Двухканальный стабилизированный диммер

Михаил Милославский
г. Москва
E-mail: **MMiloslavsky@yandex.ru**

5. ПРОГРАММНАЯ ЧАСТЬ

Продолжение. Начало в №10-12/2008

Программа составлена и отлажена в бесплатной среде разработки Visual Micro Lab версии 3.12 (http://www.amctools.com). Код программы написан на языке ассемблер, он содержит более 1500 строк, и занимает в памяти программ МК более 3 КБ. Задействованы все 32 регистра, 51 байт оперативной памяти (SRAM, ОЗУ) и 45 байт энергонезависимой памяти EEPROM.

На первый взгляд, возможности МК ATmega16, имеющего 16 КБ программной памяти, 1 КБ SRAM, 512 байт EEPROM, а также богатый набор периферийных устройств, кажутся избыточными. На самом деле, выбор МК "с запасом" сделан умышленно, чтобы иметь возможность совершенствовать устройство и наращивать его функциональные возможности (см. раздел **Дальнейшие усовершенствования**).

При необходимости код может быть перенесен на другие МК семейства ATmega. Портировать код на МК ATtiny не рекомендуется, т.к. в них отсутствует аппаратный умножитель. Программная реализация умножения нежелательна, т.к. при этом потеряется скорость реакции на изменение входного напряжения, а значит, ухудшится качество стабилизации.

5.1. Блок-схема алгоритма

Структура программы показана на **рис. 5**. Каждый блок начинается с названия файла, в котором содержится программный код данного блока. Подпрограммы обозначены блоком меньшего размера. Среди них есть функции, т.е. подпрограммы, возвращающие значения. Для упрощения они тоже называются подпрограммами. Отличие подпрограмм, расположенных в файле ProceduresINT.asm, от подпрограмм в файле ProceduresEXT.asm заключается в том, что первые предназначаются только для данного проекта, а вторые являются универсальными и могут найти применение в других

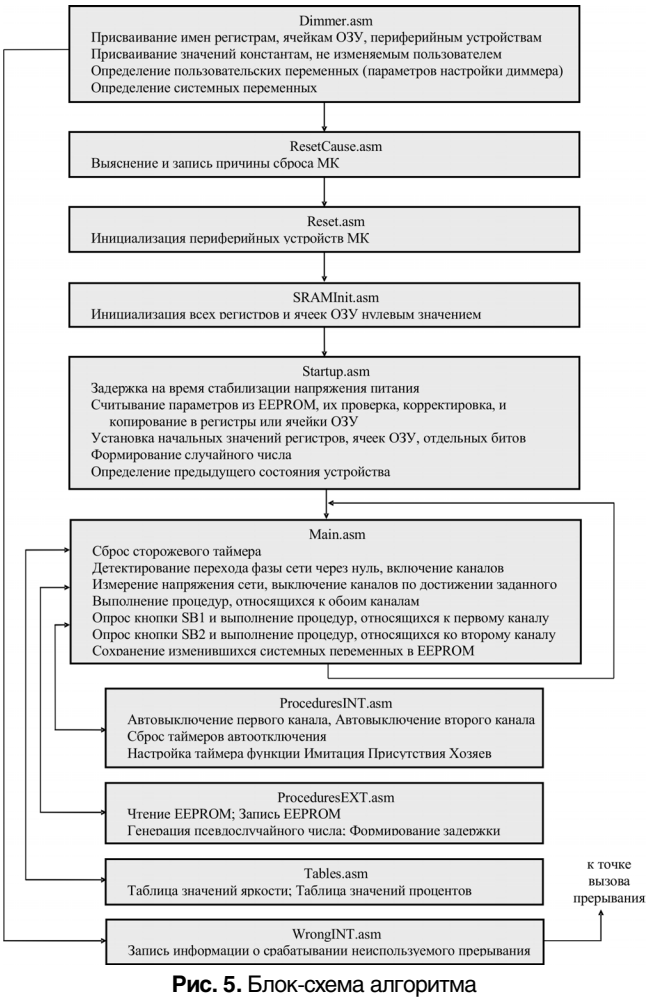

Рис. 5. Блок-схема алгоритма

проектах. Блок Startup.asm тоже использует некоторые подпрограммы, но для упрощения восприятия эти связи на блок-схеме не показаны.

Файл определений m16def.inc (на схеме не показан) модифицирован с учетом используемых в проекте ресурсов МК. В файле закомментированы ресурсы, которые определяются проектом и используются под другими названиями. Помимо этого, поставляемый в составе VMLab файл определений устарел (файл датирован 2001 годом), он не содержит определения некоторых регистров, битов и областей памяти. Применять новый файл определений из состава последней версии AVR Studio нельзя, т.к. в нем есть команды, не поддерживаемые ассемблером VMLab. Вариант использования нового ассемблера тоже не проходит – VMLab выдает ошибку несовместимости.

5.2. Описание программы

Далее кратко рассмотрены ключевые моменты программной части проекта. Дополнительная информация содержится в комментариях, которыми снабжена практически каждая команда программы. Алгоритм работы второго канала полностью аналогичен алгоритму работы первого, поэтому комментарии приводятся только для первого канала. По этой же причине в названиях регистров, процедур и т.п. номер канала либо не указывается совсем, либо обозначается буквой "X" или "x". Команды, начинающиеся с символа комментария (";") в самом начале строки, предназначены для облегчения разработки, отладки и тестирования. Они ускоряют ход выполнения программы на этапе отладки. Закомментированные команды, перед которыми есть символ табуляции, не используются в данном проекте, но оставлены в качестве шаблона для других применений. Это ускоряет ход выполнения основной программы и сокращает общий объем кода.

Программа оптимизирована по быстродействию, поскольку, чем меньше времени потребуется для выполнения кода, тем большей максимальной яркости можно достигнуть (эта взаимосвязь подробно рассмотрена далее). Особое значение имеет время выполнения процедуры ADCSampling. От этого зависит качество стабилизации, т.е. насколько быстро устройство может реагировать на колебания сетевого напряжения. В текущей реализации процедура ADCSampling выполняется за 21 мкс (без учета времени преобразования АЦП, на которое требуется 27 мкс).

Адресное пространство энергонезависимой памяти (EEPROM) разделено на три области. В первой хранятся настраиваемые пользователем параметры устройства, во второй – системные переменные, в третьей – результаты диагностики. Первая область отделена от второй несколькими пустыми ячейками, значения которых равны $FF. Третья область находится в конце адресного пространства. Так сделано для удобства восприятия данных в окне симулятора VMLab, а также для упрощения ориентирования в файле dimmer.eep, когда требуется определить параметры настройки. Первые 16 ячеек EEPROM не используются. Помимо удобства восприятия, это предохраняет содержимое первой ячейки от случайного изменения, когда регистр адреса EEPROM равен нулю. Нулевое значение присваивается этому регистру во время инициализации (блок Reset.asm). По той же причине не используется и последняя ячейка EEPROM, чтобы исключить изменение ее содержимого, при EEARH=$01 и EEARL=$FF.

Принимая во внимание ограниченное количество циклов перезаписи EEPROM (100 000 согласно описанию), в памяти сохраняются только самые необходимые данные, а именно уровень установленной пользователем яркости и текущее состояние устройства.

Прерывания не задействованы. Программный код, расположенный в блоке Main.asm, выполняется в бесконечном цикле. Тем самым удается повысить быстродействие за счет отказа от команды reti и команд работы со стеком. Тем не менее, в целях диагностики неисправностей, срабатывание любого прерывания фиксируется и запоминается в EEPROM (подпрограмма WrongINT.asm). Также в целях диагностики запоминается причина, вызвавшая сброс МК (блок ResetCause.asm). Ячейки диагностики, как уже говорилось, находятся в конце области EEPROM. Если значение хотя бы одной из них отлично от нуля, это говорит о наличии неисправности.

Сторожевой таймер настроен на максимальный период срабатывания (2 секунды). Чем больше интервал, тем легче заметить срабатывание таймера. Особенно это помогает на этапе отладки.

Супервизор питания запрограммирован на минимальный порог 2,7 В. За счет этого обеспечивается надежный запуск МК с учетом большой емкости конденсатора, установленного в фильтре питания. Супервизор гарантирует стабильный запуск независимо от того, на какое время пропадет напряжение в сети.

Энергосберегающие режимы не используются. Это связано с тем, что в течение примерно 9,5 мс каждого полупериода ведется непрерывное измерение сетевого напряжения, а в оставшиеся 0,5 мс выполняется основная программа. Переводить МК в спящий режим на столь короткое время не имеет смысла. К тому же, возврат МК из любого спящего режима требует дополнительных затрат времени, что сказывается на быстродействии.

Выключение АЦП на время выполнения основной программы тоже не имеет смысла. В ходе измерений было установлено, что при этом потребляемый МК ток снижается всего на 10 мкА.

Во время инициализации МК интерфейс JTAG и компаратор в целях снижения энергопотребления отключаются.

Выходы МК переключаются одновременно. Это необходимо для того, чтобы оба канала имели равные временны*е* задержки и работали одинаково. С этой целью в программу введен специальный буферный регистр rOutChannels. Изменение состояния выходов осуществляется только командой out pMainOut, rOutChannels. После старта МК буферный регистр содержит копию состояния порта pMainOut, большинство выводов которого, по аналогии с другими неиспользуемыми выводами, настроены как входы с внутренними подтягивающими резисторами. Поэтому никакие биты регистра rOutChannels (кроме двух старших) не должны модифицироваться программой. Также не должно изменятся состояние порта напрямую, т.к. любое изменение будет перезаписано содержимым буферного регистра.

Аналого-цифровой преобразователь (АЦП), имеющий разрешение 10 бит, используется в режиме 8-битного преобразования, поскольку высокая точность измерений не требуется. Это позволяет АЦП работать на максимальной частоте (500 кГц), что в свою очередь обеспечивает высокое

быстродействие. Согласно [15, раздел 2.8], допускается работа АЦП на частотах до 1 МГц. Уменьшенное до 8-ми бит разрешение служит также дополнительной защитой от помех. Для сокращения количества команд управления, АЦП работает в режиме автозапуска, сигналом к которому является окончание предыдущего цикла преобразования. Первое преобразование запускается при инициализации МК.

Таймеры не используются. Как выяснилось, они создают помехи работе АЦП. Для их исключения существует специальный энергосберегающий режим "ADC Noise Reduction". Но поскольку режимы энергосбережения по причинам, указанным ранее, не используются, временны*е* выдержки реализованы на основе регистров или ячеек ОЗУ.

По окончании инициализации всем регистрам и ячейкам ОЗУ присваивается нулевое значение (блок SRAMInit.asm). Таким образом, не нужно помнить о том, чтобы все переменные были проинициализированы, т.е. не содержали бы случайных чисел. Такой подход сокращает количество ошибок типа "то работает, то не работает", "работает то так, то эдак". Кроме этого, значительно облегчается поиск ошибок.

После инициализации регистров и ОЗУ продолжение программы приостанавливается на несколько секунд (осуществляется программная задержка). За это время напряжение питания МК и образцовое напряжение АЦП успевают стабилизироваться на номинальном уровне.

Для повышения быстродействия задействованные ячейки EEPROM копируются перед началом работы либо в регистры, либо в ОЗУ (блок Startup.asm). В большинстве случаев для этой цели используется специальный макрос EEPROMtoSRAM, находящийся в файле Macros.asm (на блок-схеме алгоритма не показан).

При таком подходе появляется возможность проверки считанных из EEPROM данных. Если введенный пользователем параметр настройки является недопустимым, он преобразуется к ближайшему допустимому значению (меньшему или большему). Это действие, условно называемое нормализация, выполняется не для всех ячеек, а только для тех, допустимые значения которых однозначны, а также для тех, *не*допустимые значения которых могут привести к явным ошибкам в работе устройства.

Допустимые значения ячеек EEPROM, а также значения по умолчанию указаны при описании ячеек в комментариях (блок Dimmer.asm).Значения некоторых ячеек умышленно ограничены диапазоном, удобным для восприятия и запоминания. Например, продолжительность включенного состояния функции ИПХ, задаваемая ячейкой eOwnersAtHomeOnPeriod, ограничена значением 240 минут, хотя функция могла бы исправно работать и при eOwnersAtHomeOnPeriod=255. Однако значение 240 минут (4 часа) более привычно для запоминания и использования, нежели 255 минут (4,25 часа).

Одной из задач инициализации является однократное измерение сетевого напряжения. Поскольку заранее невозможно предсказать, в какой момент времени светильник будет подключен к сети, результатом этого измерения является случайное число, используемое далее в подпрограмме генератора псевдослучайных чисел (блок Procedures.EXT.asm).

Перед тем как передать управление основному блоку программа определяет, в каком состоянии находилось устройство перед отключением питания. В зависимости от результата устанавливаются те или иные флаги, сообщающие основной программе о том, какие действия необходимо предпринять. Например, если в момент пропадания сетевого напряжения был включен первый канал, то после восстановления электроснабжения он включится автоматически (если это было разрешено пользователем).

Как уже отмечалось, код основного блока Main.asm выполняется в бесконечном цикле. Так как детектирование перехода сетевого напряжения через нуль осуществляется в каждом полупериоде, цикл Main.asm повторяется каждые 10 мс.

Первая команда основного блока – сброс сторожевого таймера. Следует отметить, что это единственное место во всей основной программе, где осуществляется сброс сторожевого таймера.

Первоначально для детектирования момента перехода сетевого напряжения через нуль был выбран следующий алгоритм: АЦП циклически производил считывание входного напряжения и сравнивал результат преобразования с заданной константой. Как только происходило совпадение, подавалась команда на включение каналов, и цикл завершался. Из-за отсутствия кварцевой стабилизации тактовой частоты МК, для точного обнаружения момента перехода фазы сети через нуль требовался подбор константы под конкретный экземпляр МК. Был альтернативный вариант: вместо использования константы осуществлять подбор путем калибровки внутреннего RC генератора. И то, и другое отрицательно сказывалось на повторяемости устройства. Но основная причина, побудившая изменить алгоритм, заключалась в невысокой помехоустойчивости. Действительно, если помеха (всплеск сетевого напряжения) возникала близко к моменту перехода фазы сети через нуль, МК продолжал измерять входное напряжение, ожидая его совпадения с константой. Поскольку после прекращения помехи входное напряжение оказывалось больше заданного константой, МК был вынужден оставаться в цикле измерений до следующего перехода фазы. Так как во время измерения напряжения каналы находятся в выключенном состоянии, визуально такой "простой" выглядел как моргание ламп(ы), т.к. в течение 10 мс напряжение на нагрузке отсутствовало.

Измененный алгоритм основан на сравнении не с константой, а с предыдущим результатом измерения. Срабатывание происходит, когда текущий результат становится больше предыдущего. В рассмотренном ранее примере МК будет детектировать помеху как напряжение, превышающее предыдущее. Это приведет к моментальному выходу из цикла, что визуально будет практически не заметно. В обычном режиме (без воздействия помех) МК осуществит выход из цикла, как только сетевое напряжение сменит направление с убывающего на возрастающее, т.е. сразу после перехода сетевого напряжения через нуль. Таким образом, отпадает необходимость в кварцевом резонаторе, калибровке внутреннего генератора и АЦП. Также не требуется дополнительная константа и настройка устройства под конкретный экземпляр МК.

Погрешность нового алгоритма нетрудно подсчитать. Для определения момента, когда сетевое напряжение начало возрастать, достаточно двух измерений. Время одного преобразования АЦП составляет 27 мкс (см. раздел **Выбор_константы_cADCSamplesCount**). Время на обработку полученного результата (первые 5 команд процедуры PhaseDetect) составит в худшем случае 8 мкс. Значит,

максимальная погрешность равна 27 + 27 + 8 = 62 мкс. Согласно формуле

$U = A \cdot \sin(2 \cdot p \cdot f \cdot t)$,

при такой задержке мгновенное напряжение в сети будет находиться на уровне $U = 310 \cdot \sin(2 \cdot 3{,}14 \cdot 100 \cdot 62 \cdot 10^{-6}) = 12$ В (относительно номинального сетевого напряжения). Это совпадает с данными, полученными экспериментально с помощью осциллографа. Причем результат оставался стабильным для всех экземпляров МК.

Включение с интервалом 10 мс лампы мощностью 60 Вт с сопротивлением нити $230^2 / 60 = 882$ Ом при напряжении 12 В создаст ток всего 12 / 882 = 14 мА. Такой мизерный ток не способен заметно повлиять на сокращение срока службы лампы, создание помех и т.п.

Ввиду разветвленности алгоритма основной программы его наглядное изображение в графическом виде не приводится, т.к. будет затруднительно для восприятия. Отдельные процедуры основной программы описаны далее. Облегчить понимание логики работы призваны комментарии кода, а также раздел **Управление**.

Функция ИПХ использует в качестве счетчика времени во включенном и выключенном состоянии отдельную процедуру. Имеющуюся процедуру TurnOffTimeoutCh1 использовать нельзя. Тому есть три причины. Во-первых, она строго индивидуальна для каждого канала. Во-вторых, ее модификация потребовала бы слишком больших усилий (понадобилось бы большое количество условных переходов). В-третьих, если продолжительность включенного или выключенного состояния составляет нечетное число, скажем, 3 минуты, то пришлось бы делить его поровну между двумя процедурами и иметь дело с дробными числами. Вариант, когда процедура выполняется только в одном из каналов, тоже не подходит, т.к. принято, что программный код обоих каналов одинаков (для удобства модификации и подсчета времени выполнения программы).

В функции ИПХ практически везде опрашивается бит лишь одного (первого) канала. Это допустимо благодаря идентичности каналов, а также благодаря тому, что в данном режиме каналы управляются синхронно.

В момент включения функции ИПХ устанавливаются биты ebPowerState. Это сделано на случай пропадания сетевого напряжения в момент включения. Таким образом, при восстановлении напряжения работа функции будет продолжена.

В функции автоотключения процентное значение, на которое уменьшается яркость, умышленно ограничено интервалом от 10 до 90%. Такой диапазон хорошо подходит для визуального восприятия. Кроме того, уменьшение яркости менее чем на 10% недостаточно заметно.

Для определения значения регистра rSoftOffBrightChX, хранящего величину уменьшенной яркости с учетом заданного пользователем процента, в целях ускорения программы используется табличный метод (таблица PercentageTable). Для каждого заданного пользователем значения процента Y, в таблице находится округленное число X, заранее вычисленное по формуле: $X = ((100 - Y) / 100) \cdot 256$. Таблица состоит из (90 – 10) + 1 = 81 ячейки. Когда истекает время непрерывной работы канала, происходит обращение к таблице, текущее значение яркости умножается на полученное из таблицы число. Поскольку это число предварительно было умножено на 256, результат делится на 256 путем отбрасывания старшего байта, после чего результат округляется. Например, предположим, что текущая яркость, т.е. число в регистре rBrightnessChX, равно 200, а заданный пользователем процент снижения яркости равен 25%. По истечении времени непрерывной работы канала программа обратится к таблице, в которой смещению 25 соответствует число 192. Программа произведет умножение (200 • 192 = 38400), разделит результат на 256 (38400 / 256 = 150), округлит его (в данном случае в этом нет необходимости) и запишет в регистр rSoftOffBrightChX. Воспользовавшись обычным калькулятором, нетрудно убедится, что исходная яркость (200), уменьшенная на 25%, равна 150.

Несмотря на кажущееся сходство битов bIsChannelOn и ebPowerState, объединять их воедино нельзя. Вместе они предотвращают включение канала, если во время его выключения по истечении времени непрерывной работы был сбой в электросети.

Благодаря наличию на входах МК внутреннего триггера Шмитта и линии задержки, устранение дребезга контактов кнопок SB1 и SB2 не требуется.

Обработка состояния кнопок построена таким образом, что так называемое вторичное действие не поддерживается. Например: в выключенном состоянии удерживание кнопки приводит к включению канала во втором режиме. Если продолжать удерживать кнопку, то после того как канал включился, его яркость не начнет изменяться, как это обычно происходит при включенном канале. Яркость можно будет изменить только после отпускания кнопки и повторного ее удерживания. Такой подход более эргономичен, к тому же он позволяет исключить ошибки, связанные с детектированием нажатия и удерживания кнопок в зависимости от функции, режима, и текущего состояния устройства.

Реализация обработчика кнопок основана на анализе значения регистра rButtonXHoldTime, в котором хранится длительность удерживания кнопки в нажатом положении, а также на сравнении его значения с константой cButtonOnHoldSense, определяющей временной порог, начиная с которого нажатие на кнопку распознается как удерживание. В текущей версии этот порог равен 1 секунде. Регистр rButtonXHoldTime может иметь четыре значения:

1) rButtonXHoldTime = 0 (кнопка не нажата)

2) 0 < rButtonXHoldTime < cButtonOnHoldSense (кнопка нажата)

3) rButtonXHoldTime = cButtonOnHoldSense (кнопка удерживается)

4) rButtonXHoldTime = 255 (кнопка остается в нажатом положении после удерживания)

Последнее значение сигнализирует о том, что надо дождаться отпускания кнопки. Пока этот момент не наступит, никакие действия, связанные с обработкой состояния кнопки, не выполняются.

Подпрограммы чтения и записи EEPROM работают только с первыми 256 байтами. Это упрощает адресацию, т.к. старший регистр адреса всегда равен нулю и не используется.

Если в момент обращения к EEPROM идет обработка предыдущего запроса, то подпрограмма чтения дожидается его окончания. Подпрограмма записи в аналогичной ситуации немедленно прекращается. Это вполне допустимо, поскольку задержка записи даже в несколько десятков миллисекунд

не нарушает работу основной программы, и не заметна для пользователя.

В описании МК сказано, что EEPROM допускает до 100 000 циклов перезаписи. Поэтому подпрограмма осуществляет запись в ячейку только в том случае, если записываемые данные отличаются от тех, что уже записаны в этой ячейке.

В основе подпрограммы генерации псевдослучайного числа RandomNumber8bit лежит алгоритм, описанный в [6, раздел 9.33 "Последовательности, генерируемые регистрами сдвига с обратными связями"]. Используется программно реализованный 8-ми разрядный регистр сдвига с отводами от 7-го, 5-го, 4-го и 3-го разряда, над которыми производится логическая операция ИСКЛЮЧАЮЩЕЕ ИЛИ (XOR, в МК соответствует команде EOR).

Входными данными подпрограммы является т.н. seed – число, определяющее начало псевдослучайной последовательности. Это число должно быть отлично от нуля. В противном случае оно заменяется числом 113. Почему именно 113? Потому что это первое число от начала списка, удачно располагающееся с точки зрения возвращаемого результата, умноженного на два (см. замечание о недостатке ниже). Первоначально seed генерируется путем измерения сетевого напряжения в момент подключения устройства к сети, что гарантирует случайный характер этого числа.

Подпрограмма возвращает псевдослучайное число в диапазоне от 1 до 255. Период повторения чисел равен $(2^8 - 1) = 255$.

Для повышения скорости выполнения подпрограммы и сокращения количества кода содержимое регистра SREG не сохраняется, хотя подпрограмма изменяет состояние некоторых его флагов.

Практическая проверка алгоритма показала, что распределение случайных чисел получается достаточно равномерным во всем диапазоне. Таблицу всех значений псевдослучайной последовательности можно посмотреть в прилагаемом файле (файл *RandomNumber8bit.xls*). В таблице перечислены числа, возвращаемые при каждом вызове подпрограммы, начиная с seed=1.

Если внимательно проанализировать список чисел, то можно заметить следующий недостаток. Возвращаемое значение зачастую равно (или примерно равно) предыдущему значению, умноженному на два. Всего насчитывается 61 такая комбинация. При этом количество последовательных повторений (т.е. когда такой результат получается при каждом последующем вызове функции) варьируется от 1 до 7. Статистика повторов распределяется следующим образом (**таблица 2**).

Таблица 2

Количество последовательных повторений	Число комбинаций
1	30
2	15
3	8
4	4
5	2
6	1
7	1

5.3. Таблица яркости

Восприятие уровня яркости человеческим глазом имеет нелинейную зависимость. График этой зависимости приведен на **рис. 6**. Он взят из документа [16] компании Lutron Electronics – известного с 1961 года производителя регуляторов освещения. По горизонтальной оси графика отложены значения яркости, воспринимаемые человеческим глазом, по вертикальной – ее реальные значения, измеренные прибором. Будем считать, что значения по вертикальной оси соответствуют напряжению на лампе. Экспериментальная проверка подтвердила, что это допустимо.

Используя указанные на графике соотношения, требуется составить таблицу яркости BrightnessTable, т.е. перейти от указанных на рисунке значений в процентах к числам из диапазона 1…255, пригодным для дальнейшей обработки в МК.

Как видно из размещенной на рисунке формулы, а также из ее графического представления, функцией является парабола, описываемая полиномом второй степени:

$y(x) = k_1 \cdot x^2 + k_2 \cdot x + k_3$,

где k_1, k_2, k_3 – неизвестные коэффициенты.

Для нахождения коэффициентов воспользуемся средствами программы Mathcad [17, глава 16]. Файл с описываемым далее расчетом в формате Mathcad прилагается (файл *Brightness_table.xmcd*). Взяв за основу значения трех точек, указанных на рисунке, задаем векторы экспериментальных данных, дополнив их минимальными и максимальными значениями. В точке x=0, значение y равно 1, а не 0, т.к. понятия "нулевая яркость" не существует:

data_x:=(0 255·22,4% 255·50% 255·77,5% 255)^T

data_y:=(1 255·5% 255·25% 255·60% 255)^T

Далее рассчитываем полиномиальную регрессию с помощью встроенной функции Mathcad, указывая при этом степень полинома:

k:=regress(data_x,data_y,2)

Функция возвращает вектор из пяти элементов, последние три из которых являются искомыми коэффициентами (k_1, k_2, и k_3 соответственно):

k_5=3.961 x 10^{-3} k_4=-0.013 k_3=0.87

Подставив найденные коэффициенты в формулу общего вида, получаем уравнение, описывающее исходный график:

$y(x) = 3{,}961 \cdot 10^{-3} \cdot x^2 - 0{,}013 \cdot x + 0{,}87$

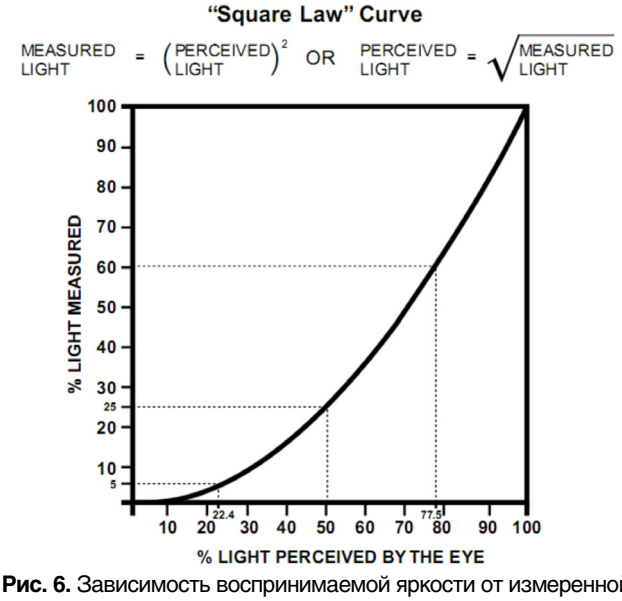

Рис. 6. Зависимость воспринимаемой яркости от измеренной

Теперь построим график данной функции, и нанесем на него для сравнения экспериментальные точки с помощью прерывистой линии (**рис. 7**).

Здесь по горизонтальной оси отложены значения регистра rBrightnessChX, обозначающие текущую яркость и задающие смещение при обращении к таблице BrightnessTable, а по вертикальной – ответные значения таблицы, записываемые в регистр rChXOffTime, которые определяют величину напряжения на лампе. Как видно из графика, приближение к исходным данным получилось достаточно точным. Кроме того, внешний вид графика совпадает с тем, который был взят за основу.

Осталось получить табличные значения функции. Задаем диапазон, для которого нужно рассчитать значения, и получаем ответ в виде **таблицы 3**.

Таблица 3

i:=0...255 y(1)=

	0
0	1
1	1
2	1
3	1
4	1
5	1
6	1
7	1
8	1
9	1
10	1
11	1
12	1

Пользуясь линейкой прокрутки таблицы, можно просмотреть все 256 ее значений. Поскольку дробная часть полученных данных нас не интересует, опции “Number of decimal places” присвоено значение 0. При этом происходит автоматическое округление. Данная опция доступна на вкладке “Number Format”, попасть на которую можно, дважды кликнув на таблице. Для отображения номеров строк, в свойствах таблицы на вкладке “Display” установлена опция “Show column\row labels”.

Записывать полученные табличные данные в память программ МК напрямую нельзя, т.к. для этого они должны быть представлены в специальном формате:

.db число 1, число 2
.db число 3, число 4

Для преобразования к требуемому формату можно воспользоваться средствами программы Excel. Файл, осуществляющий такое конвертирование, прилагается (файл *Mathcad_to_MCU_converter.xls*). Данные из таблицы Mathcad следует скопировать в столбец A этого файла, после чего столбцы B, C и D будут содержать те же данные, но в формате, пригодном для записи в память МК.

Следует иметь в виду, что поскольку минимальная и максимальная яркость являются величинами переменными, задаваемыми пользователем в ячейках eBrightnessMin и eBrightnessMax соответственно, результирующий график яркости будет отличаться от приведенного выше. Чтобы выяснить насколько, воспользуемся тем же файлом конвертера формата Mathcad в формат МК (файл *Mathcad_to_MCU_converter.xls*). Скопируем содержимое столбца “Данные Mathcad” из первого листа файла в одноименный столбец второго листа. Примем в качестве исходных данных eBrightnessMin=35 и eBrightnessMax=200. Выясним, какому значению соответствует ячейка 35, и заполним этим значением (в данном случае числом 5) все ячейки диапазона 35…0. По аналогии заполним все ячейки диапазона

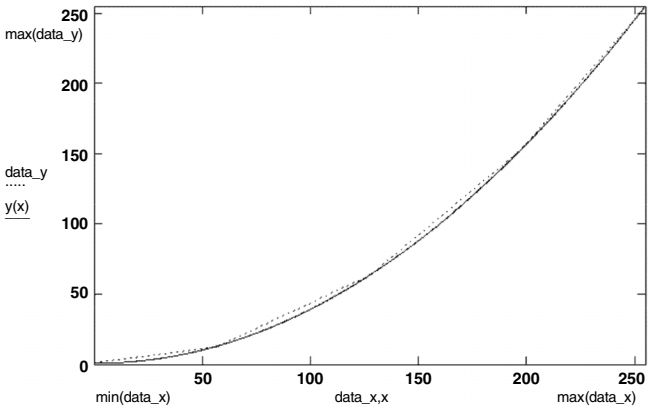

Рис. 7. График функции с нанесенными точками

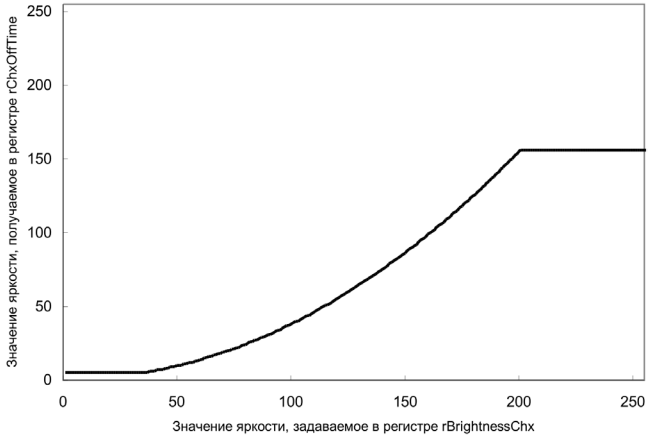

Рис. 8. График яркости с учетом минимума и максимума

200…255 значением 156. В результате, график яркости будет выглядеть, как показано на **рис. 8**.

Таким образом, при регулировке яркости используется только часть первоначального графика. Неиспользуемая часть замещается горизонтальными участками. Это – постоянные значения минимальной и максимальной яркости, продолжительность которых определяется значениями ячеек eBrightMinTimeout и eBrightMaxTimeout соответственно. О том, почему потребовалось ограничивать яркость сверху и снизу, рассказывается в следующем разделе.

Таблицу значений псевдослучайной последовательности (файл *RandomNumber8bit.zip*); вычисление табличных значений яркости (файл *Brightness_table.zip*); конвертер таблицы яркости из формата Mathcad в формат МК (файл *Mathcad_to_MCU_converter.zip*); исходные коды программы и прошивку микроконтроллера (файл *Dimmer2.zip*) вы можете загрузить с сайта нашего журнала:

http://www.radioliga.com (раздел “Программы”)
а также с сайта автора: **http://mmiloslavsky.narod.ru**

Окончание в №2/2009

Литература

15. “Characterization and Calibration of the ADC on an AVR” - Application Note AVR120, 02/06, Atmel Corporation, http://www.atmel.com/dyn/resources/prod_documents/doc2559.pdf
16. “The Eye’s Response to Light” - Lutron Electronics, 8/97, http://www.lutron.com/product_technical/pdf/360-408.pdf
17. Д.А. Гурский, Е.С. Турбина. “Вычисления в Mathcad 12” - СПб.: Питер, 2006, http://www.piter.com/books/download/978546900639