**Вступление**

Были разработаны 5 функций:

updateTime;

show;

timeShift;

updateMode;

dateMode.

Таблица глобальных переменных

|  |  |
| --- | --- |
| Имена переменных | Семантика переменных |
| out1 – out8 | Пины дешифратора |
| key1 – key6 | Пины ламп |
| lampKeys | Массив номеров пинов ламп |
| digits | Массив для текущего значения времени на шесть цифр |
| mainMode | True - режим часов, false - режим календаря |
| AntiRattlingTimer | Таймер для устранения дребезга кнопки |
| pushedKey | Нажатая кнопка |
| blinkTimer | Массив времени начала мигания для каждой цифры |
| flagBlink | Индикатор мигания для каждой цифры |
| previousNumber | Предыдущее значение для каждой цифры |
| delaysArr | Массив для мигания индикаторов (хранит время задержки на пол-секунды мигания) |

**Описание функций**

Таблица функций

|  |  |
| --- | --- |
| Идентификатор | Семантика |
| updateTime | Получает значения времени из модуля реального времени, записывает значения в глобальные переменные *second, minute, hour,* рассчитывает значение для каждого индикатора и сохраняет их в соответствующие глобальные массивы *previousNumber* и *digits.* |
| show | Используется для отображения времени, а также обеспечивает мигание индикаторами путем выставления задержек свечения индикатора. Перебирает в цикле все индикаторы и выставляет значения при помощи функции *setNumber,* миганиями управляет при помощи глобальных массивов *delaysArr, flagBlink* а так же функции *delayMicroseconds.* Индикатор включается при подаче высокого уровня сигнала на соответствующий пин и выключается после подачи низкого уровня на тот же пин. Значения для выставления на индикатор берутся из массивов *previousNumber*(при мигании индикатора) и *digits*(в противном случае). |
| timeShift | Используется для обновления значений с модуля реального времени, записи обновленных данных в переменные *second, minute, hour,* массивы *previousNumber, digits,* а также устанавливаются мигания для индикаторов, с измененными значениями с помощью массивов *blinkTimer* и *flagBlink* |
| updateMode | Обновляет день, месяц, год с модуля реального времени и записывает в массив *digits.* |
| dateMode | Используется для отображения даты. Вызывает функции *updateDate, show, updateDate, navigate.* |

**Алгоритм функций**

## Функция “show”

1. Обрабатывается следующий индикатор (от первого до шестого)
   1. Если индикатор находится в состоянии мигания
      1. Рассчитывается время, прошедшее с начала текущего мигания
      2. Если прошло менее 1000 микросекунд
         1. Если прошло менее 500 микросекунд
            1. На индикаторе устанавливается старое значение индикатора
            2. Время горения индикатора устанавливается с начала массива *delaysArr*.
         2. Если прошло более 500 микросекунд
            1. На индикаторе устанавливается новое значение индикатора
            2. Время горения индикатора устанавливается с конца массива *delaysArr*.
      3. Если прошло более 1000 миллисекунд
         1. На индикаторе устанавливается новое значение индикатора
         2. Состояние мигания на индикаторе отключается
         3. Предыдущее значение индикатора становится равно текущему
   2. Иначе
      1. Устанавливается новое значение индикатора
      2. Время горение индикатора устанавливается в 2000 микросекунды
   3. Если время горения индикатора неравно 0
      1. На пине индикатора устанавливается высокий сигнал
      2. Задержка на определенное в прошлых шагах время горения индикатора
      3. На пине индикатора устанавливается низкий сигнал
   4. Задержка на (3000 – «время горения индикатора») микросекунд
2. Если это последний индикатор
   1. выход из функции
3. Иначе
   1. Переход к шагу 1

## Функция “dateMode”

1. Запись значений даты в массив *digits* (вызов функции *updateDate*)
2. Вывод значений на индикаторы (вызов функции *show*)
3. Если минуты, часы на модуле реального времени равны «0», а секунды меньше 3х
   1. Запись значений в массив *digits* (вызов функции *updateDate*)
4. Иначе
   1. Если была нажата кнопка 1
      1. Запись значений даты в массив *digits* (вызов функции *updateTime*)
      2. Переключение в режим часов (*mainMode = true*)
      3. Выход из функции
   2. Если была нажата кнопка 2
      1. Переход в режим настройки часов (*navigate(setupKey)*)
   3. Если была нажата кнопка 3
      1. Переход в режим дивергенсметра (*navigate(diverKey)*)
5. Переход к пункту «2)»

**Текст исходного кода (листинг) с комментариями**

void updateTime() {

second = Clock.getSecond();

minute = Clock.getMinute();

hour = Clock.getHour(h12, PM);

previousNumber[0] = [0] = hour / 10;

previousNumber[1] = digits[1] = hour % 10;

previousNumber[2] = digits[2] = minute / 10;

previousNumber[3] = digits[3] = minute % 10;

previousNumber[4] = digits[4] = second / 10;

previousNumber[5] = digits[5] = second % 10;

}

void show()

{

for (int i = 0; i < 6; i++) {

int myDelayMksek;

if (flagBlink[i]) {

int myMillis = millis() - blinkTimer[i];

if (myMillis <= 1000) {

if (myMillis < 500) {

setNumber(previousNumber[i]);

myDelayMksek = delaysArr[myMillis / 10];

}

else {

setNumber(digits[i]);

myDelayMksek = delaysArr[(1000 - myMillis) / 10];

}

}

else {

setNumber(digits[i]);

flagBlink[i] = false;

previousNumber[i] = digits[i];

}

}

else {

setNumber(digits[i]);

myDelayMksek = 2000;

}

if (myDelayMksek) {// если 0, то пусть вообще не загорается

digitalWrite(lampKeys[i], HIGH);

delayMicroseconds(myDelayMksek);

//потушим первый индикатор

digitalWrite(lampKeys[i], LOW);

}

delayMicroseconds(3000 - myDelayMksek);

}

}

void timeShift() {

int bufSecond = Clock.getSecond();

if (second != bufSecond) {

second = bufSecond;

previousNumber[4] = digits[4];

previousNumber[5] = digits[5];

digits[4] = second / 10;

digits[5] = second % 10;

if (second == 0) {

minute = Clock.getMinute();

previousNumber[2] = digits[2];

previousNumber[3] = digits[3];

digits[2] = minute / 10;

digits[3] = minute % 10;

if (minute == 0) {

hour = Clock.getHour(h12, PM);

previousNumber[0] = digits[0];

previousNumber[1] = digits[1];

digits[0] = hour / 10;

digits[1] = hour % 10;

}

}

}

}

void updateDate() {

digits[0] = (Clock.getYear() / 10) % 10;

digits[1] = Clock.getYear() % 10;

digits[2] = Clock.getMonth(Century) / 10;

digits[3] = Clock.getMonth(Century) % 10;

digits[4] = Clock.getDate() / 10;

digits[5] = Clock.getDate() % 10;

}

void dateMode() {

updateDate();

while (true) {

show();

if (Clock.getSecond() < 3 && !Clock.getMinute() && !Clock.getHour(h12, PM))

updateDate();

if (pushedKey = pushButton()) {

if (pushedKey == 1) {

updateTime();

mainMode = true;

return;

}

else if (pushedKey == 3)

navigate(diverKey);

else if (pushedKey == 2)

navigate(setupKey);

}

}

}