
- кнопка уменьшения часа – кнопке “4”;
- кнопка увеличения минуты – кнопке “2”;
- кнопка уменьшения минуты – кнопке “5”;
- кнопка увеличения дня – кнопке “3”;
- кнопка уменьшения дня – кнопке “6”;
- кнопка увеличения величины подстройки – кнопке “A”;
- кнопка уменьшения величины подстройки – кнопке “B”;
- кнопка выключения побудки – кнопке “*”;

Перезагрузка часов выполняется при перезагрузке самого стенда при помощи встроенного выключателя.

Звук обеспечивается при помощи встроенного в стенд динамика. При побудке мигают также светодиоды на стенде. На рис. 12 показан контроллер в режиме установки времени.

Рис. 12. Установка времени

На рис. 13 показан режим установки будильника.

Рис. 13. Установка будильника

На рис. 14 показан режим установки значения корректировки времени.

Рис. 14. Установка корректировки хода

На рис. 15 показан режим, в котором мигают светодиоды и играет побудка.

Рис. 15. Побудка играет. Мигают светодиоды слева от кнопок

3.5. Шаблон проектирования для микроконтроллера

По спроектированным схемам связей и графам переходов создана управляющая программа на языке *C*.

Каждой входной переменной соответствует функция языка *C* с именем, совпадающим с именем входной переменной, которая возвращает целое значение. Например,

```
//Функция, соответствующая входной переменной x6
uint x6() {
     return (editedAlarmTime[editedDay].active);
}
```

Каждому выходному воздействию соответствует функция с именем, совпадающим с именем выходного воздействия, которая возвращает пустое значение. Например,

```
//Функция, соответствующая выходному воздействию z5_1
void z5_1() {
     editedCorrectValue = correctValue;
}
```

Идентификаторы состояний автоматов реализованы при помощи констант:

```
//Состояние "Обычный режим" автомата 1
#define A1_NORMAL 1
//Состояние "Редактирование времени" автомата 1
#define A1_TIME_EDIT 2
```

Автоматные переменные, хранящие текущее состояние автомата, реализованы при помощи целочисленных переменных. Например,

```
//Автоматная переменная автомата A1, инициализированная
//состоянием "Обычный режим"
uint y1 = A1_NORMAL;
```

Изоморфная реализация автоматных функций выполняется по шаблону, приведенному на рис. 16.

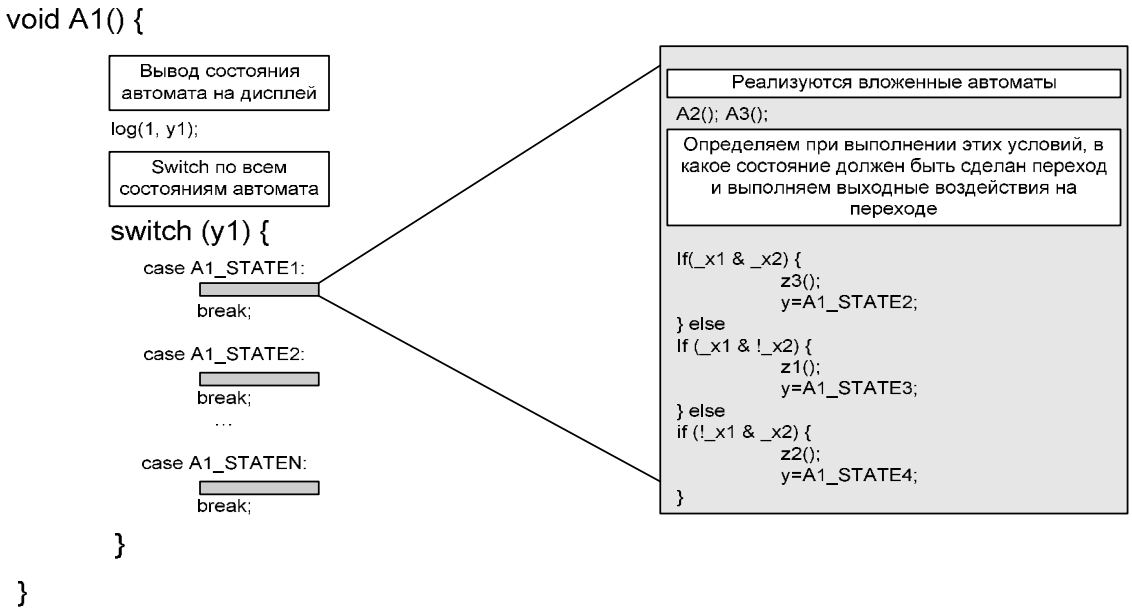

Рис. 16. Построение автоматной функции по графу переходов

Функция `main()` в программе выглядит следующим образом:

```
void main(void) {
     while (1) {
          //Обновить значения входных переменных
          UpdateVariables();
          //Вызвать автоматную функцию
          A0();
     }
}
```

Отметим, почему на рис. 16 при проверке значений входных переменных вызываются не сами функции, соответствующие входным переменным, а некие переменные, имена которых образованы путем добавления символа подчеркивания к имени входной переменной. Дело в том, что значение функции, соответствующей входной переменной может измениться в течение одного вызова автоматной функции. К тому же, вызов некоторых таких функций может быть долгим. Поэтому перед выполнением очередного цикла в функции `main()` значения функций считываются в соответствующие переменные (смотри далее функцию `UpdateVariables()`), а в самих автоматных функциях проверяется уже значения этих переменных.

```
uint _x1;
uint _x2;
uint _x3;
```

```
uint _x4;
uint _x6;
uint _x6_1;
uint _x6_2;
uint _x7_1;
uint _x8;
uint _x9;
uint _x10;
void UpdateVariables() {
     _x1 = x1();
     _x2 = x2();
     _x3 = x3();
     _x4 = x4();
     _x6 = x6();
     _x6_1 = x6_1();
     _x6_2 = x6_2();
     _x7_1 = x7_1();
     _x8 = x8();
     _x9 = x9();
     _x10 = x10();
}
```

Следует отметить, что в коде программы функция автомата `A1` реализована до функции автомата `A0`. Это сделано из тех соображений, что функция автомата `A1` вызывается из функции автомата `A0` и должна быть определена заранее.

Исходный код программного эмулятора приведен в Приложении 1.

Заключение

Данный проект демонстрирует применение автоматного подхода к проектированию и программированию микроконтроллеров. Подход требует подробного документирования, но сильно упрощает отладку.

Список литературы

1. *Шалыто А.А.* SWITCH-технология. Алгоритмизация и программирование задач логического управления. СПб.: Наука, 1998.

2. *Черномырдин А.В.* Автоматное программирование для микроконтроллеров. SWITCH-технология и интерпретаторы // Радиолюбитель. 2005. № 8.

Приложение 1. Код программного эмулятора часов

```
import sun.audio.AudioPlayer;
import sun.audio.AudioStream;
import sun.audio.AudioData;
import sun.audio.ContinuousAudioDataStream;

import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;
import java.awt.event.AdjustmentListener;
import java.awt.event.AdjustmentEvent;
import java.io.FileInputStream;
import java.io.InputStream;
import java.io.IOException;

/**
 * User: Boris Yartsev
 * Date: 02.05.2005
 * Time: 19:00:47
 */
public class ClockWindow extends JFrame implements ActionListener {
    private RTC rtc;
    private Battery batt;

    private ClockPanel clockPanel = new ClockPanel();

    private JButton okBn = new JButton("OK");
    public static final int OK_BTN = 1;
    private JButton cancelBn = new JButton("CANCEL");
    public static final int CANCEL_BTN = 2;
    private JButton onoffBn = new JButton("ON/OFF");
    public static final int ONOFF_BTN = 3;
    private JButton setClockBn = new JButton("SET CLOCK");
    public static final int SET_CLOCK_BTN = 4;
    private JButton setAlarmBn = new JButton("SET ALARM");
    public static final int SET_ALARM_BTN = 5;
    private JButton hourPlBn = new JButton("Hour+");
    public static final int HOUR_P_BTN = 6;
    private JButton hourMinBn = new JButton("Hour-");
    public static final int HOUR_M_BTN = 7;
    private JButton minPlBn = new JButton("Min+");
    public static final int MIN_P_BTN = 8;
    private JButton minMinBn = new JButton("Min-");
    public static final int MIN_M_BTN = 9;
    private JButton dayPlBn = new JButton("Day+");
    public static final int DAY_P_BTN = 10;
    private JButton dayMinBn = new JButton("Day-");
    public static final int DAY_M_BTN = 11;
    private JButton adjPlBn = new JButton("Adj+");
    public static final int ADJ_P_BTN = 12;
    private JButton adjMinBn = new JButton("Adj-");
    public static final int ADJ_M_BTN = 13;
    private JButton stopAlarmBn = new JButton("STOP ALARM");
    public static final int STOP_ALARM_BTN = 14;
    private JButton resetBn = new JButton("RESET");

    private JScrollBar batteryControl = new JScrollBar();
```

```
private BatteryView batteryLevel = new BatteryView(100);
private Blinker blinker = new Blinker();
private BellPlayer player = new BellPlayer("tada4.wav");

private int pressedBtnInd = 0;

private Automaton auto = new Automaton();

public ClockWindow() {
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    rtc = new RTC();
    batt = new Battery();

    Container cp = getContentPane();
    cp.setLayout(new GridBagLayout());
    GridBagConstraints gc = new GridBagConstraints();
    gc.insets = new Insets(5, 5, 5, 5);

    gc.fill = GridBagConstraints.BOTH;
    gc.weightx = 1.0;
    gc.weighty = 1.0;

    gc.gridx = 0;
    gc.gridy = 0;
    gc.gridwidth = 2;
    gc.gridheight = 1;

    cp.add(hourPlBn, gc);
    hourPlBn.addActionListener(this);

    gc.gridx = 0;
    gc.gridy = 3;

    cp.add(hourMinBn, gc);
    hourMinBn.addActionListener(this);

    gc.gridx = 2;
    gc.gridy = 0;

    cp.add(minPlBn, gc);
    minPlBn.addActionListener(this);

    gc.gridx = 2;
    gc.gridy = 3;

    cp.add(minMinBn, gc);
    minMinBn.addActionListener(this);

    gc.gridx = 6;
    gc.gridy = 0;

    cp.add(dayPlBn, gc);
    dayPlBn.addActionListener(this);

    gc.gridx = 6;
    gc.gridy = 3;

    cp.add(dayMinBn, gc);
    dayMinBn.addActionListener(this);

    gc.gridx = 0;
    gc.gridy = 1;
    gc.gridwidth = 16;
```

```
gc.gridheight = 2;

cp.add(clockPanel, gc);

gc.gridx = 18;
gc.gridy = 0;
gc.gridwidth = 3;
gc.gridheight = 1;

cp.add(setClockBn,  gc);
setClockBn.addActionListener(this);

gc.gridx = 21;

cp.add(setAlarmBn,  gc);
setAlarmBn.addActionListener(this);

gc.gridx = 18;
gc.gridy = 1;
gc.gridwidth = 2;
gc.gridheight = 1;

cp.add(okBn, gc);
okBn.addActionListener(this);

gc.gridx = 20;

cp.add(cancelBn, gc);
cancelBn.addActionListener(this);

gc.gridx = 22;

cp.add(onoffBn, gc);
onoffBn.addActionListener(this);

gc.gridx = 18;
gc.gridy = 2;

cp.add(adjPlBn, gc);
adjPlBn.addActionListener(this);

gc.gridx = 18;
gc.gridy = 3;

cp.add(adjMinBn, gc);
adjMinBn.addActionListener(this);

gc.gridx = 21; gc.gridwidth = 1;
cp.add(batteryLevel, gc);

gc.gridx = 22;
cp.add(blinker, gc);


gc.gridx = 20;
gc.gridy = 2;

cp.add(stopAlarmBn, gc);
stopAlarmBn.addActionListener(this);

gc.gridx = 22;
gc.gridy = 2;
```

```
        cp.add(resetBn, gc);
        resetBn.addActionListener(this);

        gc.gridx = 24;
        gc.gridy = 0;
        gc.gridwidth = 1;
        gc.gridheight = 4;
        cp.add(batteryControl, gc);
        batteryControl.setValue(80);
        batteryControl.addAdjustmentListener(
                new AdjustmentListener() {
                    public void adjustmentValueChanged(AdjustmentEvent e) {
                        batt.setCharge(e.getValue() / 100.0);
                    }
                }
        );

        setSize(600, 300);
        setVisible(true);
    }

    private class Automaton {
        public static final int MAX_CORRECT_VALUE = 30;

        public static final int WEEK_DAYS = 7;

        private Time currentTime;
        private Time editedClockTime;

        /**
         * Установки будильника
         */
        private Time alarmTime[];
        private boolean alarmActive[];

        /**
         * Редактируемые установки будильника
         */
        private Time editedAlarmTime[];
        private int editedAlarmDay;
        private boolean editedActive[];


        /**
         * Значение поправки
         */
        private int correctValue = 0;

        /**
         * Редактируемое значение поправки
         */
        private int editCorrectValue = 0;


        private int a0state;
        private int a1state;

        /**
         * Таймеры
         */
        private long secondTimer = 0;
        public static final int SECOND_VALUE = 1000;
        private long inactivityTimer = 0;
```

```
public static final int INACTIVITY_VALUE = 20000;
private long batteryTimer = 0;
public static final int BATTERY_VALUE = 10;

/**
 * Значения входных переменных
 */
private boolean x1;
private int x2;
private boolean x3;
private boolean x4;
private boolean x6;
private boolean x6_1;
private boolean x6_2;
private boolean x7_1;
private boolean x8;
private boolean x9;
private boolean x10;

public static final int A0_INIT = 1;
public static final int A0_NORMAL = 2;
public static final int A0_ENERGY_SAVING = 3;
public static final int A0_ALARM_PLAYS = 4;
public static final int A0_ALARM_OFF = 5;

public static final int A1_NORMAL = 1;
public static final int A1_TIME_EDIT = 2;
public static final int A1_ALARM_EDIT = 3;
public static final int A1_CORRECT_EDIT = 4;
public static final int A1_TIME_UPDATE = 5;


private int charge;

public static final int LOW_LIMIT = 20;

public Automaton() {
    init();
}

public void init() {
    a0state = A0_INIT;
    a1state = A1_NORMAL;

    currentTime = new Time();
    editedClockTime = new Time();
    alarmTime = new Time[WEEK_DAYS];
    alarmActive = new boolean[WEEK_DAYS];
    editedAlarmTime = new Time[WEEK_DAYS];
    editedAlarmDay = 0;
    editedActive = new boolean[WEEK_DAYS];
    correctValue = 0;
    editCorrectValue = 0;
    secondTimer = 0;
    inactivityTimer = 0;
    batteryTimer = 0;

    for (int i = 0; i < WEEK_DAYS; i++) {
        alarmTime[i] = new Time();
        editedAlarmTime[i] = new Time();
    }

    blinker.switchOff();
```

```
    player.stopPlaying();
}

public void restart() {
    init();
}

private boolean x1() {
    return charge < LOW_LIMIT;
}

private int x2() {
    int retValue = pressedBtnInd;
    pressedBtnInd = 0;
    return retValue;
}

private boolean x3() {
    Time ct = currentTime;
    int days = ct.day;
    int hour = ct.hours;
    int min = ct.minutes;
    int alhour = alarmTime[days].hours;
    int almin = alarmTime[days].minutes;

    if (!alarmActive[days]) {
        return false;
    }

    return (alhour == hour && almin == min);
}

private boolean x4() {
    return System.currentTimeMillis() - secondTimer >= SECOND_VALUE;
}

private boolean x6() {
    return editedActive[editedAlarmDay];
}

private boolean x6_1() {
    int nextDay = editedAlarmDay + 1;
    nextDay %= 7;
    return editedActive[nextDay];
}

private boolean x6_2() {
    int prevDay = editedAlarmDay + 6;
    prevDay %= 7;
    return editedActive[prevDay];
}

private boolean x7_1() {
    int nextDay = currentTime.day + 1;
    nextDay %= 7;
    return alarmActive[nextDay];
}

private boolean x8() {
    Time ct = currentTime;
    int day = ct.day;

    if (ct.hours < alarmTime[day].hours ||
```

```
            ((ct.hours == alarmTime[day].hours) &&
            (ct.minutes < alarmTime[day].minutes))) {
        return alarmActive[day];
    } else {
        day++;
        day %= 7;
        return alarmActive[day];
    }
}

private boolean x9() {
    return System.currentTimeMillis() -
            inactivityTimer > INACTIVITY_VALUE;
}

private boolean x10() {
    return System.currentTimeMillis() - batteryTimer > BATTERY_VALUE;
}


private void z0() {
    if (rtc.secChanged()) {
        currentTime.nextSec();
        rtc.dropChanged();
    }

}

/**
 * Вывести на дисплей текущее время.
 */
private void z1() {
    clockPanel.updateTime(currentTime);
    blinker.repaint();
    batteryLevel.repaint();
}

/**
 * Вывести на дисплей редактируемое время без секунд.
 */
private void z2() {
    clockPanel.updateTimeWithoutSeconds(editedClockTime);
}

/**
 * Вывести на дисплей редактируемое время будильника на редактируемый
 * день без секунд.
 */
private void z3() {
clockPanel.updateTimeWithoutSeconds(editedAlarmTime[editedAlarmDay]);
}

/**
 * Вывести на дисплей редактируемое значение корректировки хода.
 */
private void z5() {
    clockPanel.updateCorrectTime(editCorrectValue);
}

/**
 * Подготовить редактируемое время часов.
 */
private void z2_1() {
```

```
    Time time = currentTime;
    editedClockTime.day = time.day;
    editedClockTime.hours = time.hours;
    editedClockTime.minutes = time.minutes;
    editedClockTime.seconds = 0;
}

/**
 * Подготовить редактируемое время будильника на неделю. Сделать
 *текущим редактируемым временем следующий день.
 */
private void z3_1() {
    for (int i = 0; i < WEEK_DAYS; i++) {
        editedActive[i] = alarmActive[i];
        editedAlarmTime[i].minutes = alarmTime[i].minutes;
        editedAlarmTime[i].hours = alarmTime[i].hours;
        editedAlarmTime[i].day = alarmTime[i].day;
    }

    editedAlarmDay = (currentTime.day + 1) % 7;
}

/**
 * Подготовить редактируемую корректировку хода.
 */
private void z5_1() {
    editCorrectValue = correctValue;
}

/**
 * Сохранить значение времени.
 */
private void z6() {
    currentTime.seconds = editedClockTime.seconds;
    currentTime.minutes = editedClockTime.minutes;
    currentTime.hours = editedClockTime.hours;
    currentTime.day = editedClockTime.day;
}

/**
 * Сохранить значения будильников.
 */
private void z7() {
    for (int i = 0; i < WEEK_DAYS; i++) {
        alarmActive[i] = editedActive[i];
        alarmTime[i].minutes = editedAlarmTime[i].minutes;
        alarmTime[i].hours = editedAlarmTime[i].hours;
    }
}

/**
 * Сохранить значение корректировки хода.
 */
private void z8() {
    correctValue = editCorrectValue;
}

/**
 * Активировать индикатор будильника.
 */
private void z9_1() {
    clockPanel.updateAlarmActive(true);
    //alarmActiveLabel.setText("ON");
```

```
}

/**
 * Деактивировать индикатор будильника.
 */
private void z9_2() {
    clockPanel.updateAlarmActive(false);
    //alarmActiveLabel.setText("OFF");
}

/**
 * Увеличить часы в редактируемом времени на 1.
 */
private void z10_1() {
    editedClockTime.hours++;
    editedClockTime.hours %= 24;
}

/**
 * Уменьшить часы в редактируемом времени на 1.
 */
private void z10_2() {
    editedClockTime.hours--;
    editedClockTime.hours = (editedClockTime.hours + 24) % 24;
}

/**
 * Увеличить минуты в редактируемом времени на 1.
 */
private void z11_1() {
    editedClockTime.minutes++;
    editedClockTime.minutes %= 60;
}

/**
 * Уменьшить минуты в редактируемом времени на 1.
 */
private void z11_2() {
    editedClockTime.minutes--;
    editedClockTime.minutes = (editedClockTime.minutes + 60) % 60;
}

/**
 * Увеличить дни в редактируемом времени на 1.
 */
private void z12_1() {
    editedClockTime.day++;
    editedClockTime.day %= 7;
}

/**
 * Уменьшить дни в редактируемом времени на 1.
 */
private void z12_2() {
    editedClockTime.day--;
    editedClockTime.day = (editedClockTime.day + 7) % 7;
}

/**
 * Сделать редактируемым следующий день.
 */
private void z14_1() {
    editedAlarmDay ++;
```

```
    editedAlarmDay %= 7;
}

/**
 * Сделать редактируемым предыдущий день.
 */
private void z14_2() {
    editedAlarmDay += 6;
    editedAlarmDay %= 7;
}

/**
 * Увеличить часы в редактируемом времени будильника на 1.
 */
private void z15_1() {
    editedAlarmTime[editedAlarmDay].hours++;
    editedAlarmTime[editedAlarmDay].hours %= 24;
}

/**
 * Уменьшить часы в редактируемом времени будильника на 1.
 */
private void z15_2() {
    editedAlarmTime[editedAlarmDay].hours += 23;
    editedAlarmTime[editedAlarmDay].hours %= 24;
}

/**
 * Увеличить минуты в редактируемом времени будильника на 1.
 */
private void z16_1() {
    editedAlarmTime[editedAlarmDay].minutes++;
    editedAlarmTime[editedAlarmDay].minutes %= 60;
}

/**
 * Уменьшить минуты в редактируемом времени будильника на 1.
 */
private void z16_2() {
    editedAlarmTime[editedAlarmDay].minutes += 59;
    editedAlarmTime[editedAlarmDay].minutes %= 60;
}

/**
 * Активировать будильник на текущий день.
 */
private void z17_1() {
    editedActive[editedAlarmDay] = true;
}

/**
 * Деактивировать будильник на текущий день.
 */
private void z17_2() {
    editedActive[editedAlarmDay] = false;
}

/**
 * Увеличить корректирующее значение времени на 1 секунду.
 */
private void z19_1() {
    if (editCorrectValue < MAX_CORRECT_VALUE) {
        editCorrectValue++;
```

```
    }
}

/**
 * Уменьшить корректирующее значение времени на 1 секунду.
 */
private void z19_2() {
    if (editCorrectValue > -MAX_CORRECT_VALUE) {
        editCorrectValue--;
    }
}

/**
 * Вывести заряд батарей
 */
private void z21() {
    charge = (int) (batt.getCharge() * 100);
    batteryLevel.update(charge);
}

/**
 * Начать проигрывать звуковой сигнал
 */
private void z22_1() {
    player.startPlaying();
}

/**
 * Мигнуть светодиодами
 */
private void z22_2() {
    blinker.update();
}

/**
 * Выключить светодиоды и звуковой сигнал
 */
private void z22_3() {
    blinker.switchOff();
    player.stopPlaying();
}

/**
 * сбросить секундный таймер.
 */
private void z23() {
    secondTimer = System.currentTimeMillis();
}

private void z25() {
    inactivityTimer = System.currentTimeMillis();
}

private void z26() {
    batteryTimer = System.currentTimeMillis();
}


/**
 * проинициализировать жидкокристаллический дисплей,
 * секундный и побудочный таймеры, клавиатуру, часы реального времени.
 */
private void z40() {
```

```
    for (int i = 0; i < WEEK_DAYS; i++) {
        alarmActive[i] = false;
        alarmTime[i].hours = 10;
        alarmTime[i].minutes = 0;
        alarmTime[i].seconds = 0;
        alarmTime[i].day = i;
    }

    charge = (int) (batt.getCharge() * 100);
}

public void tick() {
    x1 = x1();
    x2 = x2();
    x3 = x3();
    x4 = x4();
    x6 = x6();
    x6_1 = x6_1();
    x6_2 = x6_2();
    x7_1 = x7_1();
    x8 = x8();
    x9 = x9();
    x10 = x10();
    a0();
}

private void a0() {
    switch (a0state) {
        case A0_INIT:
            z40();
            a0state = A0_NORMAL;
        break;

        case A0_NORMAL:
            a1();

            if (x1) {
                a0state = A0_ENERGY_SAVING;
            } else

            if (x3) {
                a0state = A0_ALARM_PLAYS;
            }
        break;

        case A0_ENERGY_SAVING:
            a1();

            if (!x1) {
                a0state = A0_NORMAL;
            }

        break;

        case A0_ALARM_PLAYS:
            a1();

            if (!x3) {
                z22_3();
                a0state = A0_NORMAL;
            } else

            if (x2 == STOP_ALARM_BTN)  {
```

```
                z22_3();
                a0state = A0_ALARM_OFF;
            } else {
                z22_1();
                z22_2();
            }
        break;

        case A0_ALARM_OFF:
            a1();

            if (!x3) {
                a0state = A0_NORMAL;
            }
        break;
    }
}

private void a1() {
    switch(a1state) {
        case A1_NORMAL:

            if (x2 == SET_CLOCK_BTN) {
                z2_1();
                z2();
                z9_2();
                z25();
                a1state = A1_TIME_EDIT;
            } else

            if ((x2 == SET_ALARM_BTN) && x7_1) {
                z3_1();
                z3();
                z9_1();
                z25();
                a1state = A1_ALARM_EDIT;
            } else

            if ((x2 == SET_ALARM_BTN) && !x7_1) {
                z3_1();
                z3();
                z9_2();
                z25();
                a1state = A1_ALARM_EDIT;
            } else

            if ((x2 == ADJ_P_BTN) || (x2 == ADJ_M_BTN)) {
                z5_1();
                z5();
                z9_2();
                z25();
                a1state = A1_CORRECT_EDIT;
            } else

            if (x10) {
                z21();
                z26();
            } else

            if (x4 && x8) {
                z1();
                z23();
                z9_1();
```

```
    } else

    if (x4 && !x8) {
        z1();
        z23();
        z9_2();
    } else {
        z0();
    }

break;

case A1_TIME_EDIT:

    if (x2 == OK_BTN) {
        z6();
        a1state = A1_TIME_UPDATE;
    } else

    if (x2 == CANCEL_BTN || x9) {
        a1state = A1_TIME_UPDATE;
    } else

    if (x2 == HOUR_P_BTN) {
        z10_1();
        z2();
        z25();
        z0();
    } else

    if (x2 == HOUR_M_BTN) {
        z10_2();
        z2();
        z25();
        z0();
    } else

    if (x2 == MIN_P_BTN) {
        z11_1();
        z2();
        z25();
        z0();
    } else

    if (x2 == MIN_M_BTN) {
        z11_2();
        z2();
        z25();
        z0();
    } else

    if (x2 == DAY_P_BTN) {
        z12_1();
        z2();
        z25();
        z0();
    } else

    if (x2 == DAY_M_BTN) {
        z12_2();
        z2();
        z25();
        z0();
```

```
    } else

    if (x10) {
        z21();
        z26();
    } else {
        z0();
    }


break;

case A1_ALARM_EDIT:
    if (x2 == OK_BTN) {
        z7();
        a1state = A1_TIME_UPDATE;
    } else

    if ((x2 == CANCEL_BTN) || x9) {
        a1state = A1_TIME_UPDATE;
    } else

    if (x2 == HOUR_P_BTN) {
        z15_1();
        z3();
        z25();
        z0();

    } else

    if (x2 == HOUR_M_BTN) {
        z15_2();
        z3();
        z25();
        z0();
    } else

    if (x2 == MIN_P_BTN) {
        z16_1();
        z3();
        z25();
        z0();
    } else

    if (x2 == MIN_M_BTN) {
        z16_2();
        z3();
        z25();
        z0();
    } else

    if ((x2 == DAY_P_BTN) && x6_1) {
        z14_1();
        z9_1();
        z3();
        z25();
        z0();
    } else

    if ((x2 == DAY_P_BTN) && !x6_1) {
        z14_1();
        z9_2();
        z3();
```

```
            z25();
            z0();
        } else

        if ((x2 == DAY_M_BTN) && x6_2) {
            z14_2();
            z9_1();
            z3();
            z25();
            z0();
        } else

        if ((x2 == DAY_M_BTN) && !x6_2) {
            z14_2();
            z9_2();
            z3();
            z25();
            z0();
        } else

        if ((x2 == ONOFF_BTN) && x6) {
            z17_2();
            z9_2();
            z25();
            z0();
        } else

        if ((x2 == ONOFF_BTN) && !x6) {
            z17_1();
            z9_1();
            z25();
            z0();
        } else

        if (x10) {
            z21();
            z26();
        } else {
            z0();
        }


    break;

    case A1_CORRECT_EDIT:
        if ((x2 == OK_BTN) && x8) {
            z8();
            z9_1();
            z0();
            z1();
            a1state = A1_NORMAL;
        } else

        if (((x2 == CANCEL_BTN) || x9) && x8) {
            z9_1();
            z0();
            z1();
            a1state = A1_NORMAL;
        } else

        if ((x2 == OK_BTN) && !x8) {
            z8();
            z9_2();
```

```
                    z0();
                    z1();
                    a1state = A1_NORMAL;
                } else

                if (((x2 == CANCEL_BTN) || x9) && !x8) {
                    z9_2();
                    z0();
                    z1();
                    a1state = A1_NORMAL;
                } else

                if (x2 == ADJ_P_BTN) {
                    z19_1();
                    z5();
                    z25();
                    z0();
                } else

                if (x2 == ADJ_M_BTN) {
                    z19_2();
                    z5();
                    z25();
                    z0();
                } else

                if (x10) {
                    z21();
                    z26();
                } else {
                    z0();
                }


            break;

            case A1_TIME_UPDATE:
                if (x8) {
                    z9_1();
                    z0();
                    z1();
                    a1state = A1_NORMAL;
                } else

                if (!x8) {
                    z9_2();
                    z0();
                    z1();
                    a1state = A1_NORMAL;
                }
            break;
        }
    }
}

public void autoTick() {
    auto.tick();
}

public class Battery {
    private double charge = 0.8;
    public static final double LOW_LIMIT = 0.2;
```

```
    public double getCharge() {
        return charge;
    }

    public void setCharge(double charge) {
        this.charge = charge;
    }
}

public class Time {
    private final String[] days = {
        "MON",
        "TUE",
        "WED",
        "THU",
        "FRI",
        "SAT",
        "SUN"
    };

    /**
     * from 0 to 23
     */
    public int hours;
    /**
     * from 0 to 59
     */
    public int minutes;
    /**
     * from zero to 59
     */
    public int seconds;
    /**
     * from 0 to 6
     */
    public int day;


    public void nextSec() {
        seconds++;
        if (seconds / 60 > 0) {
            minutes++;
            seconds = 0;
        }

        if (minutes / 60 > 0) {
            hours++;
            minutes = 0;
        }

        if (hours / 24 > 0) {
            day++;
            hours = 0;
        }

        if (day / 7 > 0) {
            day = 0;
        }
    }


    public String toWithoutSecondsTime() {
        String out = hours < 10 ? "0" : "";
```

```
            out += hours + ":";
            out += minutes < 10 ? "0" : "";
            out += minutes;
            out += "   ";
            out += " " + days[day];
            return out;
        }

        public String toString() {
            String out = hours < 10 ? "0" : "";
            out += hours + ":";
            out += minutes < 10 ? "0" : "";
            out += minutes + ":";
            out += seconds < 10 ? "0" : "";
            out += seconds;
            out += " " + days[day];

            return out;
        }
    }

    private class ClockPanel extends JComponent {
        private MediaTracker tracker;
        //the number of images in this object
        private int count = 10 + 7 + 4;

        private int space = 5;
        private Image [] numbers = {
            Toolkit.getDefaultToolkit().getImage("0.png"),
            Toolkit.getDefaultToolkit().getImage("1.png"),
            Toolkit.getDefaultToolkit().getImage("2.png"),
            Toolkit.getDefaultToolkit().getImage("3.png"),
            Toolkit.getDefaultToolkit().getImage("4.png"),
            Toolkit.getDefaultToolkit().getImage("5.png"),
            Toolkit.getDefaultToolkit().getImage("6.png"),
            Toolkit.getDefaultToolkit().getImage("7.png"),
            Toolkit.getDefaultToolkit().getImage("8.png"),
            Toolkit.getDefaultToolkit().getImage("9.png")
        };

        private Image [] weekDays = {
            Toolkit.getDefaultToolkit().getImage("mn.png"),
            Toolkit.getDefaultToolkit().getImage("tu.png"),
            Toolkit.getDefaultToolkit().getImage("wd.png"),
            Toolkit.getDefaultToolkit().getImage("th.png"),
            Toolkit.getDefaultToolkit().getImage("fr.png"),
            Toolkit.getDefaultToolkit().getImage("st.png"),
            Toolkit.getDefaultToolkit().getImage("sn.png")
        };

        private Image plus =
Toolkit.getDefaultToolkit().getImage("plus.png");
        private Image minus =
Toolkit.getDefaultToolkit().getImage("minus.png");
        private Image dots =
Toolkit.getDefaultToolkit().getImage("tchk.png");
        private Image alarm =
Toolkit.getDefaultToolkit().getImage("alarm.png");

        private Time time = null;
        private boolean alarmActive = false;

        private int state = 0;
```

```
public static final int NORMAL_STATE = 0;
public static final int WITHOUT_SECONDS_STATE = 1;
public static final int ADJ_STATE = 2;

public int value = 0;

public ClockPanel() {
    time = new Time();
    tracker = new MediaTracker(this);
    int id = 0;

    for (int i = 0; i < numbers.length; i++) {
        tracker.addImage(numbers[i], id);
        id++;
    }

    for (int i = 0; i < weekDays.length; i++) {
        tracker.addImage(weekDays[i], id);
        id++;
    }

    tracker.addImage(plus, id);
    id++;
    tracker.addImage(minus, id);
    id++;
    tracker.addImage(dots, id);
    id++;
    tracker.addImage(alarm, id);
    id++;

    setDoubleBuffered(true);
}

public Dimension getPreferredSize() {
    return new Dimension(357, 74);
}

public void setSize(int width, int height) {
    super.setSize(width, height);
}

public void setSize(Dimension d) {
    super.setSize(d);
}

public void updateTime(Time time) {
    state = NORMAL_STATE;
    this.time.hours = time.hours;
    this.time.minutes = time.minutes;
    this.time.seconds = time.seconds;
    this.time.day = time.day;

    SwingUtilities.invokeLater(new Runnable() {
        public void run() {
            ClockPanel.this.repaint();
        }
    });
}

public void updateTimeWithoutSeconds(Time time) {
    state = WITHOUT_SECONDS_STATE;
    this.time.hours = time.hours;
    this.time.minutes = time.minutes;
```

```
        this.time.seconds = time.seconds;
        this.time.day = time.day;

        SwingUtilities.invokeLater(new Runnable() {
            public void run() {
                ClockPanel.this.repaint();
            }
        });
    }

    public void updateCorrectTime(int value) {
        state = ADJ_STATE;
        this.value = value;
        SwingUtilities.invokeLater(new Runnable() {
            public void run() {
                ClockPanel.this.repaint();
            }
        });
    }

    public void updateAlarmActive(boolean alarmActive) {
        this.alarmActive = alarmActive;
        SwingUtilities.invokeLater(new Runnable() {
            public void run() {
                ClockPanel.this.repaint();
            }
        });
    }

    protected void paintComponent(Graphics g) {
        int done = 0;
        for (int i = 0; i < count; i++) {
            if (tracker.checkID(i, true)) {
                done++;
            }
        }

        if (done == count) {
            switch(state) {
                case NORMAL_STATE: {
                    int x = space;
                    int y = space;

                    g.drawImage(numbers[time.hours / 10], x, y, null);

                    x = x + space + numbers[0].getWidth(null);

                    g.drawImage(numbers[time.hours % 10], x, y, null);

                    x = x + space + numbers[0].getWidth(null);

                    g.drawImage(dots, x, y, null);

                    x = x + space + dots.getWidth(null);

                    g.drawImage(numbers[time.minutes / 10], x, y, null);

                    x = x + space + numbers[0].getWidth(null);

                    g.drawImage(numbers[time.minutes % 10], x, y, null);

                    x = x + space + numbers[0].getWidth(null);
```

```
            g.drawImage(dots, x, y, null);

            x = x + space + dots.getWidth(null);

            g.drawImage(numbers[time.seconds / 10], x, y, null);

            x = x + space + numbers[0].getWidth(null);

            g.drawImage(numbers[time.seconds % 10], x, y, null);

            x = x + space + numbers[0].getWidth(null);

            g.drawImage(weekDays[time.day], x, y, null);

            y = y + weekDays[time.day].getHeight(null) + space;

            if (alarmActive) {
                g.drawImage(alarm, x, y, null);
            }

            break;
        }

        case WITHOUT_SECONDS_STATE: {
            int x = space;
            int y = space;

            g.drawImage(numbers[time.hours / 10], x, y, null);

            x = x + space + numbers[0].getWidth(null);

            g.drawImage(numbers[time.hours % 10], x, y, null);

            x = x + space + numbers[0].getWidth(null);

            g.drawImage(dots, x, y, null);

            x = x + space + dots.getWidth(null);

            g.drawImage(numbers[time.minutes / 10], x, y, null);

            x = x + space + numbers[0].getWidth(null);

            g.drawImage(numbers[time.minutes % 10], x, y, null);

            x = x + space + numbers[0].getWidth(null);
            x = x + space + dots.getWidth(null);
            x = x + space + numbers[0].getWidth(null);
            x = x + space + numbers[0].getWidth(null);

            g.drawImage(weekDays[time.day], x, y, null);

            y = y + weekDays[time.day].getHeight(null) + space;

            if (alarmActive) {
                g.drawImage(alarm, x, y, null);
            }
            break;
        }
        case ADJ_STATE:
            int x = space;
            int y = space;
```

```
                        if (value > 0) {
                            g.drawImage(plus, x, y, null);
                            x = x + space + plus.getWidth(null);
                        } else
                        if (value < 0) {
                            g.drawImage(minus, x, y, null);
                            x = x + space + minus.getWidth(null);
                        }

                        g.drawImage(numbers[Math.abs(value / 10)], x, y,
null);

                        x = x + space + numbers[0].getWidth(null);

                        g.drawImage(numbers[Math.abs(value) % 10], x, y,
null);

                        x = x + space + numbers[0].getWidth(null);

                        break;
                    }
                } else {
                    g.drawString("Loading...", 10, 10);
                }
            }
        }

        private class BatteryView extends JComponent {
            private MediaTracker tracker;
            //the number of images in this object
            private int count = 5;


            private int value;
            private Image bat20 =
Toolkit.getDefaultToolkit().getImage("bat20.png");
            private Image bat40 =
Toolkit.getDefaultToolkit().getImage("bat40.png");
            private Image bat60 =
Toolkit.getDefaultToolkit().getImage("bat60.png");
            private Image bat80 =
Toolkit.getDefaultToolkit().getImage("bat80.png");
            private Image bat100 =
Toolkit.getDefaultToolkit().getImage("bat100.png");

            public BatteryView(int value) {
                this.value = value;
                tracker = new MediaTracker(this);

                tracker.addImage(bat20, 0);
                tracker.addImage(bat40, 1);
                tracker.addImage(bat60, 2);
                tracker.addImage(bat80, 3);
                tracker.addImage(bat100, 4);
            }



            public void update(int value) {
                this.value = value;
                SwingUtilities.invokeLater(new Runnable() {
                    public void run() {
                        BatteryView.this.repaint();
```

```
            }
        });
    }

    public Dimension getPreferredSize() {
        return new Dimension(125, 9);
    }
    public void paintComponent(Graphics g) {
        int done = 0;
        for (int i = 0; i < count; i++) {
            if (tracker.checkID(i, true)) {
                done++;
            }
        }

        if (done == count) {
            if (value < 20) {
                g.drawImage(bat20, 0, 0, null);
            } else
            if (value < 40) {
                g.drawImage(bat40, 0, 0, null);
            } else
            if (value < 60) {
                g.drawImage(bat60, 0, 0, null);
            } else
            if (value < 80) {
                g.drawImage(bat80, 0, 0, null);
            } else
            if (value <= 100) {
                g.drawImage(bat100, 0, 0, null);
            }
        } else {
            g.drawString("Loading...", 0, 0);
        }
    }
}

private class Blinker extends JComponent {
    private MediaTracker tracker;
    //the number of images in this object
    private int count = 5;

    private int imageShown = 0;
    private boolean upDirection = true;
    private static final int UPDATE_COUNT = 5;
    private int updateValue;

    private Image[] blinks = {
        Toolkit.getDefaultToolkit().getImage("blink1.png"),
        Toolkit.getDefaultToolkit().getImage("blink2.png"),
        Toolkit.getDefaultToolkit().getImage("blink3.png"),
        Toolkit.getDefaultToolkit().getImage("blink4.png"),
        Toolkit.getDefaultToolkit().getImage("blink5.png"),
    };

    public Blinker() {
        tracker = new MediaTracker(this);

        for (int i = 0; i < blinks.length; i++) {
            tracker.addImage(blinks[i], i);
        }
    }
```

```
    public Dimension getPreferredSize() {
        return new Dimension(28, 28);
    }

    public void update() {
        updateValue++;
        if (updateValue > UPDATE_COUNT) {
            updateValue = 0;
            if (upDirection) {
                if (imageShown == blinks.length - 1) {
                    imageShown = blinks.length - 2;
                    upDirection = false;
                } else{
                    imageShown++;
                }

            } else {
                imageShown--;
                if (imageShown == 0) {
                    imageShown = 1;
                    upDirection = true;
                } else {
                    imageShown--;
                }
            }
            SwingUtilities.invokeLater(new Runnable() {
                public void run() {
                    Blinker.this.repaint();
                }
            });
        }
    }

    public void switchOff() {
        imageShown = 0;
        updateValue = 0;
        upDirection = true;

        SwingUtilities.invokeLater(new Runnable() {
            public void run() {
                Blinker.this.repaint();
            }
        });
    }

    public void paintComponent(Graphics g) {
        int done = 0;
        for (int i = 0; i < count; i++) {
            if (tracker.checkID(i, true)) {
                done++;
            }
        }

        if (done == count) {
            g.drawImage(blinks[imageShown], 0, 0, null);
        } else {
            g.drawString("Loading...", 0, 0);
        }
    }
}

private class RTC {
    public long sec = System.currentTimeMillis() / 1000;
```

```
    public boolean secChanged() {
        long curTime = System.currentTimeMillis() / 1000;
        return (curTime != sec);
    }

    public void dropChanged() {
        sec = System.currentTimeMillis() / 1000;
    }
}

private class BellPlayer {
    public static final int PLAYING_STATE = 1;
    public static final int STOP_STATE = 3;
    public static final int ERROR_STATE = 5;
    private int playerState = STOP_STATE;

    private ContinuousAudioDataStream cas = null;
    public BellPlayer(String filename) {
        InputStream in = null;
        AudioStream as = null;
        AudioData data = null;
        try {
            in = new FileInputStream(filename);
            as = new AudioStream(in);
            data = as.getData();
        } catch (IOException ioe) {
            playerState = ERROR_STATE;
        }

        if (data != null) {
            cas = new ContinuousAudioDataStream(data);
        }
    }

    public void startPlaying() {
        if (playerState == STOP_STATE) {
            AudioPlayer.player.start(cas);
            playerState = PLAYING_STATE;
        }
    }

    public void stopPlaying() {
        if (playerState == PLAYING_STATE) {
            AudioPlayer.player.stop(cas);
            playerState = STOP_STATE;
        }
    }
}

public void actionPerformed(ActionEvent e) {
    if (e.getSource() == okBn) {
        pressedBtnInd = OK_BTN;
    }

    if (e.getSource() == cancelBn) {
        pressedBtnInd = CANCEL_BTN;
    }

    if (e.getSource() == onoffBn) {
        pressedBtnInd = ONOFF_BTN;
    }
```

```
        if (e.getSource() == setClockBn) {
            pressedBtnInd = SET_CLOCK_BTN;
        }

        if (e.getSource() == setAlarmBn) {
            pressedBtnInd = SET_ALARM_BTN;
        }

        if (e.getSource() == hourPlBn) {
            pressedBtnInd = HOUR_P_BTN;
        }

        if (e.getSource() == hourMinBn) {
            pressedBtnInd = HOUR_M_BTN;
        }

        if (e.getSource() == minPlBn) {
            pressedBtnInd = MIN_P_BTN;
        }

        if (e.getSource() == minMinBn) {
            pressedBtnInd = MIN_M_BTN;
        }

        if (e.getSource() == dayPlBn) {
            pressedBtnInd = DAY_P_BTN;
        }

        if (e.getSource() == dayMinBn) {
            pressedBtnInd = DAY_M_BTN;
        }

        if (e.getSource() == stopAlarmBn) {
            pressedBtnInd = STOP_ALARM_BTN;
        }

        if (e.getSource() == resetBn) {
            auto.restart();
        }

        if (e.getSource() == adjPlBn) {
            pressedBtnInd = ADJ_P_BTN;
        }

        if (e.getSource() == adjMinBn) {
            pressedBtnInd = ADJ_M_BTN;
        }

    }

    public static void main(String[] args) {
        ClockWindow clock = new ClockWindow();
        clock.pack();
        clock.setResizable(false);

        while(true) {
            try {
                Thread.sleep(10);
                clock.autoTick();
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
```

```
        }
    }
```

Приложение 2. Код программы для стенда *SDK 1.1*

```
#include "ADuC812.h"

//На микроконтроллере, на котором производилась
//отладка не вполне корректно работали часы реального времени.
//Поэтому код драйвера часов несколько отличается.
//Следующий макрос обозначает, что часы работают некорректно
//и драйвер следует компилировать немного иначе
#define FUCKED_RTC

typedef unsigned char uchar;
typedef unsigned short ushort;
typedef unsigned int uint;
typedef unsigned long ulong;

//Структура для хранения часов, минут, секунд, дней и факта включенности
//будильника на день
typedef struct {
      uchar hours;
      uchar minutes;
      uchar seconds;
      uchar days;
      uchar active;
} Time;

uchar weekdays[7][4] = {
      "MON", "TUE", "WED", "THU", "FRI", "SAT", "SUN"
};

//Текущее значение времени
Time currentTime;

//Значение редактируемого времени
Time editedTime;

//Значение редактируемого времени будильника
Time editedAlarmTime[7];

//Значение дня, на который редактируется будильник
uint editedDay;

//Значение времени и включенности будильника
Time alarmTime[7];

//Максимальное значение корректировки времени
#define MAX_CORRECT 30
//Минимальное значение корректировки времени
#define MIN_CORRECT -30

//Редактируемое значение корректировки
int editedCorrectValue;


//Текущее значение корректировки
int correctValue;


//Значение секундного таймер
ulong secondTimer;
```

```
//Значение таймера бездействия
ulong inactivityTimer;

//Значение таймера батареи
ulong batteryTimer;

//Текущее значение времени
ulong timerValue;
```

```
//---------------------------------
// Драйвер интерфейса I2C
//---------------------------------

void Delay(void)
{
      char ch = 0;
    while (ch++ < 2);
}

bit SendByte(uchar ch) {
      char cnt;
      bit ack;

    MDE = 1;
    MCO = 0;

    for(cnt = 0; cnt < 8; cnt++, ch <<= 1) {
        MDO = (ch & 0x80) ? 1 : 0;
        MCO = 1;
        Delay();
        MCO = 0;
    }

    MDE = 0;
    MCO = 1;
    Delay();
    ack = MDI;
    MCO = 0;
    return ack;
}

void Start(void) {
    MDE = 1;
    MDO = 1;
    MCO = 1;
    Delay();
    MDO = 0;
    Delay();
    MCO = 0;
}

void Stop(void) {
    MDE = 1;
    MCO = 0;
    MDO = 0;
    MCO = 1;
    Delay();
    MDO = 1;
    Delay();
    MDE = 0;
}


bit Begin(unsigned char addr) {
    Start();
    return SendByte(addr);
}

void Ack(void) {
    MDE = 1;
    MCO = 0;
    MDO = 0;
```

```
    MCO = 1;
    Delay();
    MCO = 0;
}

void Nack(void) {
    MDE = 1;
    MCO = 0;
    MDO = 1;
    MCO = 1;
    Delay();
    MCO = 0;
}

bit GetAck(uchar address) {
    I2CM = 1;

    if (Begin(address & 0xFE)){
      Stop();
      return 0;
    }

    Stop();

    return 1;
}


uchar RecvByte(void) {
      char cnt;
      uchar ch = 0;

    MDE = 0;
    MCO = 0;

    for(cnt = 0; cnt < 8; cnt++) {
        ch <<= 1;
        MCO = 1;
        Delay();
        ch |= MDI;
        MCO = 0;
    }
    return ch;
}

bit ReceiveBlock(uchar address, uchar addr, uchar xdata * block, uchar len) {
      uchar l, ch;

    I2CM = 1;
    address = (address & 0xFE);

    if (Begin(address)) {
      Stop();
      return 1;
    }

    if (SendByte(addr)) {
      Stop();
      return 1;
    }

    Delay();
    Delay();
```

```
    address = (address | 1);

    if(Begin(address)) {
      Stop();
      return 1;
    }

    Delay();

    if(len - 1) {
        for(l = 0; l < (len-1); l++) {
            ch = RecvByte();
            Ack();
            *block++ = ch;
        }
    }

    ch = RecvByte();
    Nack();
    *block = ch;
    Stop();
    return 0;
}

bit TransmitBlock(uchar address, uchar addr, uchar xdata * block, uchar len)
{
      uchar ch,l;

    I2CM = 1;
    address = (address & 0xFE);

    if (Begin(address)) {
      Stop();
      return 1;
      }

    if (SendByte(addr)) {
      Stop();
      return 1;
    }

    for(l = 0; l < len; l++, block++) {
        ch = *block;
        if (SendByte(ch)) {
            Stop();
            return 1;
        }
    }

    Stop();
    return 0;
}

//---------------------------------
// Конец драйвера I2C
//---------------------------------
```

```
//---------------------------------
// Драйвер часов реального времени
//---------------------------------

#define RTC_ADDRESS     0xA0
#define BCD2Bin(val)    (((val) & 0xF) + ((((val) & 0xF0) >> 4) * 10))
#define Bin2BCD(val)    (((val) % 10) + (((val) / 10) << 4))


typedef struct {
    uchar hsec;
    uchar sec;
    uchar min;
    uchar hour;
} RTCTIME;

bit SetMode(uchar xdata * mode) {
    if( !GetAck(RTC_ADDRESS) )
        return 1;

    if( TransmitBlock(RTC_ADDRESS, 0, mode, 1) )
        return 1;

    return 0;
}

//Часы работают некорректно
#ifdef FUCKED_RTC

uchar fixedday;
uchar fixedsec;
uchar fixedmin;
uchar fixedhour;

uchar oldsec = 0;

bit UpdateTime() {
      uchar newsec;
    RTCTIME t;

    if (!GetAck(RTC_ADDRESS))
        return 1;

    if (ReceiveBlock(RTC_ADDRESS, 1, (uchar xdata*) &t, 4))
        return 1;

    newsec = BCD2Bin(t.sec);

      if (oldsec != newsec) {
            fixedsec++;
            if (fixedsec >= 60) {
                  fixedmin++;
                  fixedsec = 0;
            }

            if (fixedmin >= 60) {
                  fixedhour++;
                  fixedmin = 0;
            }

            if (fixedhour >= 24) {
                  fixedday++;
                  fixedhour = 0;
```

```
            }

            if (fixedday >= 7) {
                  fixedday = 0;
            }
      }
      oldsec = newsec;
      return 0;
}

bit GetTime(RTCTIME xdata * time) {
      time->sec = fixedsec;
      time->min = fixedmin;
      time->hour = fixedhour;
      return 0;
}

bit SetTime(RTCTIME xdata* time) {
      fixedsec = time->sec;
      fixedmin = time->min;
      fixedhour = time->hour;
      return 0;
}

bit GetDay(uchar xdata* day) {
      *day = fixedday;
      return 0;
}

bit SetDay(uchar xdata* day) {
    fixedday = *day;
      return 0;
}


#elif

//Часы работают корректно

bit GetTime(RTCTIME xdata * time) {
      uchar h;

    if (!GetAck(RTC_ADDRESS))
        return 1;

    if (ReceiveBlock(RTC_ADDRESS, 1, (uchar xdata*) time, 4))
        return 1;

    time->hsec = BCD2Bin(time->hsec);
    time->sec  = BCD2Bin(time->sec);
    time->min  = BCD2Bin(time->min);
    h          = BCD2Bin(time->hour & 0x3F);

    if( time->hour & 0xC0 ) {
        if(time->hour < 12) {
            time->hour = h + 12;
        } else {
            time->hour = 0;
        }
    } else {
        time->hour = h;
    }
```

```
    return 0;
}

bit SetTime(RTCTIME xdata* time) {
      uchar h;
      RTCTIME t;

    if(!GetAck(RTC_ADDRESS)) {
        return 1;
    }


    t.hsec = Bin2BCD(time->hsec);
    t.sec  = Bin2BCD(time->sec);
    t.min  = Bin2BCD(time->min);
    t.hour = Bin2BCD(time->hour & 0x3F);

    if( TransmitBlock(RTC_ADDRESS, 1, (uchar xdata*) &t, 4) )
        return 1;

    return 0;
}

bit GetDay(uchar xdata* day) {
    if (!GetAck(RTC_ADDRESS))
        return 1;

    if (ReceiveBlock(RTC_ADDRESS, 6, (uchar xdata*) day, 1))
        return 1;

    (*day) = (*day) >> 5;
      return 0;
}

bit SetDay(uchar xdata* day) {
    if(!GetAck(RTC_ADDRESS)) {
        return 1;
    }

    (*day) = (*day) << 5;

    if( TransmitBlock(RTC_ADDRESS, 6, (uchar xdata*) day, 1) )
        return 1;

    return 0;

}



#endif


//---------------------------------
// Конец драйвера часов
//---------------------------------
```

```
//---------------------------------
// Драйвер интерфейса MAX
//---------------------------------

//Регистр клавиатуры
#define KB          0x0

//Регистр данных ЖК-дисплея
#define DATA_IND  0x1

//Управляющий регистр ЖК-дисплея
#define C_IND           0x6

//Адрес в памяти, куда подключен интерфейс MAX
#define MAXBASE 0x8


//Регистр динамика
#define ENA 0x4

//Регистр светодиодов
#define LEDLINE 0x7

//Запись значения по регистру
void WriteMax(uchar xdata *regnum, uchar val) {
      uchar oldDPP=DPP;
      DPP=MAXBASE;
      *regnum=val;
      DPP=oldDPP;
}

//Чтение значения из регистра
uchar ReadMax(uchar xdata *regnum) {
      uchar oldDPP=DPP;
      uchar val=0;

      DPP=MAXBASE;
      val=*regnum;
      DPP=oldDPP;
      return val;
}

//---------------------------------
// Конец драйвера MAX
//---------------------------------
```

```
//---------------------------------
// Дравер клавиатуры
//---------------------------------

#define OK_BTN 10
#define CANCEL_BTN 14
#define ONOFF_BTN 15
#define SET_CLOCK_BTN 2
#define SET_ALARM_BTN 6

#define HOUR_P_BTN 0
#define HOUR_M_BTN 1
#define MIN_P_BTN 4
#define MIN_M_BTN 5
#define DAY_P_BTN 8
#define DAY_M_BTN 9

#define STOP_ALARM_BTN 3

#define ADJ_P_BTN 12
#define ADJ_M_BTN 13



uchar lastScanned = 100;

bit ScanKB(uchar *ch) {
      uchar row, col, rownum, colnum, button;
      uint i;

    for(colnum = 0; colnum < 4; colnum++) {

        col = 0x1 << colnum;

        WriteMax(KB, ~col);

        for(rownum = 0; rownum < 4; rownum++) {
            row = ReadMax(KB) & (0x10 << rownum);

            if(!row) {
                for(i = 0; i < 10000; i++) continue;

                row = ReadMax(KB) & (0x10 << rownum);

                if(!row) {

                    button = (uchar) ((colnum << 2) + rownum);

                    if (button != lastScanned) {
                        *ch = button;
                        lastScanned = button;
                        return 1;
                    } else {
                        return 0;
                    }
                }
            }
        }
    }

    lastScanned = 100;
    return 0;
}
```

```
//---------------------------------
// Конец драйвера клавиатуры
//---------------------------------
```

```
//---------------------------------
// Драйвер ЖК-дисплея
//---------------------------------


#define CLEAR           0x01
#define HOME            0x02
#define ENTRY_MODE      0x04
#define DISPLAY_CTRL    0x08
#define SHIFT           0x10
#define FUNCTION_SET    0x20
#define RAM_CG          0x40
#define RAM_DD          0x80
#define BF_AC           0x80

#define INCR            0x02
#define DISPLAY_SHIFT   0x01
#define DISPLAY_ON      0x04
#define CURSOR_ON       0x02
#define BLINK           0x01
#define DISPLAY         0x08
#define RIGHT           0x04
#define EIGHT_BITS      0x10
#define TWO_LINE        0x08


void Strobe(char c) {
      uint i;

    WriteMax(C_IND,c | 0x1);
    WriteMax(C_IND,c & 0xFE);

    for (i = 0; i < 300; i++) continue;
}


void InitializeLCD() {
      uint i;

      for(i = 0; i < 4000; i++) continue;

    WriteMax(DATA_IND, FUNCTION_SET|EIGHT_BITS);
    Strobe(0x8);
    for(i = 0; i < 1500; i++) continue;

    WriteMax(DATA_IND, FUNCTION_SET|EIGHT_BITS);
    Strobe(0x8);
    for(i = 0; i < 50; i++) continue;

    WriteMax(DATA_IND, FUNCTION_SET|EIGHT_BITS);
    Strobe(0x8);

    WriteMax(DATA_IND, FUNCTION_SET|EIGHT_BITS|TWO_LINE);
    Strobe(0x8);

    WriteMax(DATA_IND, DISPLAY_CTRL);
    Strobe(0x8);

    WriteMax(DATA_IND, CLEAR);
    Strobe(0x8);

    WriteMax(DATA_IND, ENTRY_MODE | INCR);
    Strobe(0x8);
```

```
    WriteMax(DATA_IND, DISPLAY_CTRL | DISPLAY_ON);
    Strobe(0x8);

}

void SetCharAt(uint x, uint y, uchar value) {
    WriteMax(DATA_IND, RAM_DD | (x + 0x40 * y));
    Strobe(0x8);

    WriteMax(DATA_IND, value);
    Strobe(0xC);

}

//---------------------------------
// Конец драйвера ЖК-дисплея
//---------------------------------
```

```
//---------------------------------
// Входные переменные
//---------------------------------

uint x1() {
      return 0;
}

uint x2() {
      uchar c;

      if (ScanKB(&c)) {
            return c;
      }

      return -1;
}


uint x3() {
      uchar d = currentTime.days;
      uchar h = currentTime.hours;
      uchar m = currentTime.minutes;
      uchar ah = alarmTime[d].hours;
      uchar am = alarmTime[d].minutes;

      if (!alarmTime[d].active) {
            return 0;
      }

      return ((h == ah) && (m == am));
}

uint x4() {
      if (timerValue - secondTimer >= 1) {
            return 1;
      }

      return 0;
}

uint x6() {
      return (editedAlarmTime[editedDay].active);
}


uint x6_1() {
      uint day = editedDay + 1;
      uint active;

      if (day > 6) {
            day = 0;
      }

      active = (uint) editedAlarmTime[day].active;
      return active;
}

uint x6_2() {
      uint day;
      uint active;

      if (editedDay == 0) {
```

```
            day = 6;
      } else {
            day = editedDay - 1;
      }

      active = (uint) editedAlarmTime[day].active;
      return active;
}

uint x7_1() {
      uchar d = currentTime.days;
      d++;
      if (d > 6) {
            d = 0;
      }

      return (alarmTime[d].active);
}

uint x8() {
      uchar d = currentTime.days;
      uchar h = currentTime.hours;
      uchar m = currentTime.minutes;

      uchar ah = alarmTime[d].hours;
      uchar am = alarmTime[d].minutes;
      uchar act = alarmTime[d].active;

      if ((h < ah) || ((h == ah) && (m < am))) {
            return act;
      } else {
            d++;

            if (d > 6) {
                  d = 0;
            }

            return alarmTime[d].active;
      }
}


uint x9() {
      if (timerValue - inactivityTimer >= 20) {
            return 1;
      }

      return 0;
}

uint x10() {
      if (timerValue - batteryTimer >= 30) {
            return 1;
      }

      return 0;
}

//---------------------------------
// Конец блока входных переменных
//---------------------------------
```

```
//---------------------------------
// Выходные воздействия
//---------------------------------


void z0() {
    RTCTIME time;
      uchar day;

      #ifdef FUCKED_RTC
      UpdateTime();
      #endif

      GetTime(&time);
      GetDay(&day);

      if (currentTime.seconds != time.sec) {
            timerValue++;
      }

      currentTime.hours = time.hour;
      currentTime.minutes = time.min;
      currentTime.seconds = time.sec;
      currentTime.days = day;
}

void z1() {

      uchar h = currentTime.hours;
      uchar m = currentTime.minutes;
      uchar s = currentTime.seconds;
      uchar d = currentTime.days;

      SetCharAt(0, 0, h / 10 + '0');
      SetCharAt(1, 0, h % 10 + '0');
      SetCharAt(2, 0, ':');

      SetCharAt(3, 0, m / 10 + '0');
      SetCharAt(4, 0, m % 10 + '0');
      SetCharAt(5, 0, ':');

      SetCharAt(6, 0, s / 10 + '0');
      SetCharAt(7, 0, s % 10 + '0');

      SetCharAt(9, 0, weekdays[d][0]);
      SetCharAt(10, 0, weekdays[d][1]);
      SetCharAt(11, 0, weekdays[d][2]);
}


void z2() {
      uchar h = editedTime.hours;
      uchar m = editedTime.minutes;
      uchar s = editedTime.seconds;
      uchar d = editedTime.days;

      SetCharAt(0, 0, h / 10 + '0');
      SetCharAt(1, 0, h % 10 + '0');
      SetCharAt(2, 0, ':');

      SetCharAt(3, 0, m / 10 + '0');
      SetCharAt(4, 0, m % 10 + '0');
      SetCharAt(5, 0, ' ');
```

```
	SetCharAt(6, 0, ' ');
	SetCharAt(7, 0, ' ');

	SetCharAt(9, 0, weekdays[d][0]);
	SetCharAt(10, 0, weekdays[d][1]);
	SetCharAt(11, 0, weekdays[d][2]);

}

void z3() {
	uchar d = editedDay;
	uchar h = editedAlarmTime[d].hours;
	uchar m = editedAlarmTime[d].minutes;

	SetCharAt(0, 0, h / 10 + '0');
	SetCharAt(1, 0, h % 10 + '0');
	SetCharAt(2, 0, ':');

	SetCharAt(3, 0, m / 10 + '0');
	SetCharAt(4, 0, m % 10 + '0');
	SetCharAt(5, 0, ' ');

	SetCharAt(6, 0, ' ');
	SetCharAt(7, 0, ' ');

	SetCharAt(9, 0, weekdays[d][0]);
	SetCharAt(10, 0, weekdays[d][1]);
	SetCharAt(11, 0, weekdays[d][2]);

}

void z5() {
	int ec = editedCorrectValue;

	SetCharAt(0, 0, ' ');
	SetCharAt(1, 0, ' ');
	SetCharAt(2, 0, ' ');
	SetCharAt(3, 0, ' ');
	SetCharAt(4, 0, ' ');

	if (ec > 0) {
		SetCharAt(5, 0, '+');
	} else

	if (ec < 0) {
		SetCharAt(5, 0, '-');
	} else {
		SetCharAt(5, 0, ' ');
	}

	ec = (ec > 0) ? ec : -ec;
	SetCharAt(6, 0, (ec / 10) + '0');
	SetCharAt(7, 0, (ec % 10) + '0');

	SetCharAt( 9, 0, ' ');
	SetCharAt(10, 0, ' ');
	SetCharAt(11, 0, ' ');
}

void z2_1() {
	editedTime.hours = currentTime.hours;
	editedTime.minutes = currentTime.minutes;
```

```
	editedTime.seconds = currentTime.seconds;
	editedTime.days = currentTime.days;
}

void z3_1() {
	uint i;
	editedDay = currentTime.days;
	editedDay++;
	if (editedDay > 6) {
		editedDay = 0;
	}

	for (i = 0; i < 7; i++) {
		editedAlarmTime[i].hours = alarmTime[i].hours;
		editedAlarmTime[i].minutes = alarmTime[i].minutes;
		editedAlarmTime[i].active = alarmTime[i].active;
	}
}

void z5_1() {
	editedCorrectValue = correctValue;
}

void z6() {
	RTCTIME time;


	currentTime.hours = editedTime.hours;
	currentTime.minutes = editedTime.minutes;
	currentTime.seconds = 0;
	currentTime.days = editedTime.days;

	time.hour = editedTime.hours;
	time.min = editedTime.minutes;
	time.sec = 0;

	SetTime(&time);
	SetDay(&(currentTime.days));
}

void z7() {
	uint i;

	for (i = 0; i < 7; i++) {
		alarmTime[i].hours = editedAlarmTime[i].hours;
		alarmTime[i].minutes = editedAlarmTime[i].minutes;
		alarmTime[i].active = editedAlarmTime[i].active;
	}
}

void z8() {
	correctValue = editedCorrectValue;
}

void z9_1() {
	SetCharAt(0, 1, 'O');
	SetCharAt(1, 1, 'N');
	SetCharAt(2, 1, ' ');
}

void z9_2() {
	SetCharAt(0, 1, 'O');
	SetCharAt(1, 1, 'F');
```

```
	SetCharAt(2, 1, 'F');
}

void z10_1() {
	editedTime.hours++;
	if (editedTime.hours > 23) {
		editedTime.hours = 0;
	}
}

void z10_2() {
	if (editedTime.hours == 0) {
		editedTime.hours = 23;
	} else {
		editedTime.hours--;
	}
}

void z11_1() {
	editedTime.minutes++;
	if (editedTime.minutes > 59) {
		editedTime.minutes = 0;
	}
}

void z11_2() {
	if (editedTime.minutes == 0) {
		editedTime.minutes = 59;
	} else {
		editedTime.minutes--;
	}
}

void z12_1() {
	editedTime.days++;
	if (editedTime.days > 6) {
		editedTime.days = 0;
	}
}

void z12_2() {
	if (editedTime.days == 0) {
		editedTime.days = 6;
	} else {
		editedTime.days--;
	}
}

void z14_1() {
	editedDay++;
	if (editedDay > 6) {
		editedDay = 0;
	}
}

void z14_2() {
	if (editedDay == 0) {
		editedDay = 6;
	} else {
		editedDay--;
	}
}
```

```
void z15_1() {
	editedAlarmTime[editedDay].hours++;
	if (editedAlarmTime[editedDay].hours > 23) {
		editedAlarmTime[editedDay].hours = 0;
	}
}

void z15_2() {
	if (editedAlarmTime[editedDay].hours == 0) {
		editedAlarmTime[editedDay].hours = 23;
	} else {
		editedAlarmTime[editedDay].hours--;
	}
}

void z16_1() {
	editedAlarmTime[editedDay].minutes++;
	if (editedAlarmTime[editedDay].minutes > 59) {
		editedAlarmTime[editedDay].minutes = 0;
	}
}

void z16_2() {
	if (editedAlarmTime[editedDay].minutes == 0) {
		editedAlarmTime[editedDay].minutes = 59;
	} else {
		editedAlarmTime[editedDay].minutes--;
	}
}

void z17_1() {
	editedAlarmTime[editedDay].active = 1;
}

void z17_2() {
	editedAlarmTime[editedDay].active = 0;
}

void z19_1() {
	editedCorrectValue++;
	if (editedCorrectValue > MAX_CORRECT) {
		editedCorrectValue = MAX_CORRECT;
	}
}

void z19_2() {
	editedCorrectValue--;
	if (editedCorrectValue < MIN_CORRECT) {
		editedCorrectValue = MIN_CORRECT;
	}
}

void z21() {
}


void z23() {
	secondTimer = timerValue;
}

void z25() {
	inactivityTimer = timerValue;
}
```

```
void z26() {
      batteryTimer = timerValue;
}

void z22_1() {
      uint i;

      for(i = 0; i < 30; i++) {
            uchar c;
            WriteMax(ENA, 24);
            for (c = 0; c < 20; c++);
            WriteMax(ENA, 4);
      }
}

uchar ledsection = 15;
void z22_2() {
      if (ledsection & 128) {
            ledsection = ledsection << 1;
            ledsection += 1;
      } else {
            ledsection = ledsection << 1;
      }
      WriteMax(LEDLINE, ledsection);
}

void z22_3() {
      WriteMax(LEDLINE, 0);
}

void z40() {
      RTCTIME time;
      uint i;
      uchar mode = 0;
      InitializeLCD();

      time.hour = currentTime.hours = 0;
      time.min = currentTime.minutes = 0;
      time.sec = currentTime.seconds = 0;
      currentTime.days = 0;

      SetTime(&time);
      SetDay(&(currentTime.days));

      SetMode(&mode);

      for(i = 0; i < 7; i++) {
            alarmTime[i].hours = 10;
            alarmTime[i].minutes = 0;
            alarmTime[i].seconds = 0;
            alarmTime[i].active = 0;
      }

      correctValue = 0;

      timerValue = 0;

      secondTimer = 0;
      inactivityTimer = 0;
      batteryTimer = 0;

}
```

```
uint _x1;
uint _x2;
uint _x3;
uint _x4;
uint _x6;
uint _x6_1;
uint _x6_2;
uint _x7_1;
uint _x8;
uint _x9;
uint _x10;


void UpdateVariables() {
	_x1 = x1();
	_x2 = x2();
	_x3 = x3();
	_x4 = x4();
	_x6 = x6();
	_x6_1 = x6_1();
	_x6_2 = x6_2();
	_x7_1 = x7_1();
	_x8 = x8();
	_x9 = x9();
	_x10 = x10();
}

void log(uint at, uint state) {
	if (at == 0) {
		SetCharAt(8, 1, state + '0');
	} else {
		SetCharAt(10, 1, state + '0');
	}
}



#define A1_NORMAL 1
#define A1_TIME_EDIT 2
#define A1_ALARM_EDIT 3
#define A1_CORRECT_EDIT 4
#define A1_TIME_UPDATE 5

uint y1 = A1_NORMAL;

void A1() {

	log(1, y1);
	switch(y1) {
		case A1_NORMAL:

			if (_x2 == SET_CLOCK_BTN) {
				z2_1();
				z2();
				z9_2();
				z25();
				y1 = A1_TIME_EDIT;
			} else

			if ((_x2 == SET_ALARM_BTN) && _x7_1) {
				z3_1();
				z3();
```

```
            z9_1();
            z25();
            y1 = A1_ALARM_EDIT;
      } else

      if ((_x2 == SET_ALARM_BTN) && !_x7_1) {
            z3_1();
            z3();
            z9_2();
            z25();
            y1 = A1_ALARM_EDIT;
      } else

      if ((_x2 == ADJ_P_BTN) || (_x2 == ADJ_M_BTN)) {
            z5_1();
            z5();
            z9_2();
            z25();
            y1 = A1_CORRECT_EDIT;
      } else

      if (_x10) {
            z21();
            z26();
      } else

      if (_x4 && _x8) {
            z1();
            z23();
            z9_1();
      } else

      if (_x4 && !_x8) {
            z1();
            z23();
            z9_2();
      } else {
            z0();
      }

break;

case A1_TIME_EDIT:

      if (_x2 == OK_BTN) {
            z6();
            y1 = A1_TIME_UPDATE;
      } else

      if (_x2 == CANCEL_BTN || _x9) {
            y1 = A1_TIME_UPDATE;
      } else

      if (_x2 == HOUR_P_BTN) {
            z10_1();
            z2();
            z25();
            z0();
      } else

      if (_x2 == HOUR_M_BTN) {
            z10_2();
            z2();
```

```
            z25();
            z0();
      } else

      if (_x2 == MIN_P_BTN) {
            z11_1();
            z2();
            z25();
            z0();
      } else

      if (_x2 == MIN_M_BTN) {
            z11_2();
            z2();
            z25();
            z0();
      } else

      if (_x2 == DAY_P_BTN) {
            z12_1();
            z2();
            z25();
            z0();
      } else

      if (_x2 == DAY_M_BTN) {
            z12_2();
            z2();
            z25();
            z0();
      } else

      if (_x10) {
            z21();
            z26();
      } else {
            z0();
      }


break;

case A1_ALARM_EDIT:
      if (_x2 == OK_BTN) {
            z7();
            y1 = A1_TIME_UPDATE;
      } else

      if ((_x2 == CANCEL_BTN) || _x9) {
            y1 = A1_TIME_UPDATE;
      } else

      if (_x2 == HOUR_P_BTN) {
            z15_1();
            z3();
            z25();
            z0();

      } else

      if (_x2 == HOUR_M_BTN) {
            z15_2();
            z3();
```

```
      z25();
      z0();
} else

if (_x2 == MIN_P_BTN) {
      z16_1();
      z3();
      z25();
      z0();
} else

if (_x2 == MIN_M_BTN) {
      z16_2();
      z3();
      z25();
      z0();
} else

if ((_x2 == DAY_P_BTN) && _x6_1) {
      z14_1();
      z9_1();
      z3();
      z25();
      z0();
} else

if ((_x2 == DAY_P_BTN) && !_x6_1) {
      z14_1();
      z9_2();
      z3();
      z25();
      z0();
} else

if ((_x2 == DAY_M_BTN) && _x6_2) {
      z14_2();
      z9_1();
      z3();
      z25();
      z0();
} else

if ((_x2 == DAY_M_BTN) && !_x6_2) {
      z14_2();
      z9_2();
      z3();
      z25();
      z0();
} else

if ((_x2 == ONOFF_BTN) && _x6) {
      z17_2();
      z9_2();
      z25();
      z0();
} else

if ((_x2 == ONOFF_BTN) && !_x6) {
      z17_1();
      z9_1();
      z25();
      z0();
} else
```

```
        if (_x10) {
              z21();
              z26();
        } else {
              z0();
        }


break;

case A1_CORRECT_EDIT:
        if ((_x2 == OK_BTN) && _x8) {
              z8();
              z9_1();
              z0();
              z1();
              y1 = A1_NORMAL;
        } else

        if (((_x2 == CANCEL_BTN) || _x9) && _x8) {
              z9_1();
              z0();
              z1();
              y1 = A1_NORMAL;
        } else

        if ((_x2 == OK_BTN) && !_x8) {
              z8();
              z9_2();
              z0();
              z1();
              y1 = A1_NORMAL;
        } else

        if (((_x2 == CANCEL_BTN) || _x9) && !_x8) {
              z9_2();
              z0();
              z1();
              y1 = A1_NORMAL;
        } else

        if (_x2 == ADJ_P_BTN) {
              z19_1();
              z5();
              z25();
              z0();
        } else

        if (_x2 == ADJ_M_BTN) {
              z19_2();
              z5();
              z25();
              z0();
        } else

        if (_x10) {
              z21();
              z26();
        } else {
              z0();
        }
```

```
            break;

            case A1_TIME_UPDATE:
              if (_x8) {
                    z9_1();
                    z0();
                    z1();
                    y1 = A1_NORMAL;
              } else

              if (!_x8) {
                    z9_2();
                    z0();
                    z1();
                    y1 = A1_NORMAL;
              }
            break;

      }
}

#define A0_INIT 1
#define A0_NORMAL 2
#define A0_ENERGY_SAVING 3
#define A0_ALARM_PLAYS 4
#define A0_ALARM_OFF 5

uint y0 = A0_INIT;


void A0() {
      log(0, y0);
      switch (y0) {

            case A0_INIT:
                  z40();
                  y0 = A0_NORMAL;
            break;

            case A0_NORMAL:
                  A1();

                  if (_x1) {
                        y0 = A0_ENERGY_SAVING;
                  } else

                  if (_x3) {
                        y0 = A0_ALARM_PLAYS;
                  }
            break;

            case A0_ENERGY_SAVING:
                  A1();

                  if (!x1) {
                        y0 = A0_NORMAL;
                  }

            break;

            case A0_ALARM_PLAYS:
                  A1();
```

```
                  if (!_x3) {
                        z22_3();
                        y0 = A0_NORMAL;
                  } else

                  if (_x2 == STOP_ALARM_BTN)  {
                        z22_3();
                        y0 = A0_ALARM_OFF;
                  } else {
                        z22_1();
                        z22_2();
                  }
            break;

            case A0_ALARM_OFF:
                  A1();

                  if (!_x3) {
                        y0 = A0_NORMAL;
                  }
            break;
      }
}



void main(void) {
      while (1) {
            UpdateVariables();
            A0();
      }
}
```