DKled Команды управления

1. Запуск

Сразу после включения контроллер ищет в папке /hcd/ файл, имя которого начинается на “0000”: “0000.hcd”, “0000-index.hcd”, “0000MyGreateProject.hcd” и т.д. Будет открыт самый первый сохранённый файл “0000…”.
Имя каждого файла должно начинаться с 4 цифр в формате HEX (0...9 a...f)

Если в директории нет нужного файла, нет самой директории на карте или даже отсутствует карта памяти как таковая, контроллеры с ПО версии 0.3 начинают работать в режиме реагирования на внешние команды из портов USART и регулярно проверяь, не появилась ли карта памяти с искомой папкой и стартовым файлом, чтобы немедленно начать читать его содержимое.

2. Синтаксис команд.

Пробелы игнорируются.*
Символы должны быть в кодировке ASCII (латинская версия).
Каждая команда должна быть на своей строке.
Все команды, кроме **SR Q<>P<>** и **UxT Q<>P<>** выполняются как только будет встречен символ конца строки, конца файла или символ начала комментария “;”.

Команда **SR Q<Count>P<>** выполняется непосредственно в процессе чтения символов после P, и будут закончены когда будет прочитано N символов, какими бы они ни были.

У некоторых команд есть числовые параметры. В таких случаях возможны два равноправных варианта:
с символом “P”перед параметром , если число записывается в 16-ричном формате (0...9 a...f),
или “N”, если число в десятичном (0...9).
В таблице команд оба варианта записи приведены один под другим. Помните, что числа больше 9 в разных форматах отличаются, к примеру , 10 HEX это 16 DEC. Параметр после Q всегда HEX.
ОЧЕНЬ ВАЖНО правильно записать значение параметра Q для **SR Q<Count>P<> ,** поскольку неправильное указание <Count> может привести к пропуску команд или ошибкам в работе, которые контроллер не отследит самостоятельно.

После символа “;” можно написать комментарий для себя, все символы после него и до начала следующей строки будут проигнорированы.

* Команды отправки текста через порты или записи в файл, и записи сырых данных в память учитывают все символы, включая пробелы.

3. Список команд

	Версия	Команда	Описание
	Вывод		
	0.3 0.5 0.7	M6 P<LED type>	Устанавливает тип управляемых устройств <LED type> может быть: WS2812 для светодиодов типа WS2812b SK6812 для светодиодов типа SK6812 WK0012 для смеси из WS2812 и SK6812 - может работать нестабильно, так как зависит от допусков по частотам SDSG90 - управление серводвигателями (стандартно для SG90 параметры можно настроить) WS00SD и SK00SD - гибридные режимы для одновременного управления цифровыми светодиодами и аналоговыми устройствами
	Вывод на светодиоды		
	0.1	S<0…7> P<RRGGBB> <…>	Присвоить на вывод <0…7> цвета в 16-чном формате (RRGGBB на светодиод). Количество светодиодов, на которые будет осуществлён вывод, определяется как количество групп по 6 HEX символов. Данная команде не осуществляет вывод цветов, только подготавливает его. Обновление происходит после команд G4 или G6.
	0.3	S<0…7> P<...> <G> <…> <L> <...> <H> <...>	Можно перечислить несколько номеров выводов чтобы записать одинаковые последовательности. Использование символа “G” или “g” сигнализирует о пропуске светодиода при обновлении цветов. Символ “H” или “h” это сокращение для FFFFFF (ярко-белый). Символ “L” или “l” - для цвета 000000 (чёрный). Если перед таким символом будет неоконченное значение цвета (меньше 6 цифр), это значение не будет учтено.
	0.5+	S<0…7> P<...> <G> <…> <L> <...> <H> <...> <swap> <...> <put> <...> <take> <...>	Можно перечислить несколько номеров выводов чтобы записать одинаковые последовательности цветов. Символ “G” или “g” сигнализирует о пропуске светодиода при обновлении цветов и прочих операциях.

			O<address> - переход на светодиод в позиции <address>. Позволяет пропускать сразу много светодиодов или, (если нужно), возвращаться к более ранним светодиодам. <address> - номер пикселя на линии, от 000 до максимального для данного контроллера (575), строго 3 цифры в формате HEX Символ “H” или “h” это сокращение для FFFFFF (ярко-белый). Символ “L” или “l” - для цвета 000000 (чёрный). Символ “?” перед цветом указывает, что нужно сравнить фактический цвет светодиода с записанным после символа (сокращения “L”, “H” так же работают, яркость также учитывается). Если указано несколько выводов, сравнивает значение цвета каждого светодиода. Если есть отличия, то увеличивает количество несовпадений цветов на 1. Можно поменять местами цвета двух пикселей или скопировать цвет одного в цвет другого. Для этого вместо цвета нужно указать одну из команд (5 символов): S<line><address> (swap) меняет местами цвета текущего пикселя и пикселя на выводе <line> (0…7) с номером <address>(000…. 23F) P<line><address> (put) копирует цвет текущего пикселя в пиксель на выводе <line> с номером <address> T<line><address> (take) копирует в текущий пиксель цвет пикселя на выводе <line> с номером <address> M<line><address> (merge) добавляет в текущий пиксель цвет пикселя на выводе <line> с номером <address> (по принципу OR) R<line><address> (reduct) ослабляет цвет пиксела на величину яркости пикселя на выводе <line> с номером <address>, где ff - 100% от яркости текущего пиксела а 0 - это 0% отдельно по каждому каналу (R, G, B) <line> - номер вывода (только одного), от 0 до 7, строго одна цифра. <address> - номер пикселя на линии, от 000 до максимального для данного контроллера 23F (575), строго 3 цифры в формате HEX (если указано несколько выводов, то макросы “S…”, “P…”, “T…”, “M…”, “R…” применяются только к пикселю на выводе с наименьшим номером, например команда S 012 P … S2001 поменяет местами с первым пикселем на 2м выводе только текущий пиксель на 0-м выводе и оставит без изменений текущие пиксели на 1м и 2м выводах.) Символ “>” повторяет последнее значение цвета, включая “L” и “H”, или последний макрос (“S…”, “P…”, “T…”, “M…”, “R…” и “?”),

			игнорируя ранее встретившиеся перемещения по массиву светодиодов (“G” и “O…”): S0005 >> эквивалентно S0005 S0005 S0005 Символ “^” и “v” делает то же, что и “>”, но в случае, если происходит повторение макросов “S…”, “P…”, “T…”, “M…”, “R…”, осуществляет соответствующее действие со следующим (“^”) или предыдущим (“v”) по порядку пикселем: S0005 ^^ эквивалентно S0005 S0006 S0007 P0005 >^ эквивалентно P0005 P0005 P0006 T0005 vv эквивалентно T0005 T0004 T0003 R0005 v^ эквивалентно R0005 R0004 R0005 Если перед одним из макросов или символов, перечисленных выше, будет неоконченное значение цвета или макроса, это вызывает ошибку и контроллер переходит к чтению следующей строки.
	0.1+	SR Q<Count> P<raw data>	Быстрое обновление цветов для первых <Count> светодиодов. <raw data> это таблица из 24*<Count> байт, напрямую записываемая в память. Любые символы, включая пробелы, переводы строки и комментарии после “P” воспринимаются как данные для записи, пока не будет прочтено нужное их количество. <Count> в формате HEX. Данная команде не осуществляет вывод цветов, только подготавливает его. Обновление происходит после команд G4 или G6.
	0.5+	Si P<brightness> Si N<brightness> G28 P<brightness> G28 N<brightness>	Устанавливает яркость После “P” от 0 до 20 (в HEX) После “N” от 0 до 32 (в DEC) 0 - минимальная яркость Если нужно указать не саму яркость, а насколько её изменить, то перед значением <brightness> пишется знак “+” или “-” Контроль яркости применяется к очередной функции S<>P<>, а не к текущему изображению.
	0.7+	SM P<Total LEDs> SM N<Total LEDs>	Максимальное количество адресных светодиодов на линии. При работе обычном режиме позволяет использовать часть адресов светодиодов как палитру: при присвоении цвета или совершении операции с пикселем, номер которого больше чем <Total LEDs>, он не попадёт в список пикселей на вывод и не будет тратить время контроллера в фазе обновления светодиодов. Но его можно будет использовать при различных операциях и/или сохранять в него часть картинки. После “P” от 0 до 240 (в HEX) После “N” от 0 до 576 (в DEC)
	0.5+	M45	Сбросить количество несовпадений цветов в ноль

	Режим Быстрого видео		
	0.3+	G36 P1	Указывает, что после команды G4 или G6 в режиме управления светодиодами контроллер перейдёт в режим Быстрого видео.
	0.3+	G36 P0	Останавливает режим Быстрого видео и возвращает контроллер в режим управления светодиодами.
	0.3+	G29 P <Start frame>	Указывает, с какого кадра начать вывод в режиме Быстрого видео, <Start frame> в формате HEX.
	0.3+	G29 N <Start frame>	То же, что и G29 P<>, но <Start frame> в формате DEC.
	0.3+	G30 P <Frequency>	Определяет частоту смены кадров для режима Быстрого видео, <Frequency> в диапазоне от 32 до 640 (HEX) Hz. Если нужно указать не саму частоту, а насколько её изменить, то перед значением <Frequency> пишется знак “+” или “-”.
	0.3+	G30 N <Frequency>	То же, что и G30 P<>, но <Frequency> в формате DEC и принимает значения от 50 до 1600 Hz. Если нужно указать не саму частоту, а насколько её изменить, то перед значением <Frequency> пишется знак “+” или “-”.
	0.3+	G31 P <Frame Length>	Назначает количество пикселей на кадр для режима Быстрого видео, в формате HEX.
	0.3+	G31 N <Frame Length>	То же, что и G31 P<>, но <Frame Length> в формате DEC.
	0.3+	G32 P <Frame number>	Назначает количество отдельных кадров для режима Быстрого видео, в формате HEX.
	0.3+	G32 N <Frame number>	То же, что и G32 P<>, но <Frame number> в формате DEC.
	0.3+	G35 P <Frames to play>	Указывает, сколько кадров (по кругу) нужно проиграть, в формате HEX. Когда <Frames to play> равно 0, режим Быстрого видео будет повторять кадры до появления команды G36 P0, а чтение файла продолжится параллельно проигрыванию. Когда <Frames to play> больше 0, будет проиграно указанное количество кадров. В этом случае контроллер продолжит чтение файла после того, как будет проигран последний кадр.
	0.3+	G35 N <Frames to play>	То же, что и G35 P<>, но <Frames to play> в формате DEC.

	0.7+	G37	Регулирует, какое действие должен совершить контроллер после того, как все кадры в режиме Быстрого видео отображены, если при этом не произошло никакое другое событие. (подробное описание см. в разделе “Кнопки и события” - “Окончание проигрывания Быстрого видео”)
	Вывод на серводвигатели		
	0.3+	G0 P<servo 0 position> <servo 1 position> … <servo 7 position>	Устанавливает позицию для сервомотора (когда контроллер в режиме управления сервомашинками или в гибридном). Строго 8 групп по 4 цифры в формате HEX. Если нужно указать не саму позицию, а насколько её изменить относительно текущей, то перед значением <servo position> для соответствующего сервомотора мотора пишется знак “+” или “-” Вместо числа можно написать символ “R” или “r” (вместо 4 цифр) - тогда значение соответствующей позиции будет сгенерировано случайно. Обновление позиций происходит после команд G4 или G6.
	0.3+	G0 Q<N> P<position> G0 Q<N> N<position>	Устанавливает позицию для N-th сервомотора (0 to 7). После Q можно перечислить несколько номеров сервомоторов. <position> записывается в формате HEX после “P” или в формате DEC после “N”. Если нужно указать не саму позицию, а насколько её изменить, то перед значением <position> пишется знак “+” или “-”. Вместо числа можно написать символ “R” или “r” - тогда значение соответствующей позиции будет сгенерировано случайно, и будет одинаковым для всех сервомоторов из списка. Обновление позиций происходит после команд G4 или G6.
	0.5+	G0 Q<N> R	Устанавливает N-й сервомотор (0 to 7) в случайное положение. После Q можно перечислить несколько номеров сервомоторов. В отличие от G0 Q<N> P r, при вызове функции в таком формате для каждого сервомотора из списка будет сгенерирована своя случайная позиция.
	0.3+	G1 P <0/1>	Выключает (0) или включает (1) изменение параметров сервомоторов (и выводов вкл/выкл) в реальном времени. Вступает в силу после команды G4 или G6 в режимах управления сервомоторами или в гибридном режиме.
	0.3+	M3	Определяет все выводы как выводы для серводвигателей (в серво и гибридном режимах). Если ранее серводвигатели не были запущены командами G4 или G6, не запускает двигатели.

	0.3+	M3 P<N> <N>...	Определяет вывод N (0...7) как вывод для серводвигателя (в серво и гибридном режимах). Можно определить несколько выводов в одной команде.
	0.3+	M4	(для гибридного режима) Определяет все выводы как выводы для цифровых светодиодов Если ранее выводы не были запущены командами G4 или G6, не запускает их.
	0.3+	M4 P<N> <N>...	(для гибридного режима) Определяет вывод N (0...7) как вывод для цифровых светодиодов. Можно определить несколько выводов в одной команде.
	0.3+	M5	Останавливает вывод в серво и гибридном режимах. (В частности, после этой команды серводвигатели перестанут стремиться сохранить положение)
	0.3+	M5 P<N> <N>...	Отключает управление серводвигателем на выводе N (0… 7). Можно отключить несколько выводов в одной команде. Выбранные серводвигатели перестанут стремиться сохранить своё положение.
	0.3+	G43 P<Minimum impulse length>	определяет минимальную длину импульса в микросекундах (в формате HEX) для режима серводвигателей. Обычно берётся из спецификации двигателя. 1 секунда это F4240 (1000 000 в DEC) микросекунд. По умолчанию 1F4 (HEX) мкс.
	0.3+	G43 N<Minimum impulse length>	То же, что и G43 P<>, но параметр в формате DEC. 1 секунда это 1000 000 микросекунд. По умолчанию 500 мкс.
	0.3+	G44 P<Maximum impulse length>	определяет максимальную длину импульса в микросекундах (в формате HEX) для режима серводвигателей. Обычно берётся из спецификации двигателя. 1 секунда это F4240 (1000 000 в DEC) микросекунд. По умолчанию 9C4 (HEX) мкс.
	0.3+	G44 N<Maximum impulse length>	То же, что и G44 P<>, но параметр в формате DEC. 1 секунда это 1000 000 микросекунд По умолчанию 2500 мкс
	0.3+	G45 P<Discretization>	Количество шагов (в формате HEX) на диапазон работы двигателя. Хорошие значения 200 (512 в DEC), B4 (180 в DEC), и т.д. По умолчанию 200 (HEX) шагов.

	0.3+	G45 N<Discretization>	То же, что и G45 P <>, но параметр в формате DEC. Хорошие значения 512, 180, и т.д. По умолчанию 512 шагов.
	0.3+	G50 P<Frequency>	определяет частоту импульсов (в формате HEX) для режима серводвигателей. Может принимать значения от 0x14 до 0x640 Гц По умолчанию 32 (HEX) Hz
	0.3+	G50 N<Frequency>	То же, что и G50 P <>, но параметр в формате DEC. Может принимать значения от 20 до 1600 Гц Обычно в пределах 30-60 Гц По умолчанию 50 Гц
	0.3+	G49	Сбрасывает значения частоты, максимальной и минимальной продолжительности импульса и количества шагов: минимальная длина импульса 0.5 мс максимальная длина импульса 2.5 мс частота импульсов 50 Гц Угол поворота 180 градусов Количество шагов 512 на диапазон работы двигателя
	0.3+	M10	(для гибридного и серво режимов) Устанавливает все выводы в режим ВКЛЮЧЕНО (ON). Выводы будут в режиме ON до команд M11 / M11P<>, M3 / M3P<> и M4 / M4P<>
	0.3+	M10 P<N> <N>...	(для гибридного и серво режимов) Устанавливает вывод(ы) <N> в режим ВКЛЮЧЕНО (ON) (N в пределах 0… 7) Выводы будут в режиме ON до команд M11 / M11P<>, M3 / M3P<> и M4 / M4P<> Действует на все перечисленные после P выводы (можно разделять их пробелами для удобства)
	0.3+	M11	(для гибридного и серво режимов) Устанавливает все выводы в режим ВЫКЛЮЧЕНО (OFF). Выводы будут в режиме OFF до команд M10 / M10P<>, M3 / M3P<> и M4 / M4P<>
	0.3+	M11 P<N> <N>...	(для гибридного и серво режимов) Устанавливает вывод(ы) <N> в режим ВЫКЛЮЧЕНО (OFF) (N в пределах 0… 7) Выводы будут в режиме OFF до команд M10 / M10P<>, M3 / M3P<> и M4 / M4P<> Действует на все перечисленные после P выводы (можно разделять их пробелами для удобства)

	Контроль времени		
	0.1+	G4 P<time>	Команда является сигналом к выводу информации на светодиоды (и серводвигатели* в версии 0.3+) <time> значение паузы в миллисекундах (в HEX) до чтения следующей строчки кода Команда учитывает, сколько времени прошло на выполнение различных команд с момента чтения предыдущей команды G4 или G5 (а также M47, M98 и нажатия кнопки) Работает независимо от пауз через команды G6, G7, G8 Команда работает следующим образом: - определяет время задержки (длительность паузы), одновременно обновляя состояние выводов (светодиоды, серводвигатели) - ждёт окончания паузы - продолжает читать и выполнять команды из файла Во время паузы контроллер способен реагировать на внешние события. * В серво и гибридном режимах серводвигатели начинают стремиться занять и сохранить соответствующее положение
	0.3+	G4 N<time> G4	То же, что и G4 P<>, но <time> (время в миллисекундах) записывается в формате DEC Без параметра - в качестве значения параметра используется 0.
	0.3+	G5 P<time> G5 N<time>	<time> значение паузы в миллисекундах (записывается в формате HEX после “P” или в формате DEC после “N”) до чтения следующей строчки кода Команда учитывает, сколько времени прошло на выполнение различных команд с момента чтения предыдущей команды G4 или G5 (а также M47, M98 и нажатия кнопки) В отличие от G4, не запускает вывод на светодиоды и серводвигатели, только приостанавливая чтение.
	0.3+	G6 P<time> G6 N<time>	Команда является сигналом к выводу информации на светодиоды и серводвигатели* <time> значение времени (записывается в формате HEX после “P” или в формате DEC после “N”) в миллисекундах от начала чтения файла текущей анимации, в который продолжить исполнение этого файла. Поскольку время отсчитывается фактически от начала открывания файла, данная команда позволяет более точно синхронизировать события. Команда работает следующим образом: - останавливает выполнение программы до момента, когда от начала чтения файла наступит <time> миллисекунд

			- по наступлению времени выхода из задержки обновляет состояние выводов (светодиоды, серводвигатели) и сразу продолжает читать и выполнять команды из файла Во время паузы контроллер способен реагировать на внешние события. Работает независимо от времени срабатывания пауз через G4, G5 однако прерывает их. * В серво и гибридном режимах серводвигатели начинают стремиться занять и сохранить соответствующее положение
	0.3+	G7 P<time> G7 N <time>	Устанавливает момент времени <time> (записывается в формате HEX после “P” или в формате DEC после “N”) в миллисекундах от начала чтения файла текущей анимации, в который продолжить исполнение этого файла. Порядок исполнения команды: - останавливает выполнение программы до момента, когда от начала чтения файла наступит <time> миллисекунд - когда этот момент наступает, разрешает контроллеру продолжить чтение и исполнение файла. До наступления соответствующего момента контроллер может реагировать на внешние события и команды. Команда не влияет на расчёт времени в командах G4 и G5, но имеет более высокий приоритет на продолжение чтения файла. В отличие от G6, не является сигналом к фактическому обновлению состояния выводов с двигателями и светодиодами.
	0.3+	G8 P<time> G8 N<time>	Устанавливает текущий момент времени с начала проигрывания текущего файла анимации в миллисекундах (записывается в формате HEX после “P” или в формате DEC после “N”) Не перематывает файл назад. Не влияет на отсчёт пауз по командам G4 и G5.
	0.3+	G9	Прерывает паузу, вызванную командами G4 или G5.
	0.3+	M24	Продолжает проигрывание файла после команды M25.
	0.3+	M25	Приостанавливает проигрывание файла, в том числе и отсчитывание всех пауз.
	0.5+	M25 P<time> M25 N<time>	Повторный вызов команды M25 спустя указанное в <time> время (записывается в миллисекундах в формате HEX после “P” или в формате DEC после “N”) приведёт к возобновлению проигрывания файла. Для отключения функции значение <time> указывается 0, или не указывается вообще (“M25 P” или “M25 P0” и т.п.)

	Кнопки и события		
	События по кнопкам		
	0.5+	M86 P<0/1>	Переключение между режимами реагирования на кнопки: 0 - реагирование на каждую кнопку отдельно В этом режиме работают все кнопки 1 - реагирование на комбинацию нажатых кнопок В этом режиме работают только кнопки с 1 по 6, остальные, если они есть, отключаются. При этом порядок нажатия не важен. Комбинация определяется исходя из того, какие кнопки были нажаты на момент окончания промежутка времени, определённого командой G27 и начавшегося с момента изменения (нажатия/отпускания) кнопок. Физическое отпускание всех кнопок как комбинация не воспринимается, но можно задать событие, соответствующее такой комбинации и вызываемое исключительно командой G25.
	0.7+	M86 Q<encoder> P<button1><button2>	Добавить энкодер. Работает в режиме реагирования на каждую кнопку по отдельности. Энкодеры сообщают своё положение посредством кода Грея (00 - 01 - 11 - 10 - 00 -...) Каждый энкодер использует два входа кнопок для передачи информации в контроллер, таким образом всего энкодеров может быть не больше половины кнопок. Например, у I5O8U2 физических 5 кнопок, и соответственно можно к нему добавить 2 энкодера. <encoder> - номер добавляемого энкодера. Для I5O8U2 это 0 или 1. Указание другого значения вызывает ошибку и команда будет проигнорирована. <button1> и <button2> - номера входов для кнопок, которые подсоединяются к энкодеру. При повороте энкодера в одну сторону вызывается событие по нажатию <button1>. В противоположную - по нажатию <button2>. События по отпусканию этих кнопок не вызываются. При добавлении энкодера <button1> и <button2> не должны быть размечены другими энкодерами, в противном случае контроллер сообщает об ошибке и игнорирует команду.

	0.7+	M86 Q<encoder>	Деинициализировать энкодер с номером <encoder> Кнопки, ранее размеченные для соответствующего энкодера, становятся свободными (в частности, их можно разметить на другой энкодер или на тот же самый но в другом порядке), и реагируют как на нажатие, так и на отпускание независимо друг от друга. Работает в режиме реагирования на каждую кнопку по отдельности.
	0.7+	M86 Q	Деинициализировать все энкодеры. Все кнопки после этой команды будут работать в обычном режиме, а разметка кнопок на энкодеры сбрасывается. Работает в режиме реагирования на каждую кнопку по отдельности.
	0.1	M96 Q<button> P<file> M96 QR<button> P<file>	Определяет, какой файл анимации запустить при нажатии (Q) или отпускании (QR) кнопки <button>. <button> - номер кнопки от 0 до 7. <file> - это HEX код от 1 до 4 цифр, от 0000 до FFFF. При этом файлы “0” и “0000” - это разные файлы. Имена файлов не рекомендуется начинать с “0” чтобы избежать путаницы. Команда может быть определена в любом месте файла, вступает в силу сразу после вызова до следующего вызова команды.
	0.3+	M96 Q<button> P<file> M96 QR<button> P<file> M96 Q<button> P M96 QR<button> P	По нажатию (Q) или отпусканию (QR) кнопки запустить указанный файл. Файл анимации указывается как код <file>, <file> - это HEX код строго из 4 цифр, от 0000 до FFFE. Если после P пусто или “FFFF”, будет запущен файл, предыдущий Основному файлу, вызвавшему эту команду либо Подпрограмму с этой командой. (Файлы, открытые в режиме Подпрограммы не учитываются.) Переход осуществляется в Основном режиме. <button> - номер кнопки от 0 до F. Номер кнопок, имеющих физические контакты, начинается с 1. Например для O6I4U2 это кнопки с 1 по 4. Остальные кнопки, включая 0, можно “нажать” виртуально командой G25. В режиме реагирования на комбинацию кнопок, после Q перечисляются кнопки, с 1 по 5, комбинация которых должна привести к соответствующему событию. Если не указана ни одна кнопка, то это значит, что задаётся событие для “нулевой” комбинации.

	0.5+	M96 QR<button> P<file> M96 QR<button> P	Дополнительно в версии 0.5 и выше в команде M96 вместо кнопки можно указать прямой номер события (в HEX, в диапазоне 0…63) через QH<event> Соответствия номеров событий и кнопок/комбинаций указаны в разделе 13.
	0.5+	M96 Q<button> L<file> M96 QR<button> L<file> M96 QH<event> L<file>	По нажатию (Q) или отпусканию (QR) кнопки запланировать указанный файл в качестве реакции на конец файла, либо команду повтора. То есть когда файл закончится или встретятся команды M2, M47, M88, M89 или M98, вместо их обработки открыть файл <file> в соответствующем этой команде режиме: M98, M98 P<>, M89 - Основной, M2, M88, M88 P<>, M47, M47 P<> - текущий режим M89 P<> - игнорируется Всё остальное как и в командах M96 Q<button> P<file> и M96 QR<button> P<file>
	0.5+	M96 Q<button> J<file> M96 QR<button> J<file> M96 QH<event> J<file>	Перейти к указанному файлу сразу после окончания проигрывания очередного цикла Быстрого Видео в случае нажатия (Q) или отпускания (QR) кнопки. Если в момент события Быстрое Видео не воспроизводилось, то выполнить переход немедленно. Режим чтения файла при переходе в новый файл (Основной или Подпрограмма) не меняется. Всё остальное как и в командах M96 Q<button> P<file> и M96 QR<button> P<file>
	0.5+	M96 Q<button> i<file> M96 QR<button> i<file> M96 QH<event> i<file>	По нажатию (Q) или отпусканию (QR) кнопки запустить указанный файл как Подпрограмму Всё остальное как и в командах M96 Q<button> P<file> и M96 QR<button> P<file>
	0.5+	M96 Q<button> O M96 QR<button> O M96 QH<event> O	По нажатию (Q) или отпусканию (QR) кнопки вернуться из режима Подпрограммы в Основной файл, вызвавший переход в Подпрограмму.
	0.5+	M96 Q<button> <command> M96 QR<button> <command> M96 QH<event> <command>	При нажатии (Q) или отпускании (QR) кнопки <button> выполнить команду <command>. В режиме реагирования на комбинацию кнопок, после Q перечисляются кнопки, с 1 по 5, комбинация которых должна привести к соответствующему событию. Если не указана ни одна кнопка, то это значит, что задаётся событие для “нулевой” комбинации. Вместо кнопки можно указать прямой номер события (в HEX, в диапазоне 0…63) через QH<event> Соответствия номеров событий и кнопок/комбинаций указаны в разделе 13.

			Команда, если она поддерживается кнопками, записывается в том виде, в котором она пишется в обычном коде. Если команда отсутствует в списке ниже, она будет выполнена в обычном режиме. Поддерживаются следующие команды (см остальные части таблицы: M98, M98 P<>, M98 P<file> M98 Q<N> P <file> <file> … (файл выбирается в момент определения действия на кнопку, а не в момент нажатия!) Если команды M98 <…> вызваны в режиме Подпрограммы ссылаются на предыдущий файл, в кнопку будет записан номер Основного файла, предыдущего Основному, вызвавшему режим Подпрограммы M89, M89 P<>, M89 P<file> M89 Q<N> P <file> <file> … (файл выбирается в момент определения действия на кнопку, а не в момент нажатия!) Если команды M89 <…> вызваны в режиме Подпрограммы ссылаются на предыдущий файл, в кнопку будет записан номер предыдущего открытого файла, будь то вызвавший этот режим Основной или Подпрограмма, и при соответствующем событии этот файл будет вызван как Подпрограмма M88, M88 P<>, M88 P<file> M88 Q<N> P <file> <file> … (файл выбирается в момент определения действия на кнопку, а не в момент нажатия!) Если команды M88 <…> вызваны в режиме Подпрограммы ссылаются на предыдущий файл, в кнопку будет записан номер предыдущего открытого файла, будь то вызвавший этот режим Основной или Подпрограмма, и при соответствующем событии этот файл будет вызван как Подпрограмма M47, M47 P<Repetitions>, M47 N<Repetitions> M46 P<Repetitions>, M46 N<Repetitions> M24, M25 G4, G4 P<time>, G4 N<time> G5 P<time>, G5 N<time> G9 M90, M90 P<seed>, M90 N<seed> M3 P<N> <N>... M5 P<N> <N>... M10 P<N> <N>... M11 P<N> <N>... G1 P <0/1> G0 Q<N> P<position>, G0 Q<N> N<position>, G0 Q<N> R G28 P<brightness>, G28 N<brightness> M91 P<choice>, M91 N<choice>

			G30 P <Frequency>, G30 N <Frequency> G35 P <Frames to play>, G35 N <Frames to play> Запускает быструю анимацию в момент нажатия G36 P0, G36 P1 Приостанавливает или запускает быструю анимацию в момент нажатия
	0.5+	M96 Q<button> M96 QR<button> M96 QH<event> M96 Q<button> T M96 QR<button> T M96 QH<event> T	Без параметра или команды - отключает событие по нажатию (Q) или отпусканию (QR) кнопки <button>. Вместо кнопки можно указать прямой номер события (в HEX, в диапазоне 0…63) через QH<event> Соответствия номеров событий и кнопок/комбинаций указаны в разделе 13. Символ “T” - включает (turn on) событие по нажатию (Q) или отпусканию (QR) кнопки <button>. <button> - номер кнопки от 0 до F. Номер кнопок, имеющих физические контакты, начинается с 1. Например для O6I4U2 это кнопки с 1 по 4. Остальные кнопки, включая 0, можно “нажать” виртуально командой G25. В режиме реагирования на комбинацию кнопок, после Q перечисляются кнопки, с 1 по 5, комбинация которых должна привести к соответствующему событию. Если не указана ни одна кнопка, то это значит, что задаётся событие для “нулевой” комбинации.
	0.1	M97 Q<button> P<0/1> M97 QR<button> P<0/1>	Выключает (0) или включает (1) событие по нажатию (Q) или отпусканию (QR) кнопки <button>.
	0.3	M97 Q<button> P<0...3> M97 QR<button> P<0...3>	Настраивает режим включения события по кнопкам: 0 - отключить событие 1 - немедленное реагирование на кнопку 2 - отработка события по окончанию исполнения файла (перехват событий по M47, M98) 3 - в случае проигрывания анимации в режиме Быстрого видео обеспечивается вывод быстрой анимации до последнего записанного в память фрейма, или конца проигрывания если он произошёл раньше. В других режимах реагирование происходит немедленно.
	0.5+	M97<...>	Более не используется
	0.3	G25 Q<button> G25 QR<button>	То же, что и нажать (Q) или отпустить (QR) кнопку <button> (включая виртуальные). Событие произойдёт в соответствии с настройками кнопки.

	0.5+	G25 Q<button> G25 QR<button> G25 QH<event> G25 Q<button> ? G25 QR<button> ? G25 QH<event> ? G25 Q<button> P<?><threshold> G25 QR<button> P<?><threshold> G25 QH<event> P<?><threshold> G25 Q<button> N<?><threshold> G25 QR<button> N<?><threshold> G25 QH<event> N<?><threshold>	То же, что и нажать (Q) или отпустить (QR) кнопку <button> (включая виртуальные). Событие произойдёт в соответствии с настройками кнопки. Вместо кнопки можно указать прямой номер события (в HEX, в диапазоне 0…63) через QH<event> Соответствия номеров событий и кнопок/комбинаций указаны в разделе 13. В режиме реагирования на комбинацию кнопок, после Q перечисляются кнопки, с 1 по 5, одновременное нажатие которых эмулируется. Если не указана ни одна кнопка, то это значит, что выбирается “нулевая” комбинация. Если после указания кнопки или комбинации кнопок есть параметр “?”, то событие произойдёт если кнопка или комбинация фактически нажата в данный момент - это полезно, если в проекте используются переключатели Если после указания кнопки или комбинации кнопок есть параметр P или N (соответствует написанию <threshold> в HEX или DEC), то событие произойдёт при следующем условии: <?> - символ “>” (больше) - количество подсчитанных к этому моменту несовпадений цветов (определяется через макрос ?<...> в команде S<...> P<...> и сбрасывается по M45) пикселов больше записанного после этого символа <?> - символ “<” (меньше) - количество подсчитанных к этому моменту несовпадений цветов пикселов меньше записанного после этого символа <?> - символ “=” (равно) - количество подсчитанных к этому моменту несовпадений цветов пикселов равно записанному после этого символа <?> - символ “!” (не равно) - количество подсчитанных к этому моменту несовпадений цветов пикселов отличается от записанного после этого символа Например запись “G25 Q5 P>3” означает, что если обнаружено больше 3 (4 и более) несовпадений цветов пикселов желаемым цветам, немедленно осуществить эмуляцию нажатия на кнопку 5

	0.3+	G26 P<delay> G26 N<delay>	Если <delay> не 0, то команда определяет, через какой промежуток времени в миллисекундах зажатая кнопка/комбинация будет воспринята как повторно нажатая. Это удобно в случае, когда кнопка отвечает за изменение какого-либо параметра (например, положение серводвигателя), так как команда позволяет заменить несколько коротких нажатий одним длительным. Значение записывается в HEX после P, или в DEC после N. Если нужно указать не сам промежуток, а насколько его изменить, то перед значением <delay> пишется знак “+” или “-”. Поскольку кнопки проверяются каждые 10 мс, промежуток времени также будет округлён до десяти в большую сторону
	0.5+	G27 P<delay> G27 N<delay>	Длительность интервала на нажатие комбинации из нескольких кнопок. Работает в режиме, когда выбор команды определяется не конкретной кнопкой, а комбинацией из нажатых. В режиме участвуют кнопки с 1 по 6, а все остальные отключаются. Поскольку в реальных условиях невозможно одновременно зажать несколько кнопок, они будут зажиматься последовательно и данный параметр определяет, через сколько миллисекунд после начала нажатия первой кнопки в комбинации считать готовую комбинацию. <delay> записывается в HEX после P, или в DEC после N и может принимать значения в пределах от 20 (0x14) до 10 000 (0x2710) мс По умолчанию это 500 мс. Если нужно указать не сам промежуток, а насколько его изменить, то перед значением <delay> пишется знак “+” или “-”. Поскольку кнопки проверяются каждые 10 мс, интервал также будет округлён до десяти в большую сторону
	Конец файла и переходы		
	0.5+	M2	Конец файла - реакция на событие отложенного перехода по кнопке (M96 Q(R)<button> L<file>) Если ранее такого события не происходило, то команда ничего не делает.
	0.5+	M46 P<Repetitions> M46 N<Repetitions>	Переназначает количество оставшихся повторений текущего файла на значение <Repetitions> (записывается в формате HEX после “P” или в формате DEC после “N”) повторений (включая первое выполнение). Не вызывает повторения файла. В основном имеет смысл при использовании вместе с командой M96 <...>

	0.1+	M47	Запустить файл анимации сначала Режим чтения файла при выполнении команды (Основной или Подпрограмма) не меняется. Если команда получена по USART или USB, она прерывает текущую паузу.
	0.3+	M47 P<Repetitions> M47 N<Repetitions>	Повторяет чтение файла сначала до тех пор, пока не будет сделано <Repetitions> (записывается в формате HEX после “P” или в формате DEC после “N”) повторений (включая первое выполнение). Если после этой команды в файле есть ещё команды, то эти команды будут выполнены после того, как будут выполнены все повторения. Режим чтения файла при выполнении команды (Основной или Подпрограмма) не меняется. Если команда получена по USART или USB, она прерывает текущую паузу.
	0.5+	M91 P<choice> M91 N<choice>	Команда осуществляет заранее определённый выбор (вместо случайного) для команды M98Q<>P<>. Выбор выпадает на файл в позиции <choice> (начинается с 0) при условии, что параметр после Q равен 0 и количество файлов <choice> не больше, чем общее количество файлов в списке (например, если в списке 3 имени файлов, разных или повторяющихся, то <choice> может принимать значения 1, 2 или 3). В случаях, когда <choice> окажется больше чем файлов в списке, и когда <choice> равен 0, команда M98 Q0 P<> будет проигнорирована. Значение записывается в HEX после P, или в DEC после N. Если нужно указать не саму выбираемую позицию, а насколько её изменить, то перед значением <choice> пишется знак “+” или “-”.
	0.5+	M88	То же, что и M47 Режим чтения файла при выполнении команды (Основной или Подпрограмма) не меняется. Если команда получена по USART или USB, она прерывает текущую паузу.

	0.5+	M88 P<file>	Запускает файл анимации, начинающийся с кода <file>, <file> - это HEX код строго из 4 цифр, от 0000 до FFFE. Если команда получена по USART или USB, она прерывает текущую паузу. Если после P пусто или “FFFF”, будет запущен предыдущий файл. Режим чтения файла при выполнении команды (Основной или Подпрограмма) не меняется.
	0.5+	M88 Q<N> P <file> <file> ...	То же, что и M88 P<file>, но переход в Подпрограмму осуществляется к одному из файлов из списка, где каждый <file> - строго 4 цифры Если <N> больше 0: Выбирает один из <N> файлов случайным образом. <N> и количество <file> - кодов должны совпадать. Если <N> ,будет меньше, то часть файлов в конце списка не будет иметь шансов чтобы оказаться выбранными, а если <N> будет больше, то будет соответствующий шанс на то, что команда будет проигнорирована. Для удобства можно разделять коды файлов пробелами. Можно писать один и тот же <file> - код несколько раз, чтобы изменить шанс на его выпадение. Если <N> равно 0: Выбирает из списка файлов файл с заранее определённым через команду M91 номером, номер начинается с 1. Если в списке будет недостаточно файлов или заранее определённый номер был обозначен на 0, команда будет проигнорирована и контроллер продолжит чтение текущего файла. Режим чтения файла при выполнении команды (Основной или Подпрограмма) не меняется. Если команда получена по USART или USB, она прерывает текущую паузу.
	0.5+	M89	Возврат из режима Подпрограммы Открывает основной файл и продолжает его чтение со строки сразу после открытия другого файла в режиме Подпрограммы. Абсолютные паузы (G6 и G7) продолжают отсчитываться от момента открытия основного файла как если бы перехода в Подпрограмму не происходило. Количество повторов (если файл должен проиграться определённое количество раз по команде M47 P<>) основного файла также сохраняется.

	0.5+	M89 P<file>	Если команда вызывается из файла, читаемого в обычном режиме (Основной файл), то она осуществляет открытие файла <file> в режиме Подпрограммы. Количество повторных проигрываний Основного файла время отсчёта абсолютных пауз (G6, G7) и позиция, с которой был осуществлён переход в подпрограмму сохраняются, так чтобы при возвращении в Основной файл его чтение продолжилось так же как если бы вместо перехода в подпрограмму её содержимое находилось непосредственно в Основном файле. Если чтение файла уже осуществляется в режиме Подпрограммы, то просто осуществляет переход.
	0.5+	M89 Q<N> P <file> <file> ...	То же, что и M89 P<file>, но переход в Подпрограмму осуществляется к одному из файлов из списка, где каждый <file> - строго 4 цифры Если <N> больше 0: Выбирает один из <N> файлов случайным образом. <N> и количество <file> - кодов должны совпадать. Если <N> ,будет меньше, то часть файлов в конце списка не будет иметь шансов чтобы оказаться выбранными, а если <N> будет больше, то будет соответствующий шанс на то, что команда будет проигнорирована. Для удобства можно разделять коды файлов пробелами. Можно писать один и тот же <file> - код несколько раз, чтобы изменить шанс на его выпадение. Если <N> равно 0: Выбирает из списка файлов файл с заранее определённым через команду M91 номером, номер начинается с 1. Если в списке будет недостаточно файлов или заранее определённый номер был обозначен на 0, команда будет проигнорирована и контроллер продолжит чтение текущего файла. Если команда вызвана из Основного файла (и не проигнорирована), то чтение выбранного файла будет осуществлено в режиме Подпрограммы. Если же файл и так читается как Подпрограмма, чтение продолжится в этом режиме. Если команда получена по USART или USB, она прерывает текущую паузу.
	0.1+	M98	То же, что и M47 Режим чтения файла при выполнении команды (Основной или Подпрограмма) не меняется. Если команда получена по USART или USB, она прерывает текущую паузу.

	0.1	M98 P<file>	Запускает файл анимации, начинающийся с кода <file>, <file> - это HEX код от 1 до 4 цифр, от 0000 до FFFF. При этом файлы “0” и “0000” - это разные файлы. Имена файлов не рекомендуется начинать с “0” чтобы избежать путаницы.
	0.3+	M98 P<file>	Запускает файл анимации, начинающийся с кода <file>, <file> - это HEX код строго из 4 цифр, от 0000 до FFFE. Если команда получена по USART или USB, она прерывает текущую паузу. Если после P пусто или “FFFF”, будет запущен предыдущий файл. При вызове из режима Подпрограммы переходит в Основной режим. Если команда получена по USART или USB, она прерывает текущую паузу.
	0.3	M98 Q<N> P <file> <file> ...	Каждый <file> код - то же, что и для M98 P<file>, строго 4 цифры. Выбирает один из <N> файлов случайным образом. <N> и количество <file> - кодов должны совпадать. Для удобства можно разделять коды файлов пробелами. Можно писать один и тот же <file> - код несколько раз, чтобы изменить шанс на его выпадение. Если команда получена по USART или USB, она прерывает текущую паузу.
	0.5+	M98 Q<N> P <file> <file> ...	Каждый <file> код - то же, что и для M98 P<file>, строго 4 цифры. Если <N> больше 0: Выбирает один из <N> файлов случайным образом. <N> и количество <file> - кодов должны совпадать. Если <N> ,будет меньше, то часть файлов в конце списка не будет иметь шансов чтобы оказаться выбранными, а если <N> будет больше, то будет соответствующий шанс на то, что команда будет проигнорирована. Для удобства можно разделять коды файлов пробелами. Можно писать один и тот же <file> - код несколько раз, чтобы изменить шанс на его выпадение. Если <N> равно 0: Выбирает из списка файлов файл с заранее определённым через команду M91 номером, номер начинается с 1. Если в списке будет недостаточно файлов или заранее определённый номер был обозначен на 0, команда будет проигнорирована и контроллер продолжит чтение текущего файла. При вызове из режима Подпрограммы переходит в Основной режим. Если команда получена по USART или USB, она прерывает текущую паузу.

	Окончание проигрывания Быстрого видео		
	0.7+	G37 G37 T	Команда G37 регулирует, какое действие должен совершить контроллер после того, как все кадры в режиме Быстрого видео отображены, если при этом не произошло никакое другое событие (нажатие на кнопку, вызов команды через порт и так далее). Параметры записываются так же, как и в случае команды M96. Без параметра или команды - отключает событие. Символ “T” - включает (turn on) событие.

	0.7+	G37 <command>	При завершению Быстрого видео выполнить команду <command>. Поддерживаются следующие команды (см остальные части таблицы): M98, M98 P<>, M98 P<file> M98 Q<N> P <file> <file> … (файл выбирается в момент определения действия, а не по завершению Быстрого видео!) Если команды M98 <…> вызваны в режиме Подпрограммы ссылаются на предыдущий файл, в действие будет записан номер Основного файла, предыдущего Основному, вызвавшему режим Подпрограммы M89, M89 P<>, M89 P<file> M89 Q<N> P <file> <file> … (файл выбирается в момент определения действия, а не по завершению Быстрого видео!) Если команды M89 <…> вызваны в режиме Подпрограммы ссылаются на предыдущий файл, в действие будет записан номер предыдущего открытого файла, будь то вызвавший этот режим Основной или Подпрограмма, и при соответствующем событии этот файл будет вызван как Подпрограмма M88, M88 P<>, M88 P<file> M88 Q<N> P <file> <file> … (файл выбирается в момент определения действия, а не по завершению Быстрого видео!) Если команды M88 <…> вызваны в режиме Подпрограммы ссылаются на предыдущий файл, в действие будет записан номер предыдущего открытого файла, будь то вызвавший этот режим Основной или Подпрограмма, и при соответствующем событии этот файл будет вызван как Подпрограмма M47, M47 P<Repetitions>, M47 N<Repetitions> M46 P<Repetitions>, M46 N<Repetitions> M24, M25 G4, G4 P<time>, G4 N<time> G5 P<time>, G5 N<time> G9 M90, M90 P<seed>, M90 N<seed> M3 P<N> <N>... M5 P<N> <N>... M10 P<N> <N>... M11 P<N> <N>... G1 P <0/1> G0 Q<N> P<position>, G0 Q<N> N<position>, G0 Q<N> R G28 P<brightness>, G28 N<brightness> M91 P<choice>, M91 N<choice> G30 P <Frequency>, G30 N <Frequency> G35 P <Frames to play>, G35 N <Frames to play> Запускает быструю анимацию заново G36 P0, G36 P1 Приостанавливает или запускает быструю анимацию заново

	0.7+	G37 P<file> G37 P	По завершению Быстрого видео запустить указанный файл. Файл анимации указывается как код <file>, <file> - это HEX код строго из 4 цифр, от 0000 до FFFE. Если после P пусто или “FFFF”, будет запущен файл, предыдущий Основному файлу, вызвавшему эту команду либо Подпрограмму с этой командой. (Файлы, открытые в режиме Подпрограммы не учитываются.) Переход осуществляется в Основном режиме.
	0.7+	G37 L<file>	По завершению Быстрого видео запланировать указанный файл в качестве реакции на конец файла, либо команду повтора. То есть когда файл закончится или встретятся команды M2, M47, M88, M89 или M98, вместо их обработки открыть файл <file> в соответствующем этой команде режиме: M98, M98 P<>, M89 - Основной, M2, M88, M88 P<>, M47, M47 P<> - текущий режим M89 P<> - игнорируется Всё остальное как и в командах G37 P<file>, M96 Q<button> P<file> и M96 QR<button> P<file>
	0.7+	G37 i<file>	По завершению Быстрого видео запустить указанный файл как Подпрограмму Всё остальное как и в командах G37 P<file>, M96 Q<button> P<file> и M96 QR<button> P<file>
	0.7+	G37 O	По завершению Быстрого видео вернуться из режима Подпрограммы в Основной файл, вызвавший переход в Подпрограмму.
	Связь с другими устройствами		
	0.3+	Ui<ID>	определяет 4х символьный идентификатор (ID) для контроллера. Необходимо при коммуникации нескольких контроллеров, чтобы их отличать. Символы с кодами 0xFF (“я”) и 0x2A (“*”, звёздочка) использовать нельзя. Пробелы игнорируются.
	0.5+	Ui<ID>	Если указать меньше 4 символов, то недостающие символы будут иметь значение “*”. По умолчанию идентификатор для DKLed имеет значение “0001”

	0.3	UxT P<raw data>	Отправляет <raw data> по UART 1 (x=1) или 2 (x=2). Байты после P отправляются как есть, включая пробелы и символы комментариев. <raw data> формируется исходя из формата принимаемых на другом конце данных. Вместо символа перевода строки записывается “\r”, символ конца строки заменяется на “\n”, символ обратного слеша (“\”) на “\\”.
	0.5+	UxT P<raw data>	Отправляет <raw data> по UART 1 (x=1) или 2 (x=2). Байты после P отправляются как есть, включая пробелы и символы комментариев. <raw data> формируется исходя из формата принимаемых на другом конце данных. Вместо символа перевода строки записывается “\r”, символ конца строки заменяется на “\n”, символ обратного слеша (“\”) на “\\”. Кроме того после символа обратного слеша (“\”) можно вставить любой запрос из команды “UxA” (<answers> версия прошивки, режим и прочие). Но, в отличие от команды “UxA”, после обработки каждого отдельного запроса команда снова переходит в режим отправки текста. Например, команды “U1A v_t_nr” и “U1T \v_\t_\n\r” выдадут одинаковые строчки, а “U1T \v_t_nr” выдаст ответ только на запрос “v”, а затем строчку “_t_nr”. Либо, (допустим цвет нашего пиксела - белый), команда “U2T P\c l 1 p no 000” и “U2T P\c1000” отправят строку “FFFFFF”, а “U2T Ppixel \c1000at line 1” отправит “pixel FFFFFFat line 1” (обратите внимание что поскольку в команде после номера пикселя нет пробела, пробела нет и после кода цвета
	0.7+	UxT P<raw data>	Если указать x=0, данные будут отправлены через USB по протоколу виртуального COM-порта Если после символа “\” поставить символ комментария, все дальнейшие символы отправлены не будут
	0.3+	UxT H<data in HEX>	Отправляет байты по UART 1 (x=1) или 2 (x=2). <data in HEX> - байты, записанные в 16ричном формате, от 00 до FF. Количество отправляемых байт определяется количеством пар цифр до конца строки.
	0.7+	UxT H<data in HEX>	Если указать x=0, данные будут отправлены через USB по протоколу виртуального COM-порта.

	0.3+	UxA <answers>	Возвращает строку с ответами по UART 1 (x=1) или 2 (x=2) в зависимости от того, какие символы написаны в <answers>. <answers> может принимать следующие значения (можно подряд): v - версия прошивки t - тип контроллера f0 - код/номер текущего файла f1 - код/номер предыдущего файла i - personal ID контроллера d - режим (LED, Servo, Hybrid, Fast) m0 ... m7 -позиция серводвигателя, которую он ПРИМЕТ после ближайшего обновления позиции s - количество только что обновлённых пикселей o0 … o7 - режим работы вывода (L - Smart LED, S - servo drive, N - none) _ - пробел n - конец строки ap - период обновления в микросекундах для режима Быстрого Видео (Fast video output mode) au - частота обновления в режиме Быстрого Видео (Fast video output mode) af - количество кадров в режиме Быстрого Видео (Fast video output mode) al - размер кадра (количество пикселей) в режиме Быстрого Видео (Fast video output mode) ac - Текущий номер кадра в режиме Быстрого Видео (Fast video output mode) at -сколько кадров осталось воспроизвести в режиме Быстрого Видео (Fast video output mode) Все числовые значения в формате HEX
	0.5+	UxA <answers>	Помимо вышеперечисленных, <answers> может принимать следующие значения (можно подряд): b - яркость u - количество найденных несовпадений цветов светодиодов p - значение заранее выбранной (через команду M91) позиции из списка файлов в командах перехода на другой файл f0 - код/номер текущего файла f1 - код/номер предыдущего файла f2 - код/номер Основного файла _ - пробел n - конец строки r - переход на следующую строку \ - символ “\” 0 - символ с кодом 0x00 h<code> - символ с кодом <code> (2 цифры, в HEX) c<line><pixel> - возвращает цвет конкретного пиксела в формате RRGGBB (в HEX):

			<line> - номер вывода, 1 символ от 0 до 7, <pixel> - номер конкретного пиксела на выводе в HEX, 3 символа от 000 до 1FF (000, 001, 002 … 1FE, 1FF) Например, “\c 2 0F4” - выдай цвет 244го пиксела на 2м выводе Между цифрами (0…F или 0…f можно ставить пробелы или любые символы кроме конца и перевода строки. Символы после последнего, третьего символа <pixel> будут выведены на порт, Например (допустим цвет нашего пиксела - белый), команда “U2A P\c l 1 p no 000” и “U2A P\c1000” отправят строку “FFFFFF”, “U2A P\c line 1 pixel number 000” воспринимается как “U2A P\ce1eber 000” и мы получим “000000”, и сигнал об ошибке (“e” и “b” это тоже числа, и поскольку в позиции <line> стоит число больше 7, контроллер выдаёт цвет для пиксела на 0 выводе). После чего контроллер продолжит читать и исполнять команды.
	0.7+	UxA <answers>	Если указать x=0, данные будут отправлены через USB по протоколу виртуального COM-порта.
	0.3+	UxR <0/1>	Выключает (0) или включает (1) приём данных через UART 1 (x=1) или 2 (x=2).
	0.3+	UxF <0/1>	Выключает (0) или включает (1) ожидание завершения вывода данных через UART 1 (x=1) или 2 (x=2). Ожидание завершения данных означает, что контроллер продолжает чтение файла сценария только после того, как все отправляемые байты были отправлены через соответствующий порт. В противном случае чтение файла может продолжиться пока байты ещё отправляются. Функция полезна для синхронизации нескольких контроллеров.
	0.3+	UD <0/1>	Выключает (0) или включает (1) вывод служебной информации через UART2 (версии до 0.5) или USB (версия 0.7+)
	0.5+	UxS <0/1>	Переводит UART 1 (x=1) или 2 (x=2) в режим приёма обычных команд (0) или реагирования на отдельные байты (1). В режиме реагирования на отдельные байты UART фактически эмулирует нажатие на кнопку или на комбинацию кнопок. В этом режиме значимыми являются биты со 1 по 5 (битовая маска 00111111, или числа 0, 1,2,...63). Принятый байт в этом случае - это номер события. События задаются командами M96 (при использовании синтаксиса M96 QH<event>... число <event> соответствует принимаемому байту)

			Соответствия номеров событий и кнопок/комбинаций указаны в разделе 13. Если кнопки работают в режиме комбинации (M86 P1), каждый бит соответствует кнопке в комбинации (1й - первая кнопка, 5й - 5 кнопка). То есть команда M96 Q P<> помимо “нулевой” кнопки также определяет реакцию на число 0, M96 Q1 P<> - на число 1 M96 Q2 P<> - на число 2 M96 Q12 P<> - на число 3 … M96 Q5 P<> - на число 15 … M96 Q2345 P<> - на число 30 M96 Q12345 P<> - на число 31 Если кнопки задаются в режиме реагирования на отдельные кнопки, то соответствие между командой M96<> и байтами реакции следующее: M96 Q0 P<> - на число 0 M96 Q1 P<> - на число 1 … M96 Q9 P<> - на число 9 M96 QA P<> - на число 10 … M96 QF P<> - на число 15 M96 QR0 P<> - на число 16 (16+0) M96 QR1 P<> - на число 17 (16+1) … M96 QR E P<> - на число 30 M96 QR F P<> - на число 31
	0.3+	UxW P<time> UxW N<time>	определяет предельное время на приём команды по UART 1 (x=1) или 2 (x=2). <time> записывается в формате HEX после “P” или в формате DEC после “N”. Команды, передача которых по каким-либо причинам занимает больше этого времени, игнорируются как незавершённые.
	0.3+	UxB P<baudrate> UxB N<baudrate>	определяет baudrate для UART 1 (x=1) или 2 (x=2). Value must be written in HEX после “P” или в формате DEC после “N”
	0.7+	UCi <I2C address>	Переключает 2 вывода кнопок на передачу команд по протоколу I2C. для версии I5O8U2 вывод кнопки 3 это SCL и вывод кнопки 4 это SDA <I2C address> - собственный адрес контроллера на шине I2C

			Если в качестве <I2C address> поставить 0, протокол I2C будет отключён и выводы вернуться к режиму работы как кнопки.
	0.7+	UCS	Просканировать шину I2C. Команда выдаёт на USB список адресов (в HEX формате) всех подключённых к шине устройств.
	0.7+	UC T<slave address> H<data in HEX>	Отправляет байты по протоколу I2C на устройство с адресом <slave address> (1 или 2 цифры в формате HEX, в диапазоне 0…0x7F). <data in HEX> - байты, записанные в 16ричном формате, от 00 до FF. (аналогично UxT H<>) Количество отправляемых байт определяется количеством пар цифр до конца строки.
	Работа с файлами		
	0.3+	M23 P<dir> M23	Определяет, в какой подпапке относительно папки “/hcd” читать файлы, начиная со следующего. Если папка не указана, читает файлы непосредственно из папки “0:/hcd/”. Имя папки пишется без символов “/” и пробелов. Разницы между заглавными и строчными символами нет. Можно открыть только папки, непосредственно находящиеся в папке “0:/hcd/”. При вызове этой команды автоматически открывается файл с таким же номером, что и текущий, если файл с таким номером не найден, продолжается чтение ранее открытого файла. Режим чтения файла (Основной или Подпрограмма) не меняется, однако при попытке вернуться к предыдущему файлу контроллер будет искать его уже в новой папке (т.е. это может быть уже другой файл с тем же именем).
	0.3+	F0P <0/1>	Выключает (0) или включает (1) режим Работы с Файлами. В этом режиме нельзя читать программу из файла. При выходе из этого режима последний исполняемый файл анимации открывается заново на случай его изменения. Режимы работы выводов контроллера и чтения файла (Основной или Подпрограмма) сохраняются. При включении в качестве порта вывода автоматически выбирается текущий.
	0.3+	F1P <answers>	В режиме Работы с Файлами выводит содержимое рабочей папки. В остальных режимах выдаёт “NA”. <answers> может принимать следующие значения (можно подряд): “0” - вывести короткое имя первого объекта (файла или папки), “1” - вывести короткое имя следующего объекта (файла или папки) Имена файлов маркируются префиксом “F:”, папок - “D:”.

			Когда все объекты упомянуты, выводит “EOD” (End Of Dir) и переходит в начало списка, так что следующим снова будет упомянут первый объект. “n” - следующая строчка
	0.3+	F2P<dir> F2	Определяет, в какой подпапке относительно папки “/hcd” производить действия с файлами. Если папка не указана, работает непосредственно в папке “0:/hcd/”. Имя папки пишется без символов “/” и пробелов. Разницы между заглавными и строчными символами нет. Можно открыть только папки, непосредственно находящиеся в папке “0:/hcd/”.
	0.3+	F3P<dir>	Создаёт подпапку с именем <dir> в папке “0:/hcd/”, если её ещё не было. Если папка создана или уже существовала, делает её рабочей аналогично команде F2. Имя папки пишется без символов “/” и пробелов. Разницы между заглавными и строчными символами нет. Можно открыть только папки, непосредственно находящиеся в папке “0:/hcd/”.
	0.3+	F4P<file>	Открывает существующий файл <file> в текущей рабочей папке для чтения.
	0.3+	F5P<object>	Удаляет файл или папку (если она пустая) с именем <object> из текущей рабочей папки
	0.3+	F6P<filename>	Сохраняет временный файл в текущей рабочей папке, созданный ранее командой F9, под именем <filename>
	0.3+	F7P<filename>	Переименовывает файл, ранее открытый для чтения командой F4 на <filename> и размещает его в текущей рабочей папке (указанной командами F2 или F3).
	0.3+	F9	Создаёт в текущей рабочей папке временный файл для записи.
	0.3+	F10 P <number> F10 N <number>	Читает из файла, открытого командой F4, <number> байт начиная с того места, на котором остановились ранее. <number> после P записывается в формате HEX, либо, после N, в формате DEC. Строчка возвращается в следующем формате: 8 байт - значение <pointer> в формате HEX, байты в количестве <number>, 4 байта контрольная сумма CRC32, рассчитанная сначала всей присланной строчки.
	0.3+	F11 P <data> F11 H <data>	Записывает данные <data> во временный файл. Если данные записаны после P:

			- Записываются побайтно, за исключением следующих символов: - Вместо символа перевода строки записывается “\r”, символ конца строки заменяется на “\n”, символ обратного слеша (“\”) на “\\”. Если надо вставить другой символ, его можно записать через “\hxx”, где xx - код символа в HEX (например, символ “;” записывается как “\h3B”, а “(“ - “\h28”) - В конце записываются 4 байта CRC32 для данного набора (то есть для всех байт после P). - Если проверка CRC отключена, то 4 последних байта могут быть любыми (они не записываются в память в любом случае). Если данные записаны после H: - Все данные пишутся в HEX формате (номер символа в таблице ASCII), включая CRC32. Исключений, в отличие от записи после P, нет. - Если проверка CRC отключена, то 8 последних символов могут быть любыми цифрами в формате HEX.
	0.3+	F12P<0/1>	Выключает (0) или включает (1) проверку CRC для записываемых байт.
	0.3+	F13P<1,2>	Выбор порта вывода 1 (USART 1) или 2 (USART 2) информации для режима работы с файлами.
	0.7+	F13P<1,2,0>	Значение 0 соответствует общению через USB по протоколу виртуального COM-порта.
	Переменные среды (включая переменные из предыдущих разделов)		
	0.3+	M90 P<seed> M90 N<seed> M90	Устанавливает стартовое значение генератора случайных чисел на <seed> (в HEX после “P” или в формате DEC после “N”). Без параметра - устанавливает стартовое значение генератора случайных чисел на значение по внутреннему таймеру.
	0.5+	Si P<brightness> Si N<brightness> G28 P<brightness> G28 N<brightness>	Устанавливает яркость После “P” от 0 до 1F (в HEX) После “N” от 0 до 31 (в DEC) 0 - минимальная яркость Если нужно указать не саму яркость, а насколько её изменить, то перед значением <brightness> пишется знак “+” или “-” Контроль яркости применяется к очередной функции S<>P<>, а не к текущему изображению.
	0.3+	G29 P <Start frame> G29 N <Start frame>	Указывает, с какого кадра начать вывод в режиме Быстрого видео, <Start frame> (в HEX после “P” или в формате DEC после “N”).

	0.3+	G30 P <Frequency> G30 N <Frequency>	Определяет частоту смены кадров для режима Быстрого видео, P: <Frequency> в диапазоне от 32 до 640 (HEX) Hz. N: <Frequency> в формате DEC и принимает значения от 50 до 1600 Hz.
	0.3+	G31 P <Frame Length> G31 N <Frame Length>	Назначает количество пикселей на кадр для режима Быстрого видео, в формате HEX после “P” или в формате DEC после “N”.
	0.3+	G32 P <Frame number> G32 N <Frame number>	Назначает количество отдельных кадров для режима Быстрого видео, в формате HEX после “P” или в формате DEC после “N”.
	0.3+	G35 P <Frames to play> G35 N <Frames to play>	Указывает, сколько кадров (по кругу) нужно проиграть, в формате HEX после “P” или в формате DEC после “N”. Когда <Frames to play> равно 0, режим Быстрого видео будет повторять кадры до появления команды G36 P0. Иначе будет проиграно указанное количество кадров.
	0.3+	M6 P<LED type>	Устанавливает тип управляемых устройств <LED type> может быть: WS2812 для светодиодов типа WS2812b SK6812 для светодиодов типа SK6812 WK0012 для смеси из WS2812 и SK6812 - может работать нестабильно, так как зависит от допусков по частотам SDSG90 - управление серводвигателями (стандартно для SG90 параметры можно настроить) WS00SD и SK00SD - гибридные режимы для одновременного управления цифровыми светодиодами и аналоговыми устройствами
	0.3+	G43 P<Minimum impulse length> G43 N<Minimum impulse length>	определяет минимальную длину импульса в микросекундах P: параметр в формате HEX для режима серводвигателей. Обычно берётся из спецификации двигателя. 1 секунда это F4240 (1000 000 в DEC) микросекунд. По умолчанию 1F4 (HEX) мкс. N: параметр в формате DEC. 1 секунда это 1000 000 микросекунд. По умолчанию 500 мкс.

	0.3+	G44 P<Maximum impulse length> G44 N<Maximum impulse length>	определяет максимальную длину импульса в микросекундах Обычно берётся из спецификации двигателя. P: в формате HEX) для режима серводвигателей. 1 секунда это F4240 (1000 000 в DEC) микросекунд. По умолчанию 9C4 (HEX) мкс. N: параметр в формате DEC. 1 секунда это 1000 000 микросекунд По умолчанию 2500 мкс
	0.3+	G45 P<Discretization> G45 N<Discretization>	Количество шагов (в формате HEX после “P” или в формате DEC после “N”) на диапазон работы двигателя. Хорошие значения 200 (512 в DEC), B4 (180 в DEC), и т.д. По умолчанию 200 (HEX) или 512 (DEC) шагов.
	0.3+	G50 P<Frequency> G50 N<Frequency>	определяет частоту обновления P: параметр в формате HEX) для режима серводвигателей. По умолчанию 32 (HEX) Hz N: параметр в формате DEC. Обычно в пределах 30-60 Гц По умолчанию 50 Гц
	0.3+	G49	Сбрасывает значения частоты, максимальной и минимальной продолжительности импульса и количества шагов: минимальная длина импульса 0.5 мс максимальная длина импульса 2.5 мс Частота обновления 50 Гц Угол поворота 180 градусов Количество шагов 512 на диапазон работы двигателя
	0.5+	M86 P<0/1>	Переключение между режимами реагирования на кнопки: 0 - реагирование на каждую кнопку отдельно В этом режиме работают все кнопки 1 - реагирование на комбинацию нажатых кнопок В этом режиме работают только кнопки с 1 по 5, остальные, если они есть, отключаются. При этом порядок нажатия не важен. Комбинация определяется исходя из того, какие кнопки были нажаты на момент окончания промежутка времени, определённого командой G27 и начавшегося с момента изменения (нажатия/отпускания) кнопок. Физическое отпускание всех кнопок как комбинация не воспринимается, но можно задать событие, соответствующее такой комбинации и вызываемое исключительно командой G25.

	0.3+	G26 P<delay> G26 N<delay>	Если <delay> не 0, то команда определяет, через какой промежуток времени в миллисекундах зажатая кнопка будет воспринята как повторно нажатая. Это удобно в случае, когда кнопка отвечает за изменение какого-либо параметра (например, положение серводвигателя), так как команда позволяет заменить несколько коротких нажатий одним длительным. Значение записывается в HEX после P, или в DEC после N. Если нужно указать не сам промежуток, а насколько её изменить, то перед значением <delay> пишется знак “+” или “-”. Поскольку кнопки проверяются каждые 10 мс, промежуток времени также будет округлён до десяти в большую сторону
	0.5+	G27 P<delay> G27 N<delay>	Длительность интервала на нажатие комбинации из нескольких кнопок. Работает в режиме, когда выбор команды определяется не конкретной кнопкой, а комбинацией из нажатых. В режиме участвуют кнопки с 1 по 5, а все остальные отключаются. Поскольку в реальных условиях невозможно одновременно зажать несколько кнопок, они будут зажиматься последовательно и данный параметр определяет, через сколько миллисекунд после начала нажатия первой кнопки в комбинации считать готовую комбинацию <delay> записывается в HEX после P, или в DEC после N и может принимать значения в пределах от 20 (0x14) до 10 000 (0x2710) мс Если нужно указать не сам промежуток, а насколько её изменить, то перед значением <delay> пишется знак “+” или “-”. Поскольку кнопки проверяются каждые 10 мс, интервал также будет округлён до десяти в большую сторону
	0.5+	M46 P<Repetitions> M46 N<Repetitions>	Переназначает количество оставшихся повторений текущего файла на значение <Repetitions> (записывается в формате HEX после “P” или в формате DEC после “N”) повторений (включая первое выполнение). Не вызывает повторения файла. В основном имеет смысл при использовании вместе с командой M96 <...>
	0.5+	M91 P<choice> M91 N<choice>	Команда осуществляет заранее определённый выбор (вместо случайного) для команды M98Q<>P<>. Выбор выпадает на файл в позиции <choice> (начинается с 0) при условии, что параметр после Q равен 0 и количество файлов <choice> не больше, чем общее количество файлов в списке (например, если в списке 3 имени файлов, разных или повторяющихся, то <choice> может

			принимать значения 1, 2 или 3). В случаях, когда <choice> окажется больше чем файлов в списке, и когда <choice> равен 0, команда M98 Q0 P<> будет проигнорирована. Значение записывается в HEX после P, или в DEC после N. Если нужно указать не саму выбираемую позицию, а насколько её изменить, то перед значением <choice> пишется знак “+” или “-”.
	0.3+	UxB P<baudrate> UxB N<baudrate>	определяет baudrate для UART 1 (x=1) или 2 (x=2).<baudrate> записывается в формате HEX после “P” или в формате DEC после “N”
	0.3+	UxW P<time> UxW N<time>	определяет предельное время на приём команды по UART 1 (x=1) или 2 (x=2). <time> записывается в формате HEX после “P” или в формате DEC после “N”

4. Спецификация сырых данных

Данные в команде SR<Count>P<raw data> формируются следующим образом.

				Raw data table							
Номер вывода ->				0	1	2	3	4	5	6	7
Биты в каждом байте ->											
	Цвет (bytes)	Цвет (bits)	Номер байта данных								
Цвет первого светодиода	Зелёный	0	1й байт	1-й бит первого байта цвета для светодиода на 0-м выводе							1-й бит первого байта цвета для светодиода на 7-м выводе
		1	2								
		2	3								
		3	4								
		4	5								
		5	6								
		6	7								
		7	8								
	Красный	0	9								
		1	10								
		2	11								
		3	12								
		4	13								
		5	14								
		6	15								
		7	16								
	Синий	0	17								
		1	18								

		2	19								
		3	20								
		4	21								
		5	22								
		6	23								
		7	24	Последний бит первого байта цвета для светодиода на 0-м выводе							Последний первого байта цвета для светодиода на 7-м выводе
…											
N-ный ссвето диод	Зелё ный	0	(N-1)*24+1	1-й бит N-го байта цвета для светодиода на 0-м выводе							1-й бит N-го байта цвета для светодиода на 7-м выводе
		1	(N-1)*24+2								
		2	(N-1)*24+3								
		3	(N-1)*24+4								
	…	…	…								
	Синий	7	(N-1)*24+ 24	24-й бит N-го байта цвета для светодиода на 0-м выводе							24-й бит N-го байта цвета для светодиода на 7-м выводе

5. Стартовый файл сценария и базовые сценарии

При включении контроллер ищет и начинает читать файл .hcd, имя которого начинается на “0000”. В данный файл имеет смысл записывать команды, которые сообщают контроллеру о том, в каком режиме он должен работать (светодиоды, серводвигатели и т.д.), какие кнопки разметить как энкодер а какие отключить и так далее. После конфигурации можно в этом же файле запустить какую-либо анимацию или отправить сообщение на порты ввода-вывода.

В стартовом режиме контроллер управляет светодиодами, все кнопки включены но к ним не привязано никакого действия, порты ввода-вывода включены и имеют следующие настройки:
Бодрейт 128000
Word length = 8 bit
No Parity
1 Stop Bit
Hardware Flow Control = Disabled

В случае, если эти параметры конфигурации устраивают, можно пропустить команды конфигурации и заполнять стартовый файл сценария так же, как и любой другой. Если же требуется изменить параметры, то рекомендуется после конфигурирования вызвать следующий файл сценария и в стартовый файл уже не возвращаться.

Примеры команд, которые можно написать в стартовом файле:

M6Pws2812b; управляем светодиодами спецификации ws2812
M6Psk6812; управляем светодиодами спецификации sk6812
M6Pwk0012; управляем светодиодами сразу двух типов
M6Psdsg90; управляем сервоприводами
M6Pws00sd; управляем светодиодами спецификации ws2812 и сервоприводами
M4P01234; светодиоды на выводах 0…4
M3P567;сервы на выводах 5…7

M45; обнулить счётчик несовпадений
M90; генератор случайных чисел поставить на случайное значение
M91P2; выбирать второй файл сценария из списка в случае такой необходимости

M86 P0; реагировать на кнопки по отдельности
M86Q0P12; кнопки №1 и №2 подсоединены к энкодеру под номером 0
M86 P1; реагировать на комбинации нажатия кнопок
G26N100; выполнять действие по зажатой кнопке каждые 100 миллисекунд
G27N200; выделять на набор комбинации кнопок 200 миллисекунд с момента начала набора комбинации
M25 N1000; команда “стоп” (M25), вновь вызванная через 1 секунду или позже после того, как была применена, запускает чтение файла (как команда “старт”, M24)

G28N31; максимальная яркость
G31N16; размер (ширина) кадра для быстрого видео 16 пикселей на вывод
G30N500; частота вывода кадров в быстром режиме 500 кадров в секунду

G50N50; частота работы ШИМ 50Гц

G43N500; минимальная длина импульса ШИМ 0.5 миллисекунды

G44N3000; максимальная длительность импульса 3 миллисекунды

G45N1024; разрядность (точность) ШИМ - 1024 шага (например, точность позиционирования серводвигателя или количество ступеней яркости светодиодной ленты)

UD0; не выводить сообщения об ошибках на порт ввода-вывода (USB или UART)

Ui0001; назначить данному контроллеру идентификатор “0001”

U2R0; отключить приём данных через UART2

U1F1; контроллер не будет продолжать чтение файла пока не отправит всё, что должен отправить через UART1

U1WN200; любая команда, которую контроллер получит через UART1 должна успеть передаться за 200 миллисекунд

U2S1; UART2 при получении данных должен воспринимать каждый отдельный байт как нажатие на соответствующую реальную или виртуальную кнопку

U2BN9600; установить бодрейт для UART2 на значение 9600

Некоторые примеры сценариев.
Цветной фонарь:
0000.hcd

M96 Q4 G28 P+1;яркость +1

M96 Q5 G28 P-1;яркость -1

G26 N800 ;зажатая кнопка нажимается каждые 800мс

M98 P0500; перейти на файл 0500.hcd

0500.hcd (остальные отличаются только номерами и цветом, указанным в строке с S<>P<>)
В этом файла и в остальных файлах сценариев мы заливаем три матрицы по 64 светодиода конкретным цветом, а также указываем, на какие именно цвета переключаться по каждой из кнопок, то есть какие файлы открыть. На случай, если при работе с фонарём случайно отойдёт контакт одной из матриц и она выключится, повторяем отправку информации каждые 32 мс.

M96Q1 P 0400; при нажатии на первую кнопку перейти в файл 0400

M96Q2 P 0501; то же но в файл 0501

M96Q3 P 0529; то же но в файл 0529

S012P FF0000 >>>>>>> >>>>>>>> >>>>>>>> >>>>>>>> >>>>>>>> >>>>>>>> >>>>>>>> >>>>>>>>; на выводы 0,1 и 2 отправить красный цвет на первые 64 пикселя

G4P20; вывести картинку и подождать 32 мс

M47; повтор файла сначала на всякий случай

Пример файла с анимацией и серводвигателем:

;конфигурация

G9; сброс относительной паузы

M6Pws00sd; светодиоды и сервы

M4P01234; светодиоды на выводах 0…4

M3P567;сервы на выводах 5…7

U1bn9600; бодрейт первого UART 9600

SI P 20; яркость максимальная

M25 N1000; команда “стоп” (M25), вновь вызванная через 1 секунду или позже после того, как была применена, запускает чтение файла (как команда “старт”, M24)

;обычный сценарий, который может быть в любом файле

S0P AA0000 00AA00 0000AA 111111 111100 110011 001111; подготовить цвета светодиодов на 0 выводе

U2T PUART2 connection is working!

U1T PUART1 connection is working!

U0T PUSB connection is working!

G6N 500; вывести текущую картинку на 500 миллисекунде

S0P G g G 000000 G G G 000000 100000 010000 001000 000100 000010 000001 000000

S1P AA0000 00AA00 0000AA 111111 111100 110011 001111

G6N 1500; вывести текущую картинку на 1500 миллисекунде

S0P l>>> >>>> >>>> >>>>; выставить первые 16 светодиодов на выводе 0 на чёрный (выкл)

S1P llll llll; выставить первые 8 светодиодов на выводе 1 на чёрный (выкл)

G6N 3000; вывести текущую картинку на 3000 миллисекунде

G6N 5500;дождаться 5500 миллисекунды

M89 P 0010; войти в подпрограмму 0010.hcd

;далее помигать третьим светодиодом на выводе 0 с частотой 2 раза в секунду и поворачивать серводвигатели на 16 шагов от текущего положения

S0P gg 000000

G0 Q567 P+10; повернуть серводвигатель

G6N 6000

S0P gg 001000

G0 Q567 P+10

G6N 6500

S0P gg 000000

G0 Q567 P+10

G6N 7000

S0P gg 001000

G0 Q567 P+10

G6N 7500

S0P gg 000000

G0 Q567 P+10

G6N 8000

S0P AA0000 00AA00 0000AA

G0 Q567 P0; серводвигатели в начальное положение

G6N 9000

M98 P0025; перейти в файл 0025.hcd

Пример команды подготовки вывода на светодиоды (на 2м выводе проверить что в первом светодиоде цвет отличается от чёрного “?L”, затем перейти к шестому по порядку, он же номер 5, светодиоду “o005”, затем поменять его цвет с цветом светодиода номер два на 6м выводе “S6002”, затем пропустить два светодиода “g g”, затем вывести красный цвет “BB 00 00”, и повторить его 3 раза “>>>”):

S2P ?L o005 S6002 gg BB0000 >>>

6. Связь по UART и USB (для версии 0.7+)

Версии DKLed 0.3 и последующие способны общаться друг с другом и с периферией по протоколу UART.

Контроллер игнорирует любую строчку, которая начинается не с его ID (определяется командой **UI <ID>**). Строго 4 символа.

Вместо любого символа ID можно вставить символ “*” (звёздочка). “*” означает, что на этом месте может быть любой символ, кроме символов окончания строки и комментариев (символы с кодом 0x0A, 0x0D, 0x00, и символы “;”, “(“ и “)”). К примеру, если команда начинается с ****, такая команда будет принята любым DKLed, а если с 12**, то всеми DKLeds, чьи ID начинаются на “12”.

Следует помнить, что контроллер воспринимает любой поток данных как потенциальную команду. И если строчка не была распознана как адресованная контроллеру, он по-прежнему ждёт окончания команды и становится готов воспринимать строчку только после того, как получит символ с кодом 0x0D (возврат каретки). В случае, если нет уверенности, что все строчки на шине данных начинаются с ID и заканчиваются 0x0D, разумно начинать передачу каждой новой команды с 0x0D и потом ID.

Контроллер понимает любые пришедшие команды в соответствии со своей версией прошивки.

Чтобы обозначить, что команда закончена, следует отправить символ с кодом 0x0D (возврат каретки). В версии 0.7+ символом окончания команды являются также символы 0x0A (перевод строки), 0x00 (пустой символ) и символ начала комментариев “;” - то есть все те же символы, что работают при чтении файла. После получения этого символа контроллер приступит к выполнению команды.

В версии 0.7+ символ “(“ (открывающая скобка) приостанавливает чтение поступающих символов до того момента, как среди них встретится символ “)” (закрывающая скобка). Таким образом в скобки можно заключить последовательность символов, которые должны быть проигнорированы при чтении команды через UART - игнорируются все символы между скобками, включая символы окончания строки и комментариев. Такая возможность введена на случай, если для синхронизации нескольких устройств используются передатчики, которые передают данные небольшими пакетами и сопровождают их дополнительной информацией.

DKLed исполняет команды по очереди в порядке их поступления по UART, USB или из файла. Поскольку длинные команды могут потребовать нескольких циклов для приёма, и, в случае неполадок со связью могут оказаться незаконченными, введён предел по времени на этот процесс. По умолчанию, если команда не была полностью получена в течение 1 секунды с момента начала её приёма, она прерывается. Это время можно изменить командой **UxW <>**.

Для примера, если ранее была вызвана команда **UI 1234**, следующие команды будут прочитаны и исполнены ($0D как символ с кодом 0x0D):

```
**** S0 P 001000$0D
1234 G9 $0D
**2* S4 P 000000 g g 001010 00AF0B 000000 GGGG 0A0A0A ffffff $0D
```

эти - проигнорированы:

```
2546 G4яя
123яяя
tx=2;
```

а эта будет ожидать, когда она будет окончена, или когда закончится время на её приём:

```
**** S2P 023450 001100 00
```

Стоит использовать команду **UxT P<>** сразу после **G5** или **G7**, поскольку в этом случае она будет выполнена наиболее близко к соответствующему моменту времени.

Если несколько контроллеров соединены последовательно, то чтобы передать команду контроллеру, расположенному дальше по цепочке, можно использовать следующую конструкцию (для примера предположим, что выход номер 2 контроллера 0001 соединён со входом 2 контроллера 0011, а выход 1 контроллера 0011 соединён со входом 1 контроллера 0021):

```
U2T P;0011 U1T P\\h0D0021 G0 \\h0D;
; отправить через UART 2 строчку ";0011 U1T P\\h0D0021 G0 \\h0D;"
; то есть: сброс команды (";"),
; адрес принимающего контроллера 0011
; (обратите внимание на отсутствие пробела перед адресом)
; команда отправить текст через порт 1 U1T P
; текст "\h0D0021 G0 \h0D" (обратите внимание, что "\\" превратилось в "\")
; при отправке текста будет применено форматирование,
; так что будет отправлено  $0D0021 G0 $0D ($0D как символ с кодом 0x0D)
; что будет означать "контроллеру 0021 выполнить команду G0"
; указание контроллеру 0011 что команда завершена  (";")
```

Обратите внимание, как при выполнении команды исходным контроллером текст "\\h0D" был преобразован в "\h0D" так как макрос "\\" означает вставить символ "\". А при выполнении команды во втором контроллере этот текст был преобразован в символ 0x0D, поскольку макрос "\hxx" в тексте команды означает "вставить символ с ASCII номером xx". Если бы нам нужно было передать символ 0x0D не через один, а через два контроллера, то пришлось бы писать "\\\\h0D":

```
U2T P\h0D0011 U1T P\\h0D0021 U2T P\\\\h0D0031 G0  \\\\h0D  \\h0D\h0D
```

ВАЖНО! Команда **SR Q<> P<>** не может быть принята через внешние порты, так как может содержать любую последовательность символов и потому может вызвать конфликт значений переменных.

Бодрейт каждого UART можно настроить (в начале работы это 128000).
Другие параметры:
Word length = 8 bit
No Parity
1 Stop Bit
Hardware Flow Control = Disabled

Управление через USB.

При подключении контроллера версии 0.7 и выше к персональному компьютеру он определяется как COM-порт. Для адекватной работы с ним в системе (Windows до 7 версии, старые версии Линукс и МакОс) должен быть установлен Virtual COM Port драйвер от STM. В современных ОС драйверы уже имеются.

Команды вводятся в том же виде, в котором они записываются в файлах сценариев, вводить ID, как в случае с отправкой команды через USART не нужно. Однако в конце каждой команды нужно ввести символ с кодом 0x0D (возврат каретки), либо 0x0A (перевод строки), 0x00 (пустой символ) или один из символов начала комментариев - то есть все те же символы, что работают при чтении файла, как сигнал для исполнения команды. В противном случае контроллер будет ожидать завершения команды. Можно вводить несколько команд подряд, разделяя их символами, перечисленными выше (0x0D, 0x0A и т.д.).

Поскольку команды ввода информации в файл F11P<> и отправки теста через UART UxTP<> воспринимают символы комментариев как часть данных, для их завершения нужно использовать только 0x0D, 0x0A или 0x00.
Например ($0D как символ с кодом 0x0D, символ “;” - символ комментария, отличия выделены цветом):

U0T Pline1**_;**U0T Pline2 $0D вернёт строку “line1**_;U0T P**line2 “
U0T Pline1**_$0D**U0T Pline2 $0D вернёт строку “line1**_**line2 “

Если контроллеру недостаточно тока, он может перестать отправлять данные через USB. Такое возможно, когда общая яркость светодиодов достаточно высока, а питание осуществляется от того же USB. При этом контроллер продолжает получать и исполнять команды.

7. Синхронизация нескольких контроллеров

Команды **G8** и **G9** удобны для синхронизации нескольких контроллеров. Команды **M47** и **M98** также можно использовать с этой целью, однако они не такие гибкие.
На практике каждый контроллер запускается независимо,так что если нужно, чтобы они работали одновременно, требуется внешнее прерывание или внешние часы. Для этого в версии 0.3 и последующих следует передавать ведомым контроллерам следующие команды ($0D как символ с кодом 0x0D):

****** M47 $0D**
Одновременно начать читать текущий файл анимации сначала.
Это хорошо работает в случае коротких повторяющихся анимаций, поскольку вероятность прерываний извне за время проигрывания анимации невысока.

****** M98 P … $0D**
Одновременно открыть соответствующий файл из памяти. Не очень гибкая команда в условиях, когда каждый контроллер работает по сценарию со множеством вариаций или интерактива.

****** G9 $0D**
Если в анимации используются относительные паузы (**G4** и **G5**), возможно время от времени делать длинную паузу и и запускать продолжение чтения файлов одновременно. В результате контроллеры, которые, по каким либо причинам отработали быстрее чем надо, подождут нужного момента времени. Однако контроллер, который оказался в момент прерывания по **G9** на одном из предыдущих кадров, в результате может значительно отстать, особенно если паузы между кадрами малы.

****** G8 P … $0D**
Если в анимации используются абсолютные тайминги (**G6** и **G7**), можно указывать ведомым контроллерам в каком именно моменте времени от начала чтения файла они должны находиться. Тогда, если контроллер оказался несколько быстрее, чем надо, он притормозит, а если медленнее - то проиграет часть кадров с максимальной скоростью - и синхронизируется с остальными.

Если управляющий контроллер не только синхронизирует остальные, но и сам выполняет какие либо операции, стоит использовать команду “**UxF 1**” до отправки таймингов, и также указывать тайминг для себя сразу после передачи чтобы уменьшить возможный временной лаг, вызванный средствами связи. Например:

```
…
U1F 1
…
U1T P**** G8P200\r
G8P200
…
```

При соединении нескольких контроллеров между собой может потребоваться создать конфигурацию, в которой один или оба входа контроллера (USART RX1 и/или RX2) окажутся соединены с каким-либо выходом этого же контроллера (USART TX1, или TX2). Тогда, чтобы контроллер не реагировал на собственные команды перед отправкой байт следует отключать приём командой **U1R0** для RX1 и **U2R0**

для RX2. А после отправки, если нужно, включать их через **U1R1** и **U2R1** соответственно. Однако следует понимать, что контроллер не проверяет, занята ли линия при отправке данных. Так что подобные конфигурации будут стабильно работать только если сценарии построены так, что два контроллера не будут одновременно отправлять данные по этому соединению (то есть не возникнет зашумления команд).

8. Управление серводвигателями и аналоговыми устройствами

Версии DKLed 0.3 и последующие способны одновременно управлять несколькими сервомоторами. Для перехода в соответствующий режим используется команда **M6 P SDSG90**.

Положение как каждого сервомотора по отдельности, так и всех сервомоторов вместе определяется через команду **G0**, и должно быть не более чем количество шагов на рабочий диапазон (512 по умолчанию). Если оно будет больше, то будет уменьшено до максимального значения в диапазоне. Двигатель принимает положение исходя из своего рабочего диапазона и того, сколько задано шагов в процентах от максимального. К примеру, для рабочего диапазона в 180 и количества шагов в 512, значение 256 переместит вал двигателя в положение 90 градусов, а 128 - в 45 градусов. Однако стоит помнить, что количество шагов по умолчаниию и точность позиционирования двигателей может не совпадать, из-за чего возможны небольшие ошибки в позиционировании двигателя, в пределах аппаратной погрешности.

Положение серводвигателей изменяется (при необходимости) по команде **G4** или **G6**. Чтобы включить/выключить способность серводвигателей реагировать как только их параметры изменятся, используется команда **G1**. При смене режимов работы контроллера командой **M6** эта настройка выключается. Эта настройка полезна в случаях, когда положения двигателей должны меняться со скоростями, близкими к скорости включения/выключения управляющих импульсов и тем самым могут влиять на интерпретацию импульсов двигателями.

M5 отключает двигатели от питания, позволяя им вращаться свободно. Если команда применена без параметров, то двигатели, которые не были отключены индивидуально, будут включены по коменде **G4** или **G6**.
Если нужно отключить отдельные серводвигатели, используется команда, **M5 P <servo output>**. Чтобы вновь включить двигатель, используется команда **M3** без параметров (для всех двигателей одновременно) или с параметром (**M3 P<>**) для указания конкретного двигателя.

Параметры работы серводвигателей по умолчанию:

- Минимальная длительность импульса 0.5 мс
- Максимальная длительность импульса 2.5 мс
- Частота генерации импульсов 50 Hz
- Угол поворота (рабочий диапазон) 180 градусов
- 512 шагов на рабочий диапазон

Эти параметры можно менять с помощью команд **G43, G44, G45** и **G50** и вернуть к исходным настройкам с помощью команды **G49**.

Если это необходимо, можно включить или выключить напряжение на отдельных выводах.

- **M10** (без параметров) включает все выводы.
- **M10 P<>** включает явно указанные выводы.
- **M11** (без параметров) выключает все выводы.

M11 P<> выключает явно указанные выводы.

Состояние выводов, определённое командами **M10 и M11,** изменится как только соответствующие выводы будут использованы другими командами. Режим, в котором эти выводы будут поддерживать состояние, маркируется как “N” если обратиться к контроллеру через команду **UxA<>**.

9. Гибридный режим

В гибридном режиме можно одновременно управлять сервомоторами и адресными светодиодами, такими как WS2812, а также включать и отключать постоянный ток на отдельных выводах. Соответственно каждый вывод можно настроить на управление соответствующим устройством. Для того, чтобы использовать этот режим, нужно вызвать команду **M6 P WS00SD или M6 P SK00SD**.

Гибридный режим подходит для медленных анимаций, поскольку все выводы обновляются с частотой, на которую настроены сервомоторы, обычно 50-100 Гц. Также важно убедиться, что выводам сервомоторов соответствует пустая анимация, где все цвета равно **000000,** при необходимости этого можно добиться вводом команды **Sx P 000000 000000 ...000000**. Если этого не сделать, сервомоторы могут работать с ошибками или оказаться неспособными достичь тех или иных положений.

Следующие команды работают вот так:

M4 (без параметров) настраивает все выводы на работу со светодиодами.

M4 P<> настраивает явно указанные выводы на работу со светодиодами.

M3 (без параметров) настраивает все выводы на работу с серводвигателями.

M3 P<> настраивает явно указанные выводы на работу с серводвигателями.

M5 P<> немедленно останавливает работу указанных выводов, если они ранее помечены как серводвигательные. Однако эти выводы не перенастраиваются на что либо другое.

M10 и M11 работают так же, как и в серво режиме.

10. Режим Быстрого видео

Режим быстрого видео разработан для вывода цветов на светодиоды настолько быстро, насколько это возможно. Для этого контроллер загружает анимацию в память целиком и проигрывает её по кругу, пока не будет проиграно нужное количество кадров или пока его не остановят извне или изнутри специальной командой. Обычно данный режим нацелен на вывод, так что длинные команды могут быть неэффективны или их выполнение может тормозить. Также общий размер анимации (общее количество пикселей во всех кадрах для каждого вывода) зависит от размера памяти в контроллере и не может быть большим.

Кадры вводятся посредством команд **Sx P<>** или **SR Q<> P<>** единым потоком. То есть, если были введены цвета для 20 светодиодов как на 1 так и на 5 выводы в обычном режиме работы, а размер кадра указан в 10 пикселей, цвета с 0 по 9 будут цветами для соответствующих же светодиодов в первом кадре, а с 10 по 19 - для тех же самых светодиодов с 0 по 9, но уже во втором кадре.

В этот режим можно перейти только из режима управления светодиодами, посредством команды **G36 P1**. Эта команда только подготавливает вывод, но не запускает его. Фактический запуск анимации осуществляется через команды **G4** или **G6**. Если количество проигрываемых кадров, указанное ранее в команде **G35,** больше 0, то контроллер автоматически выйдет из этого режима в обычный как только соответствующее количество кадров будет проиграно. При этом контроллер прекратит чтение файла на время проигрывания анимации, но можно остановить этот режим вручную или извне командой **G36 P0**.

Если же количество кадров в **G35** было 0, остановка возможна только командой **G36 P0**, но в этом случае контроллер продолжит чтение файла параллельно с проигрыванием анимации, хотя, при высоких частотах, будет делать это медленнее. При смене режимов работы контроллера командой **M6** этот режим также отключается.

Как и в обычном режиме управления светодиодами, все выводы обновляются одновременно. Так что бывает полезно разбить кадр на отдельные участки, каждый из которых будет выводится по своему выводу, чтобы уменьшить количество пикселей на вывод на кадр и увеличить доступное для хранения в памяти количество кадров и частоту обновления картинки.

Параметры назначаются следующими командами:

G29 определяет, с какого кадра начать проигрывать анимацию. Не может быть больше, чем общее количество кадров

G30 определяет частоту кадров в диапазоне от 50 до 1600 Гц. Фактическая максимальная частота зависит от количества пикселей на каждом выводе на кадр, и если 16 пикселей позволяют проигрывать анимацию с частотой в 1600 Гц, то 32 пикселя - только 800 Hz

G31 определяет количество пикселей на каждом выводе на кадр.

G32 определяет количество кадров в памяти. Кадры проигрываются по кругу.

При выполнении все эти команды учитывают ограничения контроллера и управляемых светодиодов, а именно частоты работы и объём памяти. По этому каждая команда проверяет остальные параметры на соответствие и подстраивает их при необходимости.

В версии 0.5 и выше команды **G35** и **G36** можно вызывать по нажатию кнопок. В случае, когда на кнопку закодирована команда **G36 P0**, нажатие (отпускание) немедленно останавливает проигрывание Быстрого видео. Когда на кнопку закодирована команда **G36 P1**, нажатие (отпускание) запускает анимацию, не дожидаясь команд **G4** или **G6**. Количество проигрываемых кадров при этом регулируется ранее встретившейся командой **G35**. Если же на кнопку закодирована команда **G35** с соответствующим параметром, то происходит немедленное проигрывание указанного в ней числа кадров.

11. Режим работы с файлами

Режим работы с файлами разработан для возможности читать и записывать анимации (и другие файлы) на карту памяти контроллера без необходимости её вынимания и переставления. Файлы читаются и записываются по тому же каналу связи, что и команды управления. Это может быть полезно в случае, если доступ к контроллеру затруднён.

Для перехода в режим работы с файлами используется команда **F0P1**. Это единственная команда этого режима, которая может работать когда контроллер читает файлы анимации. В режиме работы с файлами чтение файлов анимаций приостанавливается, а после выхода из этого режима проигрываемый файл запускается сначала на случай, если он был изменён.

В режиме работы с файлами доступны папки hcd/ и подпапки в ней. В этом режиме можно получить информацию о содержимом папки, перейти из одной папки в другую, открыть любой файл для чтения, создать файл для записи и затем, после записи в него данных, сохранить его под любым свободным именем в формате 8.3 (8 знаков имени и 3 знака расширения, все буквы заглавные, без пробелов), удалить файл или пустую папку (папку, в которой есть что-либо, удалить нельзя), переименовать или переместить файл.

Чтобы переместить или переименовать файл (команда **F7**), сначала требуется открыть его для чтения командой **F4**. Затем в команде **F7** указывается новое имя. Если при этом нужно переместить файл в другую папку, её необходимо сделать активной командами **F2** или **F3** после открытия файла, и затем применить переименование. Новое имя при этом может совпадать со старым, но оно должно указываться обязательно. Если после этого нужно прочитать что-либо из этого файла, его надо открыть заново.

Команды **F2** (открыть существующую папку) или **F3** (создать новую папку) определяют, в какой папке будут производится действия по созданию, чтению, записи и удалению файлов, но не влияют на то, из какой папки контроллер будет читать анимации для их исполнения.

При чтении данных (команда **F10**), данные записываются в следующем формате:

8 байт - указатель на позицию в файле (номер байта с начала файла), с которого было начато чтение, в формате HEX.

Байты в количестве, указанном в параметре команды **F10**

Контрольная сумма CRC32, рассчитанная для всех переданных байт, включая указатель.

Если файл закончился, то выводится строка “END OF FILE” без указателей и CRC.

Для записи данных в память необходимо создать новый временный файл командой **F9**.

При записи данных (команда **F11**) не требуется указывать количество байт, однако в конце необходимо указать CRC32 для отправленных байт. CRC32 рассчитывается только для тех байт, которые отправляются непосредственно для записи. CRC32 нужно отправлять после данных даже в том случае, если его проверка отключена, однако его можно не рассчитывать и написать любое удобное значение. Он отправляется в том же виде, что и остальные данные - в байтах или в HEX кодах.
Если данные отправляются в байтах, то, если нужно использовать спецсимволы перевода строки или начала новой строки, надо записывать “\r” (поскольку символ перевода строки является командой на исполнение полученной строки), и “\n” соответственно, а символ “\” записывать в виде “\\”. Если надо вставить другой символ по номеру его кода, его можно записать через “\hxx”, где xx - код символа в HEX
Такое форматирование строки нужно делать перед расчётом CRC32 для отправляемого пакета данных.
Если запись была удачной, контроллер ответит “Data saved”.

Алгоритм расчёта CRC32 в контроллере:

```
crc = 0;
For (i = 0; i<length; i++){
        v = DATA[i];
        crc ^= (v << 24);
        for (j = 0; j < 8; j++){
                if ((crc & 0x80000000) != 0) {
                        crc = ((crc << 1) ^ 0x04C11DB7);
                } else {
                        crc <<= 1;
                }
        }
}
return crc;
```

Для удобства хранения нескольких проектов в одном месте есть возможность создавать подпапки в рабочей папке /hcd/. Для того, чтобы контроллер начал читать файлы из одной из них, используется команда **M23**. После параметра P пишется имя подпапки, в которой контроллер будет искать следующий файл. Если параметра не указано, то контроллер будет искать следующий файл в папке /hcd/. Эту команду можно использовать в любом режиме работы контроллера:

```
M23 P Dir1; начинает читать файл с тем же именем
;и в том же режиме, что и текущий читаемый,
;но из папки 0:/hcd/DIR1/
;(обратите внимание, что все буквы в названии папки заглавные, а пробелы отсутствуют)
;следующие файлы ищет тоже в этой папке

M23; возвращается в папку  0:/hcd/
;начинает читать файл с тем же именем, что и текущий
;следующие файлы также ищет в папке 0:/hcd
```

При этом команда **M23** только из какой папки контроллер будет читать анимации для исполнения, и не влияет на то, в какой папке производятся действия команд **F1 - F13**.

12. Обработка перехода на следующий файл сценария

Для того, чтобы перейти на следующий файл сценария, используются команды M98

Простой переход:

M98 P xxxx

Когда в коде встречается эта команда, контроллер сохраняет текущее внутреннее время и переходит в файл с именем “xxxx”, при условии, что этот файл найден в папке. Если в качестве xxxx указывается число FFFF, то это сигнал что надо открыть файл, который проигрывали перед текущим. Такой вариант нужен, например, когда нужно переключаться между двумя файлами, один из которых зависит от общего сценария работы, а второй одинаков для всех сценариев. В частности это может быть ситуация, когда разные анимации должны перед повтором заканчиваться одинаковыми картинками или набором команд на внешние устройства. Или когда разные анимации должны в конце вызвать одну из нескольких случайных реакций а затем начать проигрываться сначала.

Переход на случайный файл (версия 0.3+):

M98 Q<n> P xxxx xxxx … xxxx

В версии 0.3 осуществляет переход на случайный файл из списка (**размер списка ограничен, не более 200 файлов**), выбор которого осуществляется из <n> вариантов с помощью генератора случайных чисел. Используется когда нужно разнообразить сценарии. Если же случится, что списка для того, чтобы осуществить выбор, недостаточно (например в списке всего 5 файлов, а <n> равно 10 и при этом выбран случайный номер 7), то контроллер выдаёт ошибку и продолжает читать тот же файл дальше.

Версия 0.5 и выше обрабатывает ситуацию точно так же, только случай, когда случайный номер больше, чем файлов в списке не воспринимается, как ошибка, а как шанс на продолжение чтения файла.

Таким образом можно делать следующие конструкции:

…
M98 Q10 P 0001 0002 0003 0004 0005 (случайный выбор одного из пяти файлов либо продолжение программы в половине случаев)
…
M47 P6 (повторить файл 6 раз, учитывая что в половине повторов мы переходим на другой файл, маловероятно что мы сделаем все повторы)
<ещё какая-то часть сценария> (вероятность что мы дочитаем до этого куска файла совсем мала)

Переход на заранее выбранный файл (версия 0.5+):

M98 Q0 P xxxx xxxx … xxxx

В этом случае осуществляется переход на файл, номер которого в списке (с первого, но **размер списка ограничен, не более 200 файлов**) выбран ранее через команду **M91 P <номер>**. Полезна, когда надо сделать несколько сложных веток сценариев, или выбор следующей анимации по кнопкам, поскольку команда **M91** может быть определена в другом файле ранее или изменена по нажатию на кнопку. Самый простой вариант такого сценария:

Выбрать файл №1 из списка
Открыть и запустить файл со списком
Запустить его
Изменить выбор с первого на второй
Вернуться к файлу со списком
Выбрать и запустить второй файл из списка (который отличается от первого)
Изменить выбор на следующий в списке
Вернуться к файлу со списком
Выбрать и запустить очередной (третий) файл из списка

Далее, пока у нас не кончился список запускаем в заранее определённом порядке эти три файла (к примеру 1 2 3 1 3 3 2 1 2 2 2 2 3…).

Вот как может выглядеть код (сценарий бегущей строки):

0000.hcd:

…
M91 P0 (поскольку мы увеличиваем его перед перебором, то чтобы не пропустить первый файл в списке тут ставим 0)
M98 P0001

0001.hcd:

(файл списка, следующий список, 0002.hcd такой же, только порядок файлов в M98 может быть другим)
…
M91 P+1
M98 Q0 P 0010 0020 0050 0010 0020 0030 0030 0060 0010 0020 …
(мы дошли сюда поскольку счётчик вышел за границы списка)
M91 P0 (сбрасываем счётчик файлов для следующего списка, так как в одном списке может быть не более 200 файлов)
M98 P0002 (Следующий список)

0010.hcd:

(то же самое будет и в 0020.hcd, 0030.hcd и так далее
(вывод на светодиоды и двигатели того, что нам нужно, например, сдвиг по матрице светодиодов ранее нарисованных букв и рисование новой буквы, своей для каждого файла)
…
M98 P FFFF (возврат к актуальному файлу списка)

Переход по кнопкам (версия 0.5+):

Если команда **M98** вызвана из кнопки

M96 Q(R) <кнопка> M98 P xxxx
M96 Q(R) <кнопка> M98 Q <n> P xxxx xxxx … xxxx
M96 Q(R) <кнопка> M98 Q 0 P xxxx xxxx … xxxx

То контроллер осуществляет выбор в соответствии с тем, как описана команда **M98** (то есть конкретный файл, случайный или очередной из списка), однако переход на этой файл осуществляется по нажатию на кнопку.

Выбор случайного или очередного файла из списка осуществляется в момент вызова команды **M96**, а не в момент нажатия на кнопку. Соответственно если позднее была вызвана (по другой кнопке или непосредственно из файла) команда **M91 P<>**, это не повлияет на уже выбранный кнопкой файл.

Если в версии 0.3 после команды **M96 Q(R) <кнопка> P<файл>** вызвать команду **M97 Q(R) <кнопка> P 2** (в версии 0.5+ это **M96 Q(R) <кнопка> L<файл>**), кнопка будет осуществлять отложенный переход, то есть даст файлу завершиться (вернее дойти до любой ближайшей команды **M2**, **M47** или **M98**), а затем контроллер начнёт читать файл, который определён в кнопке, назависимо от того, какие параметры указаны в командах **M47** или **M98** в файле. И файл, переход из которого был осуществлён по кнопке, становится предыдущим.

Это важно понимать при работе со сценариями, в которых необходимо часто возвращаться к предыдущим файлам (как в примере с бегущей строкой в предыдущем разделе), поскольку переходы по кнопкам могут нарушать ход подобных сценариев. В случае с бегущей строкой переход по кнопкам хорошо сработает, если нужно запустить другой текст, оборвав текущий на полуслове, и плохо сработает, если надо всего-лишь вставить дополнительный символ.

Если необходимо время от времени возвращаться к тому или иному файлу списка, функцию возврата к нему можно записать в одну из виртуальных кнопок, и вместо вызова **M98 P FFFF** осуществлять виртуальное нажатие на эту самую кнопку.

Использование связки отложенного перехода по кнопкам **M97 Q(R) <кнопка> P 2** (в версии 0.5+ это **M96 Q(R) <кнопка> L<файл>**) и команды **M2** полезно, если сценарий состоит из частей, по окончании каждой из которых необходимо или перейти куда-то, или продолжить, даже если кнопка перехода была нажата в середине исполнения соответствующей части. То же самое можно сделать, если каждую из этих частей поместить в свой файл. Однако при этом может потеряться информация о том, какой файл был предыдущим, а также время на поиск начала следующего файла может слегка задержать вывод первого кадра в очередной части сценария.

Также при переходах на другие файлы следует учитывать, исполняется файл в Основном режиме или в режиме Подпрограммы.

13. Режим Подпрограммы

В версии 0.5+ отдельные ветки сценариев можно вызывать в режиме Подпрограммы. В этом режиме ветка сценария исполняется посреди вызвавшего её файла, и после того, как её выполнение закончено, вызвавший эту ветку сценария файл продолжает исполняться дальше, с того места, в котором он её вызывал.

Этот режим существует для того, чтобы можно было составлять сложные но постоянно повторяющиеся операции (например, сдвиг пикселей для бегущей строки), которые не требуют от контроллера наибыстрейшего исполнения. Либо чтобы делать комплексные макросы для кнопок, которые при этом не будут прерывать основную анимацию.

Чтобы войти в этот режим выполнения сценария используется команда **M89 P<>**. После её вызова все переходы между файлами по кнопкам или командам **M47<>**, **M89<>** и **M88<>** происходят с сохранением этого режима, **M98<>** выводит из Подпрограммы в начало указанного в ней файла. Это следует учитывать в случае, если режим Подпрограммы включается для выполнения большого количества относительно длительных операций.

Выход из режима Подпрограммы осуществляется командой **M89 без параметров**. При этом контроллер возвращается в тот файл (в таблице команд он обозначен как Основной файл), из которого режим Подпрограммы был вызван, и продолжает его читать со следующей, после вызова режима Подпрограммы, строки. Время отсчёта абсолютных пауз **G6** и **G7** при этом не сдвигается, тогда как паузы по командам **G4** и **G5** сбрасываются.

Если во время работы режима Подпрограммы исполняется последовательность файлов и нужно по кнопке вызвать предыдущий файл в этой последовательности, надо использовать конструкцию:

M96 Q<button> M89 P FFFF

Или

M96 QR<button> M89 P FFFF

Если же необходимо записать в кнопку номер файла, который исполнялся перед Основным, то используется конструкция:

M96 Q<button> M98 P FFFF

Или

M96 QR<button> M98 P FFFF

Если нужно по кнопке вызвать предыдущий файл, независимо от того, в каком режиме контроллер читает текущий, то используется конструкция:

M96 Q<button> M88 P FFFF

Или

M96 QR<button> M88 P FFFF

Эту возможность можно использовать совместно с **G25 <>** для контролируемого перехода в сложных сценариях с запутанными Подпрограммами.

Также следует учитывать, что в режиме Подпрограммы команда **M88 P FFFF** и **M89 P FFFF** приведут к одинаковому результату - запуску в режиме Подпрограммы файла, предыдущего текущему, с самого начала.

Команда **M98 P FFFF** вызовет исполнение предыдущего файла с самого начала (но при этом осуществить возврат к строке вызова уже не получится),
Команда **M89 P FFFF** вызовет исполнение предыдущего файла уже в режиме Подпрограммы, то есть станет возможным возврат с той строчке из которой файл был вызван.

Команда **M89**, вызванная в Основном режиме исполнения файла без параметров, игнорируется, **M88<>** ведёт себя так же, как и **M98<>**. По-этому **M89** можно ставить перед командами **M47<>** или **M98<>**, если нужно чтобы файл мог исполняться как Подпрограмма и потом возвращаться в Основной режим к месту вызова.

В каком режиме будет открыт очередной файл сценария при наступлении того или иного события вкратце показано в таблице 12.1. Розовым цветом обозначено, что файл будет открыт в Основном режиме и позиция, с которой должен будет читаться предыдущий Основной файл при возврате сбрасывается на начало файла, голубым - что файл будет открыт как подпрограмма, а позиция в Основном файле, который вызвал переход в этот режим сохраняется. Белый - отсутствие действия, либо действие вызовет тот режим, который в каждом конкретном случае соответствует конкретной команде из перечисленных.

Типичный пример использования подпрограммы:

Основной файл:

```
….
M89 P0100 ;вызов файла 0100.hcd как подпрограммы
; продолжение исполнения файла
…..
```

Либо:

```
….
M96 Q2 i 0100 ; (оно же M96 Q2 M89 P0100)
;по нажатию на кнопку 2 вызвать файл 0100.hcd как подпрограмму
….
```

Подпрограмма 0100.hcd:

```
;мы что-то делаем:
;рисуем картинку командой SxP<>
;перемещаем цвета пикселей посредством макросов в SxP<>
;отправляем данные по USART через UxT<>, UxA
;изменяем положения двигателей
M89 ;возвращаемся в Основной файл
```

Таблица 13.1 Режимы перехода по событиям

	Чтение Основного файла	Чтение Подпрограммы
M88	M47 (Повторить файл сначала или отреагировать на нажатие кнопки (M98 Q(R)<> L<>)	M47 (Повторить файл сначала или отреагировать на нажатие кнопки (M98 Q(R)<> L<>)
M89	-	Вернуться в Основной файл в момент перехода в Подпрограмму
M98	M47 (Повторить файл сначала или отреагировать на нажатие кнопки (M98 Q(R)<> L<>)	M47 (Повторить файл сначала или отреагировать на нажатие кнопки (M98 Q(R)<> L<>)
M47	Повторить файл сначала или отреагировать на нажатие кнопки (M98 Q(R)<> L<>)	Повторить файл сначала или отреагировать на нажатие кнопки (M98 Q(R)<> L<>
M88 P <>	Переход к файлу, чтение в **Основном режиме**, сначала	Переход к файлу, чтение в **режиме Подпрограммы**, сначала
M89 P <>	Переход к файлу, чтение в **режиме Подпрограммы**, сначала	Переход к файлу, чтение в **режиме Подпрограммы**, сначала
M98 P <>	Переход к файлу, чтение в **Основном режиме**, сначала	Переход к файлу, чтение в **Основном режиме**, сначала
M88 P FFFF	Переход к предыдущему файлу, чтение в **Основном режиме**	Переход к предыдущему файлу, чтение в **режиме Подпрограммы**
M89 P FFFF	Переход к предыдущему файлу, чтение в **режиме Подпрограммы**	Переход к предыдущему файлу, чтение в **режиме Подпрограммы**
M98 P FFFF	Переход к предыдущему файлу, чтение в **Основном режиме**	Переход к файлу, предшествовавшему Основному, из которого вызвана Подпрограмма, чтение в **Основном режиме** сначала
M97 Q(R)<button> M88	Вызвать M47 по кнопке	Вызвать M47 по кнопке
M97 Q(R)<button> M89	-	По кнопке вернуться в Основной файл в момент перехода в Подпрограмму
M97 Q(R)<button> M98	Вызвать M47 по кнопке	Вызвать M47 по кнопке
M97 Q(R)<button> M88 P FFFF	По кнопке вызвать предыдущий файл в **Основном режиме**	По кнопке вызвать предыдущий файл в **режиме Подпрограммы**
M97 Q(R)<button> M89 P FFFF	По кнопке вызвать предыдущий файл в **режиме Подпрограммы**	По кнопке вызвать предыдущий файл в **режиме Подпрограммы**

M97 Q(R)<button> M98 P FFFF	По кнопке вызвать предыдущий файл в **Основном режиме**	По кнопке вызвать файл, предшествовавшему Основному, в **Основном режиме**
M97 Q(R)<button> P xxxx	По кнопке вызвать файл, чтение в **Основном режиме**, сначала	По кнопке вызвать файл, чтение в **Основном режиме**, сначала
M97 Q(R)<button> P FFFF M97 Q(R)<button> P	По кнопке вызвать предыдущий файл в **Основном режиме**	По кнопке вызвать файл, предшествовавшему Основному, в **Основном режиме**
M97 Q(R)<button> L xxxx	По кнопке запланировать файл - подставить его значение в очередную команду **M47, M88 P <>, M89 P <>** или **M98 P <>**	По кнопке запланировать файл - подставить его значение в очередную команду **M47, M88 P <>, M89 P <>** или **M98 P <>**
M97 Q(R)<button> L FFFF M97 Q(R)<button> L	По кнопке запланировать предыдущий файл - подставить его значение в очередную команду **M47, M88 P <>, M89 P <>** или **M98 P <>**	По кнопке запланировать файл, предшествовавшему Основному - подставить его значение в очередную команду **M47, M88 P <>, M89 P <>** или **M98 P <>**
M97 Q(R)<button> J xxxx	По кнопке вызвать файл или запланировать его на конец быстрой анимации, если она проигрывается, чтение в **Основном режиме**, сначала	По кнопке вызвать файл или запланировать его на конец быстрой анимации, если она проигрывается, чтение в **режиме Подпрограммы**, сначала
M97 Q(R)<button> J FFFF M97 Q(R)<button> J	По кнопке вызвать предыдущий файл или запланировать его на конец быстрой анимации, если она проигрывается, чтение в **Основном режиме**, сначала	По кнопке вызвать файл, предшествовавшему Основному, или запланировать егона конец быстрой анимации, если она проигрывается, чтение в **режиме Подпрограммы**, сначала
M97 Q(R)<button> i xxxx	По кнопке вызвать файл, чтение в **режиме Подпрограммы**, сначала	По кнопке вызвать файл, чтение в **режиме Подпрограммы**, сначала
M97 Q(R)<button> i FFFF M97 Q(R)<button> i	По кнопке вызвать предыдущий файл, чтение в **режиме Подпрограммы**, сначала	По кнопке вызвать файл, предшествовавшему Основному, чтение в **режиме Подпрограммы**, сначала
M97 Q(R)<button> O	-	По кнопке вернуться в Основной файл в момент перехода в Подпрограмму

14. Реагирование на комбинацию кнопок

В версии 0.5+ возможно исполнение события по комбинации из нескольких одновременно нажатых кнопок вместо реакции на нажатие или отпускание отдельной кнопки.

Переключение в этот режим осуществляется по команде **M86 P1**.
Кроме того можно включить эмуляцию кнопок по байтам, приходящим по UART (команда **UxS 1**)

Когда этот режим включён, при назначении события командами **M96,** а также при эмулировании нажатия кнопок командой **G25** после операнда **Q<>** записываются те кнопки, которые в данной комбинации должны быть нажаты, а операнд **QR<>** не работает. Если вместо **Q<>** или **QR<>** написать **QH<>**, а затем число в формате HEX (от 0 до 0x3F, всего 64 варианта, из которых варианты с 0 по 0x1F соответствуют событиям по нажатию/отпусканию кнопок), то контроллер воспримет это число как значение байта в режиме реакции порта на отдельные байты. При этом есть прямое соответствие между событиями в режиме отдельного нажатия и в режиме комбинации:

Событие при реагировании на нажатие/отпускание отдельных кнопок	Комбинация кнопок в режиме Комбинации	Байт, в режиме реакции порта на отдельные байты
Нажатие логической кнопки 0	нет	00000000 или 0x00 или 0
Нажатие кнопки 1	1	00000001 или 0x01 или 1
Нажатие кнопки 2	2	00000010 или 0x02 или 2
Нажатие кнопки 3	1 + 2	00000011 или 0x03 или 3
Нажатие кнопки 4	3	00000100 или 0x04 или 4
Нажатие кнопки 5	1 + 3	00000101 или 0x05 или 5
Нажатие кнопки 6	2 + 3	00000110 или 0x06 или 6
Нажатие логической кнопки 7	1 + 2 + 3	00000111 или 0x07 или 7
Нажатие логической кнопки 8	4	00001000 или 0x08 или 8
Нажатие логической кнопки 9	1 + 4	00001001 или 0x09 или 9
Нажатие логической кнопки A (10)	2 + 4	0001010 или 0x0A или 10
Нажатие логической кнопки B (11)	1 + 2 + 4	00001011 или 0x0B или 11
Нажатие логической кнопки C (12)	3 + 4	00001100 или 0x0C или 12
Нажатие логической кнопки D (13)	1 + 3 + 4	00001101 или 0x0D или 13
Нажатие логической кнопки E (14)	2 + 3 + 4	00001110 или 0x0E или 14
Нажатие логической кнопки F (15)	1 + 2 + 3 + 4	00001111 или 0x0F или 15
Отпускание логической кнопки 0	5	00010000 или 0x10 или 16
Отпускание кнопки 1	1 + 5	00010001 или 0x11 или 17

Отпускание кнопки 2	2 + 5	00010010 или 0x12 или 18
Отпускание кнопки 3	1 + 2 + 5	00010011 или 0x13 или 19
Отпускание кнопки 4	3 + 5	00010100 или 0x14 или 20
Отпускание кнопки 5	1 + 3 + 5	00010101 или 0x15 или 21
Отпускание кнопки 6	2 + 3 + 5	00010110 или 0x16 или 22
Отпускание логической кнопки 7	1 + 2 + 3 + 5	00010111 или 0x17 или 23
Отпускание логической кнопки 8	4 + 5	00011000 или 0x18 или 24
Отпускание логической кнопки 9	1 + 4 + 5	00011001 или 0x19 или 25
Отпускание логической кнопки A (10)	2 + 4 + 5	00011010 или 0x1A или 26
Отпускание логической кнопки B (11)	1 + 2 + 4 + 5	00011011 или 0x1B или 27
Отпускание логической кнопки C (12)	3 + 4 + 5	00011100 или 0x1C или 28
Отпускание логической кнопки D (13)	1 + 3 + 4 + 5	00011101 или 0x1D или 29
Отпускание логической кнопки E (14)	2 + 3 + 4 + 5	00011110 или 0x1Eили 30
Отпускание логической кнопки F (15)	1 + 2 + 3 + 4 + 5	00011111 или 0x1F или 31
нет	6	00100000 или 0x20 или 32
нет	1 + 6	00100001 или 0x21 или 33
…	…	…
нет	1 + 2 + 3 + 4 + 5 + 6	00111111 или 0x3F или 63

К примеру в режиме реагирования на отдельные кнопки команда **M96 QR4 <>** и команда **M96 QH14 <>** будут управлять одним и тем же событием. А в режиме реагирования на комбинацию команды **G25 Q13**, **G25 QH 5** и байт **0x05** пришедший по UART в режиме реагирования на отдельные байты, вызовут одно и то же действие.

Поскольку в реальном мире невозможно идеально одновременно нажать на несколько кнопок, режим комбинации осуществляет ожидание окончания набора каждой комбинации. Длительность такого ожидания регулируется командой **G72 P<>** пределах от 20 миллисекунд до 10 секунд (по умолчанию это пол секунды). То есть команда не будет исполнена сразу в момент нажатия первой же кнопки либо в момент окончания набора комбинации. Кроме того, выбор команды осуществляется исходя из того, какие кнопки останутся нажаты к моменту окончания нажатия на комбинацию и не учитывает кнопки, которые были нажаты и затем отпущены до этого момента (то есть можно начать набирать одну комбинацию, затем отпустить часть или все кнопки и набрать другую и, если уложиться во время набора комбинации, контроллер этого не заметит). Это следует учитывать, если имеют значение тайминги переключения файлов, сдвигая паузы соответствующим образом.

Отсчёт времени ожидания осуществляется с момента изменения состояния кнопок вне периода ожидания. Таким образом, если установить период ожидания на минимальное значение, а события

назначить только на такие комбинации кнопок, что каждая отличается от любой другой состоянием хотя-бы одной кнопки, то время реакции на нажатие комбинации тоже может получиться минимальным. Также если предполагать, что кнопки будут всегда нажиматься в определённом порядке, то пропуск одной из кнопок при минимальном времени ожидания может дать свою комбинацию.

Примерами такого использования может быть использование нескольких групп гирконов, расположенных в разных частях костюма. При поднесении магнита этим группам каждая будет выдавать свою “комбинацию нажатых кнопок”, а то, что все комбинации будут отличаться друг от друга хотя-бы одной кнопкой даст дополнительную надёжность. Для пяти кнопок это 10 комбинаций, таких что ни одна из них не будет воспринята как другая в процессе набора:

123,124,125,134,135,145,234,235,245,345.

Если комбинация выполнена (по нажатию или по байту, отправленному по UART), то чтобы набрать новую комбинацию, сначала необходимо чтобы комбинация на кнопках сбросилась на нулевую. Это происходит когда все кнопки отпущены или когда истечёт период времени повторного нажатия (регулируется командой **G26**). В режиме реагирования на одиночные нажатия/отпускания кнопок выполнять это требование не нужно.

15. Условные события

В некоторых сценариях, при создании интерактивных декораций или реквизита может потребоваться включение определенной анимации исходя из состояния анимации или картинки, полученной через внешний порт (USART).

Например, это может быть декорация, в которой есть некий пульт. При нажатии кнопок на этом пульте создаётся та или иная картинка на пиксельном экране, и, в зависимости от этой картинки, запустится или не запустится какой-либо процесс.
Либо нам нужно, чтобы при нажатии на разные кнопки по декорации перемещались бегущие огни, которые, достигнув определённых позиций, “включали” лампы или “изменяли” положение каких-то висящих на серводвигателях элементов. Причём размер и количество элементов в декорации может требовать использования нескольких контроллеров, связанных по USART.

Для возможности реализации таких сценариев предусмотрены следующие элементы:
Макрос **\c<><>** для команды **UxT**, позволяет реализовывать большие бегущие панно, бегущие огни разной сложности и длительности, передавая цвета из массива одного контроллера в массив другого:

```
U1T P**** S02 P \c0000 \c0001 GG \c2004 \c2005 LL 00FF11
; отправить через первый порт сообщение всем подсоединённым контроллерам (**** ):
; на выводах 0 и 2 (S02 P) первый и второй пикселы сделать такими же цветами,
;что и у данного контроллера на 0-м выводе у первого и второго пикселов соответственно
;( \c0000 \c0001 ), затем пропустить 2 светодиода (GG),
;затем 5 и 6й пикселы сделать того же цвета что 5 и 6 у данного на втором выводе
;( \c2004 \c2005), затем 2 пиксела выключить (LL)
;и ещё один сделать зелёным с лёгким оттенком бирюзы (00FF11)
```

Макрос **?<color>** для команды **S<>P<>**, позволяет определить насколько состояние светодиодов отличается от того, которое нам нужно. Это позволит, к примеру, заметить, когда бегущий огонёк достиг своей цели, или когда набрана нужная комбинация на “пульте”:

```
M45 ;сбросить количество несовпадений
S1 P ?L ?L ?FF0000 > > ggg ?H
;проверить, отличается ли цвет пикселей в позиции 0 (первый пиксель)
;на 1 выводе от чёрного (?L),
;проверить, отличается ли цвет пикселей в позиции 1 на 1 выводе от чёрного (?L),
;проверить, отличается ли цвет пикселей в позиции 2 на 1 выводе от ярко красного (?FF0000),
;проверить, отличается ли цвет пикселей в позиции 3 на 1 выводе
;от указанного для предыдущего (>), то есть FF0000
;то же для следующего за ним
;затем пропустить пикселы в следующих трёх позициях ( ggg), нам всё равно что там,
; проверить, отличается ли цвет пикселей в позиции 8 на 1 выводе от белого (?H)
```

Тут важно отметить, что если сравниваются светодиоды сразу на нескольких выводах, то есть команда **S<>** имеет вид **“S015 P…”** и работает сразу с несколькими выводами, то макрос сравнивает записанный цвет с цветами пикселов на каждом из выводов, и считает, что если часть из них не совпадает, то это одно общее несовпадение для пикселов в данной позиции.

То есть если, прежположим, первый пиксель на 0 выводе имел цвет 110000, на 1 выводе 001100 а на 2 выводе - 000011, то команда **S012 P ?110000** выдаст одно несовпадение. Так же как и в случае, если и на 0 и на 1 выводах будет 110000 и только на 2 цвет будет другой.
Количество несовпадений сбрасывается отдельной командой **M45**, поскольку для принятия решения может потребоваться изучить, насколько совпадают разные части картинок на разных выводах с требуемым для исполнения соответствующих команд..

Команда **G25 Q(R) <> P<><>** позволяет выполнить событие (“нажать” или “отпустить” кнопку или “нажать” комбинацию кнопок) только если количество несовпадений больше (>), меньше (<), равно (=) или отличается (!) от указанного.

```
M45
S….. ; Проверяем цвета интересующих нас пикселов
…..
G25 Q 2 P>5
; если количество несовпадений больше 5, то “нажимаем” кнопку 2
G25 Q 1 P<1
; если количество несовпадений меньше 1 (то есть полное совпадение), то “нажимаем” кнопку 1
;продолжаем сценарий
```

Следует учитывать, что при присвоении цвета очередному пикселю учитывается параметр яркости. Также он учитывается при проверке. Однако если цвета были получены через USART, то контроллер не знает, была ли там какая-либо яркость, отличная от максимальной. Кроме того, если цвет не был назначен, а был перемещён из другого пикселя (макросы **“S…”, “P…”** и **“T…**“ “**M…**” в команде **S<>P<>**), то их контроль яркости игнорирует. Это значит, что в проектах, в которых предполагаются сценарии с проверками картинок на выводах следует:

либо отказаться от контроля яркости или сильно ограничить его использование,

либо осуществлять его только среди тех контроллеров, которые не пересылают информацию о своих пикселях другим контроллерам,

либо контролировать только включение/выключение пикселов (несовпадение с чёрным цветом)

DKled o4i3 вид и соединения

o4i3 имеет:

- 4 разъёма для светодиодов типа WS 2812 LED (до 512 на вывод)
- 3 разъёма для кнопок
- отдельный разъём для кнопки перезагрузки
- слот для Micro SD
- Mini USB для питания 5V

Версия прошивки 0.1

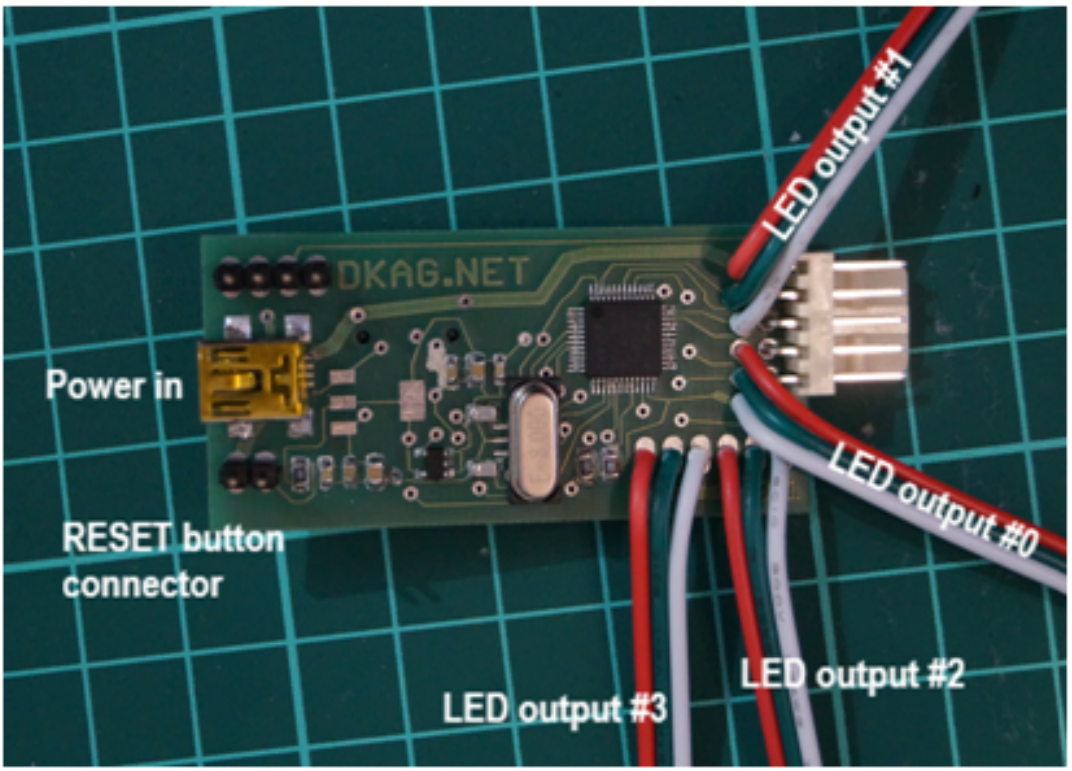


Fig. 1 Вид спереди

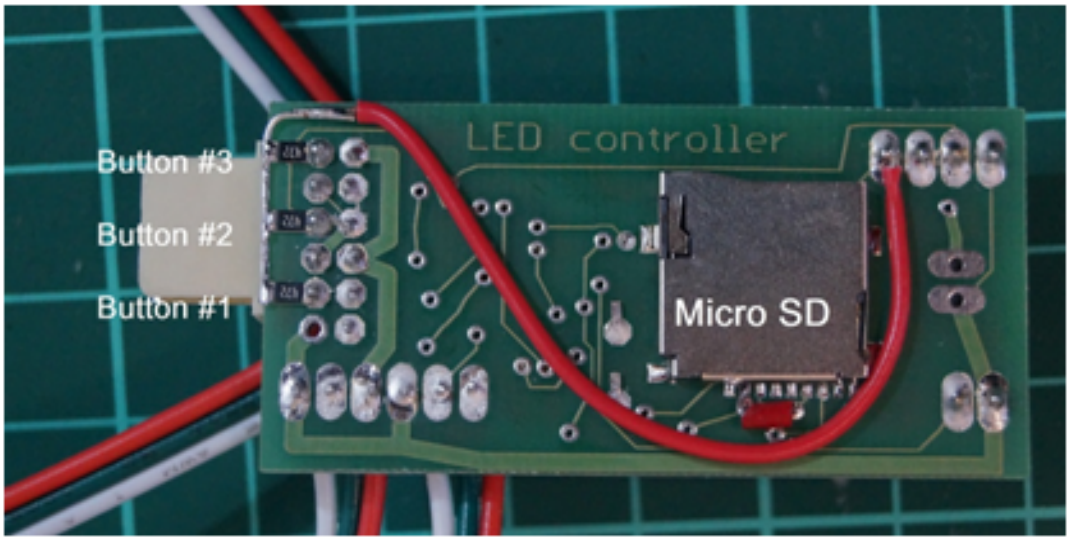


Fig. 2 Вид сзади

DKled I4O6U2 beta вид и соединения

i4o6u2 имеет:

- 6 выводов данных для светодиодов типа WS 2812 LED (до 512 на вывод) или серводвигателей (1 на вывод)
- отдельно выводы питания 0(GND) и +5вольт
- разъём для 4 кнопок с общим 0
- 2 вывода USART для связи с другими устройствами*
- место под выключатель
- слот для Micro SD
- USB для питания 5V и площадки для подпаивания питающего провода (3,7 - 5 вольт)

Версия прошивки 0.3
Штекер для USB может быть удалён при необходимости
Размеры:62 x 18 мм (50 x 18 мм без штекера)

* При изготовлении бета-версии оказалось, что контакты RX и TX на одном из USART перепутаны. В дальнейших версиях этот недостаток исправлен, однако в версии beta последовательность контактов для USART1 такова: RXD, TXD, +5V, GND, а для USART2 это TXD, RXD, +5V, GND. Будьте внимательны.
На финальной версии эта ошибка исправлена и i4o6u2 все выводы подписаны

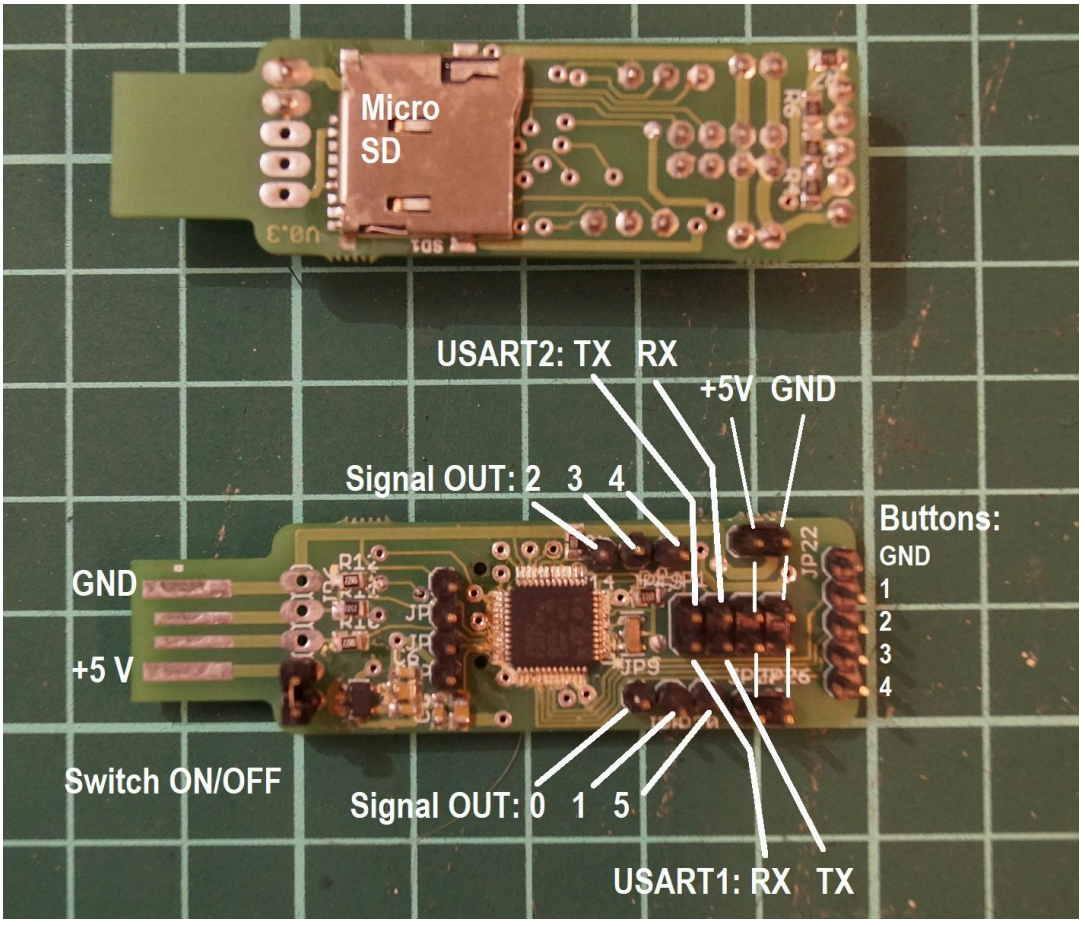


Fig. 3 Внешний Вид

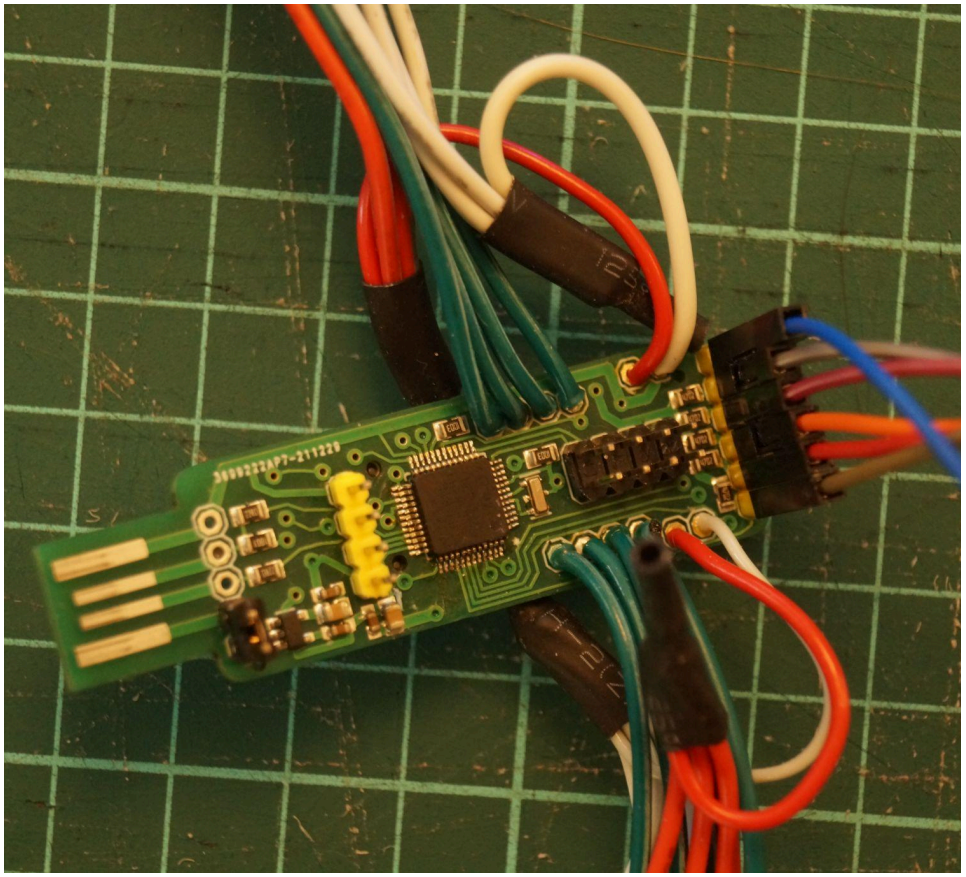
DKled I5O8U2 вид и соединения

i5o8u2 имеет:

- 8 выводов данных для светодиодов типа WS 2812 LED (до 512 на вывод) или серводвигателей (1 на вывод)
- отдельно выводы питания 0(GND) и +5вольт
- разъём для 5 кнопок с общим 0(GND)
- 2 вывода USART для связи с другими устройствами
- место под выключатель
- слот для Micro SD*
- USB для питания 5V и площадки для подпаивания питающего провода (3,7 - 5 вольт)

Версия прошивки 0.5
Штекер для USB может быть отпилен при необходимости
При втыкании контроллера в компьютер, тот определяет его как неопознанное устройство
Размеры: 62 x 18 мм (50 x 18 мм без штекера)

*поскольку для чтения/записи на SD карту используется более старый протокол(SPI), карты большого размера и карты некоторых производителей могут не работать. Наши тесты прошли карты Mirex, Transcend и SanDisk все 2, 4 и 8 ГБ. Карты SmartBuy не прошли проверку. Форматирование должно быть FAT32. Также не имеет смысла использовать карты размером больше 16Гб потому что FAT32 на больших картах работает медленнее а значит контроллер будет дольше запускаться и дольше переключаться между файлами.



Инструкция для присоединения проводов:

- Голубой - 6 точек,которые служат для присоединения 5 кнопок и общей “земли” для них. Кнопка воспринимается нажатой когда она замкнута на землю
- Белый и красный - земля и питание для светодиодов соответственно. Контроллер также может питаться от этих проводов. Напряжение - в диапазоне 3-5 вольт. Если нужно управлять элементами с большим напряжением питания, следует подсоединить только общую землю, либо вообще изолировать управляемые элементы через реле или транзисторы
- Зелёные (по с каждой стороны) - сигнальные провода. Напряжение на них не превышает 3 вольт. Их можно подсоединять к сигнальным проводам управляемых светодиодов, к низкоточным элементам, к проводам управления реле, транзисторов и серводвигателей с небольшим напряжением питания.
- Красный и белый места - питание и земля соответственно, также соответствуют выводам на контактах USB штекера, встроенного в плату с этой стороны. Питают контроллер и могут питать небольшое (до 150 штук на полной мощности) управляемых светодиодов. Предназначены для подсоединения провода USB. Также питание (+5 вольт) может быть прервано, если вместо джампера (слева) поставить выключатель
- Розовый - USART для связи с другими устройствами. Также может использоваться для питания контроллера и светодиодов, потребляющих до 5 вольт