Руководство по языку **MMBasic**
версия 4.4B

Geoff Graham

Для получения новых версий руководства и подробностей по MMBasic,
перейдите по ссылке http://mmbasic.com

или http://geoffg.net/maximite.html

Перевел на русский язык Скоморохов Алексей. Россия, г. Ставрополь.

MMBasic – это совместимая с Microsoft BASIC реализация языка, которая способна работать с числами с плавающей точкой, строковыми переменными, длинными именами переменных, массивами чисел с плавающей точкой, массивами строк с различными размерами, так же есть возможность мощной обработки строк.

MMBasic была изначально написана для небольшого компьютера Maximite на основе микроконтроллеров PIC32 от Microchip. В настоящее время, MMBasic работает на различных аппаратных платформах, включая DOS

Это руководство описывает язык MMBasic. Для получения дополнительной информации по запуску MMBasic на определенных платформах, пожалуйста, обратитесь к следующей документации или веб-сайтов:

Maximite, mini-Maximite: Maximite Hardware Manual по адресу: http://geoffg.net/maximite.html

UBW32 experimenter board: Colour Maximite on the UBW32 по адресу: http://geoffg.net/ubw32.html

DOS: DOS MMBasic ReadMe по адресу: http://mmbasic.com/downloads.html

DuinoMite series: DuinoMite MMBasic ReadMe включено в обновление DuinoMite.

CGMMSTICK1 и CGCOLORMAX2 от CircuitGizmos по адресу http://www.circuitgizmos.com

Модуль DTX2-4105C от Dimitech по адресу http://dimitech.com

Maximite от M & M Automation (Турция) по адресу http://www.maximite-tr.com

В данном руководстве «Maximite» или «MM» относится к оригинальному семейству монохромных Maximite (Maximite, mini Maximite, CGMMSTICK1, DuinoMite и DTX2-4105C). «Color Maximite» или «CMM» относится к цветной версии Maximite которая также включает UBW32 и CGCOLORMAX2. «DOS» относится к версии, которая работает в среде DOS box под Windows,

Содержание

Основные функции

Ввод команд и программ

В строке приглашения (сразу после символа «>») можно ввести команду, после чего нажать клавишу ввода, что вызовет выполнение команды. Это полезно для тестирования команд и наблюдения результатов их работы.
Чтобы изменить программу, вы можете воспользоваться командой EDIT, которая вызывает полноэкранный редактор, встроенный в MMBasic. Номера строк не являются обязательными, но если вы используете их, вы можете вводить программу, указывая номер перед каждой строкой.
При вводе строки в командной строке, её можно редактировать с помощью курсорных клавиш (влево/вправо – для перемещения вдоль строки, вверх/вниз – для перемещения по списку ранее введенных команд), кнопки Delete (для удаления символа) и кнопки Insert (для переключения между вставкой и перезаписью символов).
Программы сохраняются в памяти и могут быть просмотрены командой LIST, запущены с помощью команды RUN или очищены командой NEW. Вы можете прервать MMBasic в любое время, набрав CTRL-C и на экране появится строка приглашения.

Клавиатура/Дисплей

Ввод можно выполнять с помощью любой PS/2 клавиатуры или персонального компьютера, с помощью терминала, через USB или UART. Клавиатура и интерфейс USB могут работать одновременно их можно отключать/подключать в любое время без ущерба для выполняемой программы.
Вывод информации работает одновременно через интерфейс USB и стандартный монитор (VGA или композитный). Они так же могут быть подключены или отключены в любое время.

Номера строк, структура программы и редактирование

Структура программной строки:

```
[Номер строки]  [метка:]  аргументы команды [: аргументы команды] …
```

Метка или номер строки могут быть использованы для обозначения строки кода. К метке предъявляются те же требования (длина, набор символов, и т.д.) что и к имени переменной, она так же не может совпадать с именами команд. При обозначении строки меткой, метка должна появиться в начале строки, но после номера строки (если используется), и оканчиваться двоеточием (:). Такие команды, как GOTO могут использовать метки или номера строк для определения строки, на которую нужно выполнить переход (при указании метки в команде перехода, метка не должна оканчиваться двоеточием). Например:

```
GOTO xxxx
    - - -
    xxxx:  PRINT "We have jumped to here"
```

Несколько команд, разделенных двоеточием могут быть введены в одну строку (INPUT A : PRINT B...).

Длинные программы (с или без номеров строк) могут быть отправлены через USB с помощью команд XMODEM (только для Maximite) или AUTO.

Хранение программ и данных

В DOS версии MMBasic поддерживаются зарегистрированные Windows буквы дисков.

В Maximite и его цветной версии доступно два «диска» для сохранения и загрузки программ и данных:

Диск «A:» – виртуальный, использует внутреннюю флеш память микроконтроллера PIC32 и имеет объем около 180 КБ в монохромном Maximite (и немного меньше в цветной или CAN версиях).

Диск «B:» – SD карта (если подключена). Поддерживаются карты MMC, SD или SDHC форматированные в FAT16 или FAT32 с ёмкостью до 32 ГБ.

Имена файлов должны быть в формате 8.3 (8 символов латинского алфавита – имя и 3 – расширение) с добавлением префикса «A:» или «B:» (так же как в DOS или Windows). Длинные имена файлов или папок не поддерживаются. Диск по умолчанию – B:, его можно изменить командой DRIVE.

При запуске Maximite, MMBasic начнет поиск файла с именем «AUTORUN.BAS» в корневом каталоге внутренней флеш памяти (диск A:), затем на карте SD (диск B:). Если файл будет обнаружен, он будет автоматически загружен и запущен, иначе MMBasic выведет строку приглашения («>») и будет ждать ввода команд. Если Maximite был перезагружен по причине срабатывания сторожевого таймера (в результате зависания системы), будет найден и запущен файл с именем «RESTART.BAS» (см. описание команды WATCHDOG).
Примечание: изображение будет пропадать на короткое время во время записи данных на диск A:. Это нормально, и связано с необходимостью отключения видеовыхода для перепрограммирования внутренней памяти.

Будте осторожны при использовании диска A:, чтобы избежать преждевременного износа внутренней флеш-памяти. Если диск A: пуст, Вы можете записывать и удалять файлы каждый день (раз в день) в течение 175 лет до нарушения его работоспособности, однако, если запись будет выполняться раз в минуту, флеш-память будет изношена в течение 6 недель.

Команды и функции хранения

Программа может быть сохранена на любом диске с помощью команды SAVE, загружена с помощью команды LOAD или добавлена к текущей программе с помощью команды MERGE. Сохраненная программа может быть загружена и запущена с помощью команды RUN. Команда RUN также может быть добавлена в запущенную программу, что позволит этой программе загрузить другую и передать ей управление. Команда CHAIN позволяет программе загружать и запускать другую программу, сохраняя текущее состояние программы (то есть, значение переменных, открытых файлов, загруженных шрифтов, открытых портов COM и т.д.). Так как программа может быть разбита на модули, команда CHAIN позволяет писать программы почти неограниченного размера при ограниченном объеме оперативной памяти.
Команда LIBRARY загружает файл, содержащий пользовательские команды и функции, которые затем могут быть вызваны запущенной программы. Это обеспечивает простой способ расширения языка, путем создания специализированных библиотек математических функций, драйверов оборудования и т.д. Команда LIBRARY также может быть использована для экономии памяти загрузкой и выгрузкой отдельных фрагментов кода в процессе выполнения программы. Файлы с данными могут быть открыты с помощью команды OPEN и прочитаны с использованием команд INPUT, LINE INPUT, или INPUT $ () или записаны с помощью PRINT или WRITE. На SD-карте данные и программы хранятся с использованием стандартного текста и могут быть прочитаны и отредактированы в Windows, Apple Mac, Linux, и т.д. До 10 файлов одновременно могут быть открыты из SD-карты, в то же время из внутреннего флэш-накопителя возможно открытие максимум одиного файла одновременно.
Вы можете получить список программ, хранящихся на диске с помощью команды FILES, удалять их с помощью KILL и переименовать их, используя NAME. Текущий рабочий каталог на SD-карте можно изменить с помощью CHDIR. Новый каталог может быть создан командой MKDIR или удален командой RMDIR.
Указание имени файла может быть задано строковой константой (т.е. заключено в двойные кавычки) или строковой переменной. Это означает, что вы должны использовать кавычки, если вы прямо указываете имя файла (например, KILL “TEST.BAS”). Тем не менее, кавычки необязательны, если команда используется в командной строке. Обратите внимание, что символы (такие как +, -, *) и ключевые слова (команды MMBasic) в имени файла, написанном без кавычек, приведет к ошибке. Кавычки всегда требуется, если команда используется в программе.

Время

Вы можете получить текущую дату и время, используя функции DATE $ и TIME $, Вы можете установить их, назначив новую дату и время. Цветная версия Maximite имеет часы реального времени с резервным питанием, на других Maximite календарь начнется с полуночи 1 января 2000 при включении питания. На версии DOS будет использоваться системное время.
Вы можете задержать выполнение программы на заданное количество миллисекунд, используя команду PAUSE. MMBasic также поддерживает внутреннюю функцию секундомера (таймера), который отсчитывает время в миллисекундах. Вы можете сбросить таймер в ноль или загрузить любое другое число путем присвоения таймеру этого значения. Используя команду SETTICK в Maximite, Вы можете настроить до четырех "тиков", которые будет генерировать регулярные прерывания с периодом от одной миллисекунды до более месяца. См. о прерываниях ниже.

Выражения

В большинстве случаев, где требуется номер или строка, Вы можете также использовать выражение. Например:

```
FNAME$ = “TEST”: RUN FNAME$ + “.BAS”  'будет запущено “TEST.BAS”
```

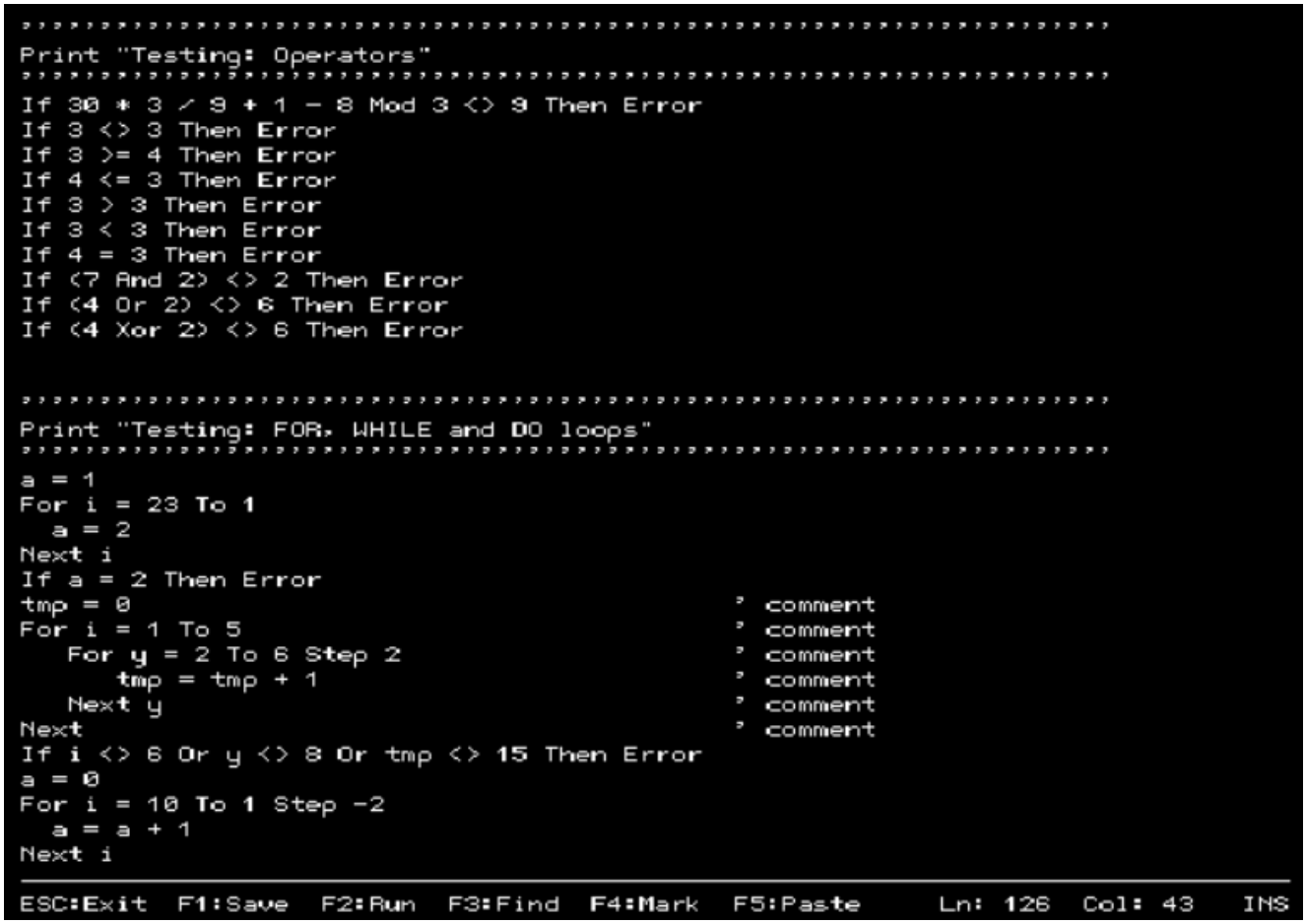
Структурированные операторы

MMBasic поддерживает несколько структурированных операторов.
Команды DO WHILE … LOOP и их вариации позволяют легко строить циклы без использования оператора GOTO.
Определенные подпрограммы и функции позволяют легко добавить свои собственные команды в MMBasic.
Команды IF… THEN при необходимости могут охватывать много строк совместно с операторами ELSEIF … THEN, ELSE и ENDIF. Например:

```
IF <условие> THEN          ' Начало структуры выбора IF
  <операторы>
ELSEIF <условие> THEN      ' дополнительные условие (опционально)
  <операторы>
ELSE                       ' если ни одно условие не выполнено (опционально)
  <операторы>
ENDIF                      ' окончание структуры IF
```

Полноэкранный редактор

Важной функцией MMBasic является полноэкранный редактор (не доступен в версии DOS). Он будет работать с помощью подключенного дисплея (VGA или композитного) и через USB с VT100 совместимым эмулятором терминала (рекомендуется Tera Term).



Полноэкранный редактор вызывается с помощью команды EDIT. Если просто ввести EDIT, редактор автоматически начнет редактирование того, что находится в оперативной памяти. Если память пуста Вам будет представлен пустой экран.

Курсор будет автоматически установлен на последнем месте, где Вы выполняли редактирование и, если ваша программа только что была остановлена ошибкой, курсор будет указывать на линию, которая вызвала ошибку. Вы также можете запустить редактор с именем файла (например, EDIT "file.ext") и редактор будет редактировать этот файл, оставляя память программ нетронутыми. Это удобно для изучения или изменения файлов на диске, без нарушения Вашей программы.

Если Вы привыкли к редакторам, таким как Блокнот, Вы увидите, что работа в полноэкранном редакторе Вам знакома. Курсорные клавиши будут двигаться курсор в тексте, home и end установят курсор в начало или конец строки. Page up и Page down будут перемещать на страницу вверх или вниз. Кнопка delete будет удалить символ в позиции курсора, а backspace будет удалить символ перед курсором. Клавиша insert будет переключать режимы между вставкой и заменой.

Необычным сочетанием клавиш является то, что двойное нажатие home переместит курсор в начало программы, а двойное нажатие end – в конец.

В нижней части экрана в строке состояния перечислены различные функциональные клавиши, используемые в редакторе и их действия. Более подробно о них:

ESC Приведет к выходу из редактора с отменой всех изменений и возврату к командной строке. Если Вы изменяли текст, потребуется подтверждение перед закрытием редактора.

F1: SAVE	Сохранит программу в памяти программ, затем произойдет возвращение к командной строке. При редактировании файла на диске, сохранение будет выполняться на диск.
F2: RUN	Сохранит программу в памяти программ и незамедлительно запустит её.
F3: FIND	Запросит текст, который Вы хотите найти. При нажатии кнопки enter, курсор будет перемещен к началу первой найденной записи.
SHIFT-F3	После того, как Вы использовали функцию поиска вы можете повторить поиск того же текста, нажатием сочетания SHIFT-F3.
F4: MARK	Функция описана ниже.
F5: PASTE	Вставит текст (в текущей позиции курсора) который был вырезан или скопирован ранее.
CTRL-F	Вставит (в текущей позиции курсора) файл находящийся на диске. Обратите внимание, что в то время как это сочетание не указано в строке состояния, оно доступно всегда.

Вы также можете использовать клавиши управления вместо функциональных клавиши, перечисленных выше:

LEFT	Ctrl-S	RIGHT	Ctrl-D	UP	Ctrl-E	DOWN	Ctrl-X
HOME	Ctrl-U	END	Ctrl-K	PageUp	Ctrl-P	PageDn	Ctrl-L
DEL	Ctrl-]	INSERT	Ctrl-N	F1	Ctrl-Q	F2	Ctrl-W
F3	Ctrl-R	Shift-F3	Ctrl-G	F4	Ctrl-T	F5	Ctrl-Y

Если Вы нажали клавишу F4, редактор перейдет в режим выделения. В этом режиме можно использовать клавиши со стрелками, чтобы отметить часть текста, который будет выделен. После этого можно удалить, вырезать или скопировать выделенный текст. В этом режиме строка состояния изменится, чтобы показать функции функциональных клавиш в режиме выделения:

ESC	Выход из режима выделения.
F4: CUT	Вырезать выделенный текст в буфер обмена.
F5: COPY	Копировать . выделенный текст в буфер обмена.
DELETE	Удалить выделенный текст, оставив буфер обмена без изменений.

Лучший способ познакомиться с полноэкранным редактором – это просто запустить его и экспериментировать.

Редактор – очень продуктивный метод написания программ. Используя команду OPTION Fnn, Вы можете запрограммировать функциональную клавишу для генерации команды «EDIT». Так, всего одним нажатием, Вы можете перейти в редактор, где можно внести изменения. Затем, нажав клавишу F2 можно сохранить и запустить программу. Если Ваша программа останавливается с ошибкой, Вы можете нажать функциональную клавишу редактирования и вернуться в редактор с курсором, установленным на линии, которая вызвала ошибку. Этот цикл: редактирование/запуск/редактирование выполняется очень быстро.

Если Вы использует полноэкранный редактор через USB с эмулятором терминала Terra Term, измените размер окна Terra Term до 80 символов в 36 строках. Для получения подробной информации см. приложение G.

Обратите внимание, что эмулятор терминала может потерять свою позицию в тексте после нескольких быстрых нажатий клавиш (таких как стрелка вверх или вниз). Если это произойдет, Вы можете нажать клавишу HOME дважды, она заставит редактор перейти к началу программы и перерисовать экран.

Ввод/вывод

Следующие функции поддерживаются только в Maximite (не для DOS версии).

Внешний ввод/вывод (I/O)

Вы можете настраивать внешние выводы I/O командой SETPIN, устанавливать их выходное значение командой PIN()=, и читать их текущее состояние функцией PIN(). Цифровой ввод/вывод использует ноль для выдачи низкого и единицу для высокого уровня напряжения. Аналоговый ввод возвращает измеренное напряжение в формате с плавающей точкой.

Оригинальный Maximite имеет 20 выводов I/O, пронумерованных от 1 до 20. Выводы от 1 до 10 могут быть использованы для аналогового ввода и цифрового ввода/вывода с максимальным входным напряжением 3.3 В. Выводы от 11 до 20 только цифровые и поддерживают напряжения до 5 В и могут быть переключены в режим с открытым коллектором. Версия DuinoMite имеет полностью отличающиеся и запутанные назначения выводов см. описание в файле «DuinoMite MMBasic ReadMe.pdf»

Нормальный цифровой выход имеет выходные напряжения 0 В (низкий уровень) и 3.3 В (высокий уровень), но Вы можете использовать режим с открытым коллектором, чтобы управлять цепями с питанием 5 В. Это означает, что при низком выходном уровне, вывод будет заземлен, но при высоком выходном уровне выходное сопротивление будет велико. Таким образом, если вывод настроен на работу с открытым коллектором, потребуется «подтягивающий» резистор между выходом и источником питания 5 В. Типовое сопротивление «подтягивающего» резистора находится в пределах от 1 кОм до 4.7 кОм.

Разъем Arduino

В дополнение к описанным выше 20 выводам I/O, цветная версия Maximite имеет 20 выводов на внутреннем разъеме, совместимом с Arduino (40 всего выводов I/O суммарно). Они обозначаются от D0 до D13 для цифровых и от A0 до A5 для аналоговых выводов соответственно.

В выражениях SETPIN и PIN Вы можете использовать имена D0, D1, и т.д. или соответствующие им номера (D0 = 21, D1 = 22, и т.д. и A0 = 35, A1 = 36, и т.д.). Цифровые выводы (D0 – D13) имеют те же характеристики (5 В, открытый коллектор, и т.д.) что и цифровые выводы с 11 по 20. Аналоговые входы (A0 – A5) имеют возможности аналогичные внешним выводам от 1 до 10.

Коммуникации

Поддерживается два последовательных порта со скоростью передачи до 19200 бод с настраиваемыми объемами буферов и опциональным аппаратным контролем потока данных. Последовательные порты открываются командой OPEN или любой командой либо функцией, использующей имя файла, которое может быть назначено порту для передачи или приема информации. См. Приложение A для получения подробной информации.

Связь по шине I^2C в режиме мастера или подчиненного осуществляется с помощью восьми команд (см. Приложение B, для получения подробной информации). MMBasic полностью поддерживает режимы мастера и подчиненного, 10-ти битную адресацию, маскирование адреса, функцию общего вызова, а также арбитраж шины (для предотвращения конфликтов на шине в режиме мульти-мастера).

Последовательный интерфейс SPI поддерживается командой «SPI». См. приложение D. Однопроводной интерфейс «1-Wire» так же поддерживается. См. Приложение C.

Прерывания

Большинство внешних I/O выводов могут быть настроены на вызов прерывания путем использования команды SETPIN совместно с многими другими активными прерываниями (включая прерывания по таймеру). Прерывания могут быть настроены на срабатывание по нарастанию и по спаду цифрового сигнала и вызывают незамедлительный переход на определенную строку, метку или определяемую пользователем подпрограмму. Адрес перехода может быть различным для каждого из прерываний. Возврат из прерывания осуществляется выражением IRETURN за исключением случая использования пользовательских подпрограмм (в этом случае используются END SUB или EXIT SUB). В программе обработки прерываний могут использоваться команды GOTO и GOSUB для вызова других подпрограмм.

Если произошло два и более прерываний, их приоритет будет соответствовать номеру вывода (т.е. прерывание по входу 1 будет иметь высший приоритет). Во время обработки прерывания, остальные прерывания блокируются до выхода из прерывания командой IRETURN. Во время обработки прерывания (как и в остальное время) значение на входе, используемом для прерывания, доступно для чтения при использовании функции PIN().

До четырех прерываний с различными временными периодами (в миллисекундах) могут быть настроены выражением SETTICK. Эти прерывания имеют наименьший приоритет. Использование команды ON KEY позволяет генерировать прерывание при нажатие любой клавиши.

Прерывания могут происходить в любое время, но они блокируются при выполнении команды INPUT. Если Вам нужен ввод с клавиатуры и прерывания, используйте функцию INKEY$ или команду ON KEY. Прерывание основной программы не повлияет на её работу, если переменные, используемые основной программы, не изменятся во время прерывания

Для большинства программ MMBasic будет реагировать на прерывания с задержкой до 100 мкс. Чтобы предотвратить замедление основной программы, прерывание должно быть коротким и заканчиваться быстро, насколько это возможно. Также не забудьте отключить прерывания, когда они не требуются – фоновые прерывания могут вызвать непредсказуемые ошибки.

Аудио и ШИМ выход

Существует несколько способов использования звукового выхода. Вы можете воспроизводить синтезированную музыку, тональные сигналы или программно задавать напряжение (ШИМ).

Воспроизведение MOD файлов

Команда PLAYMOD запускает воспроизведение синтезированной музыки в фоновом режиме, во время работы программ. Аудиофайл должен быть в формате MOD и файл, содержащий музыку, должен быть расположен на внутреннем накопителе (диск A:). Модель Colour Maximite воспроизводит высококачественный стереозвук.

Музыкальный формат файлов MOD берет начало от MOD файлов использовавшихся в системе Amiga, которая была разработана в конце 1980-х. Это не музыкальная запись (как MP3 файл), а набор инструкций для синтеза музыки. На оригинальном Amiga эта задача была выполнена аппаратными средствами. MMBasic будет считывать эти файлы и непрерывно воспроизводить в фоновом режиме, когда будет выполняться программа. Учтите, что синтез звука требователен к ресурсам ЦПУ и использует оперативную память и это негативно повлияет на время выполнения команд. Описание формата MOD можно найти по адресу: http://en.wikipedia.org/wiki/MOD_(file_format) Большое количество файлов, которые можно воспроизводить на Maximite можно найти на сайте: http://modarchive.org (ищите файлы с разрешением .MOD). Поскольку файл должен находиться на диске A:, целесообразно использовать небольшие файлы для воспроизведения. Вы так же можете создавать свои собственные файлы, используйте трекер, например: http://www.modplug.com

TONE

Эта команда создает два тональных сигнала (в Colour Maximite), которые будут выводиться раздельно на левый и правый канал звука. В монохромном Maximite генерируется только один тон. Тон синтезируется в виде синусоиды и может задаваться в диапазоне от 1 Гц до 20 кГц с разрешением 1 Гц и высокой точностью, так как он синхронизирован с кварцевым генератором PIC32. Изменение частоты происходит без прерывания вывода, таким образом, возможна плавная перестройка по диапазону. Время воспроизведения может быть указано в миллисекундах, и тон будет воспроизводиться в фоновом режиме (то есть, программа продолжит работать).

SOUND

Эта команда добавлена для обеспечения совместимости со старыми программами. Она генерирует одночастотные прямоугольные импульсы. В новых программах рекомендуется использование команд TONE или PWM.

ШИМ

Команда PWM (широтно-импульсная модуляция) позволяет Maximite генерировать прямоугольные импульсы с программно управляемым коэффициентом заполнения. Изменением коэффициента заполнения Вы можете генерировать программно управляемое выходное напряжение, для управления внешними аналоговыми устройствами (источники питания, контроллеры двигателей, и т.д.). Colour Maximite имеет два канала ШИМ, тогда как у монохромного Maximite он один.

Частота обоих выходов ШИМ одинакова и может задаваться в пределах от 20 Гц до 1МГц. Коэффициент заполнения может задаваться в пределах от 0% до 100% с разрешением 0.1% когда частота ниже 50 кГц (при частоте выше 50 кГц разрешение в 1% доступно до частоты 500 кГц).

При включении Maximite или когда используется команда PWM OFF, ШИМ выходы переводятся в высокоимпендансное состояние (Z-состояние). Итак, если Вы хотите, чтобы вывод ШИМ имел низкий логический уровень по умолчанию (нулевая мощность в большинстве случаев), вы должны использовать «подтягивающий» резистор на землю. Аналогично, если Вы хотите, чтобы по умолчанию, был высокой уровень (полная мощность) необходимо подключить «подтягивающий» резистор к шине питания 3,3 В.

Эта команда использует звуковой выход для формирования сигнала ШИМ, следовательно компоненты подключенные к этому выходу, возможно, потребуется изменить или удалить.

Графика и работа с цветом

Графика

Графические команды работают только с видеовыходом (не через USB). Координаты измеряются в пикселях с координатой x по горизонтали и y по вертикали. Верхняя левая часть экрана соответствует x = 0 и y = 0, увеличение чисел соответствует движению вниз и вправо. Количество пикселей на экране определяется доступными только для чтения переменными MM.HRES и MM.VRES которые изменяются в зависимости от выбранного видео режима (VGA или композитный PAL / NTSC)

Вы можете очистить экран командой CLS. Индивидуальный пиксель может быть включен или выключен а его цвет можно установить командой PIXEL (x, y) =. Вы можете рисовать линии и прямоугольники используя команду LINE, и круги, используя CIRCLE. Вы также можете установить положение экрана (в пикселях) для вывода (команда PRINT) с помощью @ (x, y). Команда SAVEBMP сохранит изображение на текущем экране в виде файла BMP. Команда LOADBMP загружает и отображает растровое изображение, которое хранится на SD-карте.

Работа с цветом

Цветная версия Maximite поддерживает восемь цветов (черный, синий, зеленый, голубой, красный, фиолетовый, желтый and белый). Монохромный Maximite и DuinoMite поддерживают только два – черный и белый. В большинстве случаев, Вы можете задать цвет числом –1 для инвертирования цвета пикселя (это полезно для анимации).

В MMBasic Вы можете указывать цвет по имени или по соответствующему номеру, где:

Черный: black = 0; синий: blue = 1; зеленый: green = 2, и т.д. до белого: white = 7. Такие команды как LINE и CIRCLE используют эти имена или их номера для определения цвета объекта, например:

```
CIRCLE (100, 100), 50, CYAN      ‘нарисует круг голубого цвета.
CIRCLE (100, 100), 50, 3         ‘так же нарисует голубой круг (cyan = 3).
```

Вы также можете задать цвет, который будет использоваться по умолчанию для всего экрана, командой COLOUR. Например: `COLOUR PURPLE` переключит вывод текста на фиолетовый (везде, где не указан другой цвет). Команда COLOUR кроме того принимает второй параметр, для цвета фона. Таким образом:

`COLOUR YELLOW,BLUE` – будет отображать желтый текст на синем фоне.

В дополнение к команде COLOUR, Вы можете изменить цвет текста путем встраивания цветовых кодов в строки с помощью функции CLR $ (). В следующем примере каждое слово будет отображаться разными цветами:

```
Txt$ = "This is " + CLR$(RED) "red " + CLR$(YELLOW) + "yellow"
     PRINT Txt$
```

Вы также можете использовать эту функцию, чтобы установить цвет фона, добавляя второй параметр. например:

```
     PRINT CLR$(YELLOW,RED) " ALARM "
```

Если функция используется без параметров (т.е. CLR$()) она сбрасывает цвет на установленный по умолчанию предыдущей командой COLOUR. Цвета также сбрасываются автоматически после завершения команды PRINT.

Эта функция просто генерирует два символа, где первый символ является числом 128 плюс номер цвета переднего плана, а второй символ является числом 192 плюс номер цвета фона. Вы можете использовать этот трюк для встраивания цветовых команд в любом тексте, даже в тексте, считываемом из файла на SD карте.

Цветовые режимы

Видеосистема может быть настроена на один из четырех режимов с помощью команды MODE. Это позволяет программисту добиться компромисса между количеством используемых цветов, графическим разрешением и объемом памяти, требуемой видеодрайвером. Режимы 1 и 4 используют наименьшее количество памяти, режим 3 использует наибольшее. Синтаксис команды MODE:

`MODE colour-mode, palette` ‘«colour-mode» может быть одним из четырех чисел:

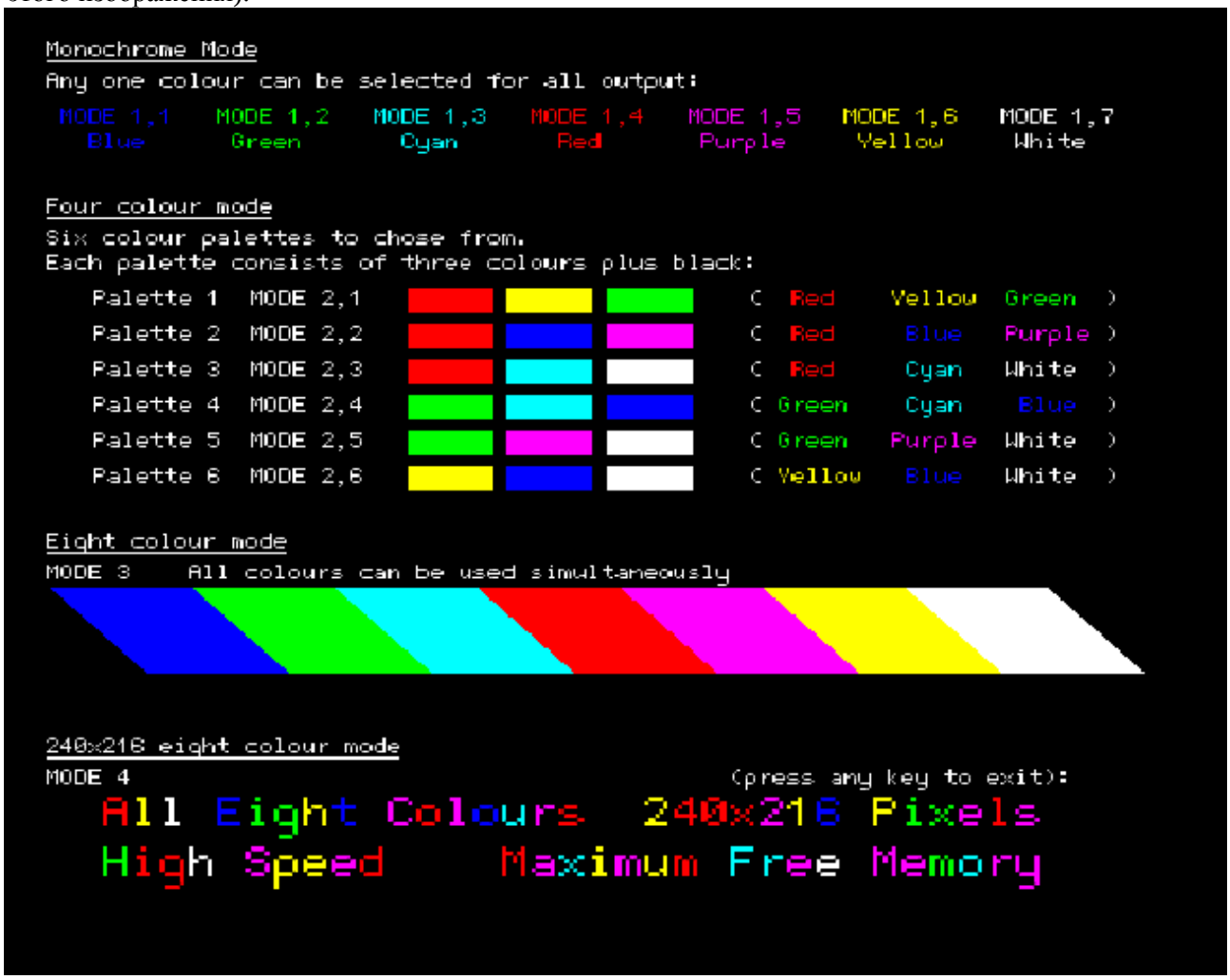
1 Монохромный режим. В этом режиме цветной MMBasic работает так же как монохромный и имеет максимум доступной памяти для программ и данных. Второй аргумент (palette) команды MODE выбирает цвет, который будет использован для всего экрана. Это может быть любой цвет от черного до белого.

2 Четырехцветный режим. В этом режиме доступно 4 цвета, включая черный. Используемые цвета выбираются числами от 1 до 6 во втором аргументе команды MODE (см. рисунок ниже или описание команды MODE)

3 Режим с восемью цветами. В этом режиме все цвета доступны и могут быть использованы одновременно в любом месте экрана. Аргумент «palette» не требуется, и будет игнорироваться, если указан. Режим 3 использует наибольший объем памяти, но её остается достаточно много для программ и данных. Этот режим выбран по умолчанию при включении цветной версии Maximite

4 Режим со сниженным разрешением до 240x216 пикселей. В этом режиме все восемь цветов доступны, а разрешение видео уменьшается вдвое (это означает, что символы и графика будут увеличены два раза). Этот режим лучше всего подходит для игр, так как он оставляет максимальное количество свободной памяти, и вывод графики выполняется очень быстро. Аргумент «palette» не требуется, и будет игнорироваться, если указан.

На рисунке иллюстрация того, как выглядят цветовые режимы (Был использован режим 3 для создания этого изображения):



Вы можете изменить режим и палитру в любое время и так часто, как Вам нужно, даже во время работы программы.

Переопределение цвета линий развертки

В монохромном режиме (режим 1) есть дополнительная возможность для изменения цвета каждой горизонтальной линии пикселей на экране с помощью команды SCANLINE. Эта функция предназначена в первую очередь для программистов, пишущих игры и предоставляет ограниченный контроль над цветом, обеспечивая при этом максимальное количество свободной памяти. Синтаксис команды:

```
SCANLINE colour, startline, endline
```

Эта команда может быть использована только в режиме 1, 7 (монохромный с белым цветом) и используется для установки цвета каждой горизонтальной линии пикселей на экране. «colour» – любой из 8 цветов. «startline» – начальная линия, которая будет окрашена выбранным цветом, «endline» – последняя. Линии развертки нумеруются с 0 вверху до 431 внизу экрана. Нумерация та же, что и для вертикальных координат пикселей.
Вы можете комбинировать команды SCANLINE чтобы установить несколько различных цветов для нескольких линий. Пример:

```
SCANLINE RED, 0, 9          ' сделает верхние 10 линий красными
SCANLINE YELLOW, 120        ' сделает линию 120 желтой
SCANLINE BLUE, 200, 219     ' сделает полосу из 20 строк синими
```

Чтобы отключить переопределение цветов линий, введенное командой SCANLINE, вы можете использовать команду MODE, повторно выбрав режим 1, или переключить систему в другой режим. Оно также автоматически выключается, когда управление возвращается командной строке.

Игровые функции

В MMBasic 4.x добавлен ряд функций, которые предназначены, для облегчения написания игр на Maximite.

MODE 4

Цветной режим 4 описанный в предыдущем разделе, в основном, предназначен для игр. Он предоставляет все восемь цветов и оставляет много свободной памяти для других аспектов анимированной игре (спрайты, массивы и т. д.). Так как этот режим имеет уменьшенное разрешение экрана, графические операции выполняются гораздо быстрее, так как необходимо манипулировать меньшим числом пикселей при выводе на экран.

BLIT

Эта команда будет двигать фрагмент экрана из одного места в другое. Место назначения может перекрывать зону источника и команда BLIT скопирует видеоданные правильно, чтобы избежать сбоев. На цветном Maximite Вы можете также самостоятельно указать, какую цветовую плоскость нужно скопировать. Этот метод перемещения видеоданных гораздо быстрее, чем копирование пикселей один за другим и позволяет осуществлять быструю анимацию на экране. Она также может быть использована для копирования по шаблону (например, границ на кирпичной стене), при построении полного изображения.

Спрайты

Спрайт – это графический образ размером 16x16 бит, который может перемещаться по экрану независимо от фона. Во время отображения спрайта, MMBasic автоматически сохраняет отображаемые на экране текст и графику под спрайтом. Когда спрайт отключается, MMBasic восстановит фон.

Спрайты находятся в файле, который загружается в оперативную память командой SPRITE LOAD,количество спрайтов в файле ограничено только объемом доступной памяти. Каждый спрайт может содержать пиксели любого цвета (в цветном Maximite), также Вы можете использовать прозрачные пиксели, через которые будет видно фон.

Существует специальная функция, которая предупреждает о том, что спрайт столкнулся с краем экрана или другим спрайтом. См. в Приложении H подробное описания создания спрайтов и управления ими.

LOADBMP

Эта команда загрузит цветное или монохромное битовое изображение и отобразит его в заданном месте экрана. Это полезно для загрузки фоновых игровых экранов.

Шрифты

Команда FONT обычно используется для загрузки пользовательских буквенно-цифровых символов, которые могут быть 64 пикселей в высоту и 255 пикселей в ширину.

Это означает, что специально разработанный шрифт может быть использован для загрузки специально созданных графических изображений и отображения их в любом месте экрана на большой скорости.

PEEK/POKE

С командами PEEK и POKE Вы можете использовать постоянные ключевые слова для доступа к специальным разделам памяти (например, видеопамять), эти ключевые слова будут действовать и в будущих версиях MMBasic. Это облегчает доступ к внутренним структурам данных в MMBasic.

Функция KEYDOWN

Функция KEYDOWN позволяет легко определять то, что пользователь удерживает кнопку на PS/2 клавиатуре (например, кнопку курсора). Пока кнопка нажата, функция KEYDOWN будет возвращать числовое значение соответствующее нажатой кнопке, если ни одна кнопка не нажата, будет возвращаться нулевое значение.

Функция KEYDOWN также удалит любые символы в буфере ввода клавиатуры во время игры, однако пользователь зачастую не убирает пальцы с клавиш, даже когда игра заканчивается. По этой причине программа должна включать следующую строку, которая будет ждать, пока пользователь отпустит клавиши, и очистит буфер

```
DO WHILE KEYDOWN AND INKEY$ <> "": LOOP
```

Подпрограммы и функции

Подпрограммы и функции полезны для организации программ, так как их легко изменить и читать. Подпрограмма или функция – это просто часть программного кода, которая содержится внутри модуля и может быть вызвана из любой точки в пределах вашей программы. Это так же, как если бы Вы добавили свою собственный команду или функцию в язык.

Например, предположим, что вы хотели бы иметь команду FLASH в MMBasic, её работа будет заключаться в мигании индикатора питания на Maximite. Вы можете задать подпрограмму, следующим образом:

```
Sub FLASH
  Pin(0) = 1
  Pause 100
  Pin(0) = 0
End Sub
```

Затем, в своей программе просто используйте команду FLASH для того, чтобы мигнуть индикатором питания. Например:

```
IF A <= B THEN FLASH
```

Если подпрограмма FLASH находится в памяти программ, можете попробовать запустить её из командной строки, как любую другую команду в MMBasic. Определение подпрограммы FLASH может быть где угодно в программе, но обычно подпрограммы размещаются в начале или конце. Если MMBasic встречает в программе определение подпрограммы, он просто пропустит её.

Аргументы подпрограмм

Подпрограммы могут иметь аргументы (иногда называемые списком параметров). В определении подпрограммы они выглядят так:

```
Sub MYSUB (arg1, arg2$, arg3)
  <statements>
  <statements>
End Sub
```

Когда Вы вызываете подпрограмму, Вы можете задать значения аргументов. Например:

```
MYSUB  23, "Cat", 55
```

Внутри подпрограммы, `arg1` будет иметь значение `23`, `arg2$` – значение `"Cat"`, и т.д. Аргументы действуют внутри подпрограммы как обычные переменные, и будут очищены после завершения подпрограммы. Вы можете иметь переменные с теми же именами, что и в основной программе, отличные от аргументов определенных в подпрограмме (однако есть риск сильно усложнить отладку, так как изменение значений этих переменных повлияет на основную программу).
При вызове подпрограммы Вы можете указать меньше значений, чем требуется. Например:

```
MYSUB  23
```

В этом случае недостающие значения будет приравнены к нулю или пустой строке. Например `arg2$` будет установлен в «» а `arg3` в 0. Это позволяет иметь опциональные значения и если значения не указаны при вызове, вы можете предпринять специальные меры.

Вы можете также оставить пустым значение в середине списка аргументов. Например:

```
MYSUB  23, , 55
```

Значение аргумента `arg2$` будет установлено в «».

Локальные переменные

Внутри подпрограммы Вам могут потребоваться переменные для разных задач. Если Вы не хотите чтобы в переносимом коде имя, выбранное для внутренней переменной, конфликтовало с переменной с тем же именем в основной программе, Вы можете определить переменную как локальную (LOCAL).

Например, эта подпрограмма FLASH расширена нами, чтобы принимать аргумент (nbr) который показывает сколько раз надо моргнуть индикатором питания:

```
Sub FLASH ( nbr )
  Local count
  For count = 1 To nbr
    Pin(0) = 1
    Pause 100
    Pin(0) = 0
```

```
    Pause 150
  Next count
End Sub
```

Переменная, используемая для счета (`count`) представлена как локальная, это значит, что она (как и список аргументов) существует только внутри подпрограммы и будет очищена после её завершения. Вы можете иметь переменную `count` в Вашей основной программе и она будет отличаться от переменной `count` в Вашей подпрограмме.

Если Вы не определили переменную как локальную, она будет создана автоматически и будет доступна и в основной программе и в подпрограмме, как обычная переменная.

Вы можете сделать несколько переменных локальными одной командой LOCAL. If an item is an array the LOCAL command will also dimension the array (ie, you do not need the DIM command) Если элемент является массивом команда LOCAL сделает локальными и размеры массива (то есть, вам не нужна команда DIM). Например:

```
LOCAL NBR, STR$, ARR(10, 10)
```

Вы так же можете использовать локальные переменные для GOSUB подпрограмм. Например:

```
  GOSUB MySub
       ...
MySub:
   LOCAL X, Y
   FOR X = 1 TO ...
   FOR Y = 5 TO ...
  <statements>
   RETURN
```

Значения X и Y будут доступны до достижения выражения RETURN и будут отличаться от других переменных с аналогичными именами в теле основной программы.

Функции

Функции похожи на подпрограммы. Основным различием является то, что функции используются для возвращения значения выражения. Например, Вам нужна функция для выбора максимального из двух значений, Вы должны её описать следующим образом:

```
Function Max(a, b)
  If a > b
    Max = a
  Else
    Max = b
  EndIf
End Function
```

Затем Вы воспользуетесь ей в выражении:

```
SetPin 1, 1 : SetPin 2, 1
Print "The highest voltage is" Max(Pin(1), Pin(2))
```

Требования к списку аргументов те же, что у подпрограмм. Разница в том, что при вызове функции, список аргументов указывается в скобках, а для вызова подпрограммы скобки не обязательны. Для получения значения из функции, Вы присваиваете это значение имени функции внутри неё. Если имя функции оканчивается символом $, функция возвращает строку, в противном случае она возвращает число.

Внутри функции, её имя действует как обычная переменная.

В качестве другого примера показано использование функции для форматирования времени в AM/PM формат:

```
Function MyTime$(hours, minutes)
  Local h
  h = hours
  If hours > 12 Then h = h - 12
  MyTime$ = Str$(h) + ":" + Str$(minutes)
  If hours <= 12 Then
    MyTime$ = MyTime$ + "AM"
  Else
    MyTime$ = MyTime$ + "PM"
  EndIf
End Function
```

Как вы можете видеть, имя функции используется как обычная локальная переменная внутри подпрограммы. Только тогда, когда происходит возврат из функции, значение, присвоенное `MyTime$`, становится доступным для выражения, которое её вызвало. Этот пример также показывает, что Вы можете использовать локальные переменные внутри функций, так же как в подпрограммах.

Передача аргументов по ссылке

Если Вы используете обычную переменную (то есть, не выражение) в качестве значения при вызове подпрограммы или функции, аргумент будет указывать на переменную, используемую в вызове и, при любых изменениях аргумента в вашей подпрограмме, будет также изменяться и предоставленная переменная. Это называется передачей аргумента по ссылке.

Например, Вы можете задать подпрограмму для обмена значениями между двумя переменными:

```
Sub Swap a, b
  Local t
  t = a
  a = b
  b = t
End Sub
```

В Вашей программе нужно использовать переменные для обоих аргументов:

```
Swap nbr1, nbr2
```

В результате выполнения, значения переменных `nbr1` и `nbr2` поменяются местами.

Если не требуется возвращение значения через аргумент, Вы не должны использовать аргумент в виде переменной общего назначения внутри подпрограммы или функции. Это связано с тем, что другой пользователь Вашей подпрограммы может невольно использовать переменную для её вызова и эта переменная будет непредсказуемо изменена вашей подпрограммой. Вместо этого гораздо безопаснее переместить аргумент в локальную переменную и манипулировать с ней внутри подпрограммы/функции.

Дополнительные примечания

Для каждой подпрограммы или функции может использоваться только одна команда END SUB или END FUNCTION. Чтобы покинуть подпрограмму до её окончания (т.е. до достижения строки с командой END SUB), Вы должны использовать команду EXIT SUB. Это вызовет эффект аналогичный достижению выражения END SUB.Аналогично используйте EXIT FUNCTION для досрочного выхода из функции.

Вы не можете использовать массивы в списке аргументов подпрограммы или функции, однако при вызове можно указывать их отдельные элементы. Например, это правильный способ вызова подпрограммы перестановки значений (описана выше):

```
Swap dat(i), dat(i + 1)
```

Такая конструкция может использоваться для сортировки массивов.

Загружаемые библиотеки

Использование процедур и функций должно снизить необходимость добавлять специальные команды в MMBasic. Например, было несколько запросов с предложением добавить в язык функции битовых смещений. Теперь Вы можете сделать это самостоятельно. Это функция сдвиг вправо:

```
Function RShift(nbr, bits)
   If nbr < 0 or bits < 0 THEN ERROR "Invalid argument"
   RShift = nbr \ (2^bits)
End Function
```

Теперь Вы можете использовать эту функцию как часть языка:

```
a = &b11101001
b = RShift(a, 3)
```

После запуска этого фрагмента кода, переменная b будет иметь двоичное значение `11101.`

Подпрограммы или функция предназначены для достижения переносимости частей кода, которые можно вставлять в любую программу. Вот почему MMBasic имеет команду LIBRARY, которая позволяет загружать файлы, содержащие пользовательские подпрограммы и функции в память. Эти функции/подпрограммы будут доступны для текущей программы и неотличимы от встроенных команд и функций.

Таким образом, легко создать библиотеку функций для небольших манипуляции, таких как описанные выше, и загружать их в любой программе, где они могут понадобиться. Аналогично можно поступить со специализированными математическими функциями, драйверами для специального оборудования и так далее.

Особенности реализации MMBasic

Преобразования имен

Имена команд, функций, меток, переменных, файлов и т.д. не чувствительны к регистру. Так «Run» и «RUN» являются эквивалентными, «dOO» и «Doo» ссылаются на одну переменную.

Существует два типа переменных – числовые, которые хранят числа с плавающей точкой (например, 45.386), и строковые, которые хранят строки символов (например, «Tom»). Имена строковых переменных оканчиваются символом $ (например, name$).

Имена переменных и метки могут начинаться с буквы или знака подчеркивания и могут содержать любые алфавитные или цифровые символы, точки (.) и подчеркивания (_). Они могут иметь длину до 32 символов. Имена переменных или меток не должны совпадать с именами функций и следующими ключевыми словами: THEN, ELSE, GOTO, GOSUB, TO, STEP, FOR, WHILE, UNTIL, LOAD, MOD, NOT, AND, OR, XOR, AS. Например, выражение `step=5` недопустимо, так как STEP – это ключевое слово. Кроме того, метки не должны совпадать с именами команд.

Константы

Численные константы могут начинаться с цифр (0-9), если константа десятичная, символов &H, если константа шестнадцатеричная, &O – восьмеричная и &B – двоичная. Например, &B1000 – тоже что и десятичная константа 8.

Десятичные константы могут начинаться со знака минус (-) или плюс (+) и оканчиваться символом «E» для указания числа в экспоненциальной форме. Например, 1.6E+4 – это тоже 16000.

Строковые константы должны помещаться в кавычки (“). Например, `“Hello World”`.

Операторы и приоритеты

Операторы выполняются в порядке старшинства. Операторы с одинаковым уровнем (например, + и -) обрабатываются слева направо, в порядке возникновения в программной строке.

Арифметические операторы:

^	Возведение в степень
* / MOD	Умножение, деление, целочисленное деление и модуль (остаток)
+ -	Сложение и вычитание

Логические операции:

NOT	Логическая инверсия значения справа от оператора
<> < > <= =< >= =>	Неравенство, меньше, больше, меньше или равно, меньше или равно (альтернативная версия), больше или равно, больше или равно (альтернативная версия)
=	равенство
AND OR XOR	Конъюнкция (лог. И), дизъюнкция (лог. ИЛИ), исключающее ИЛИ

AND, OR и XOR – побитовые операторы. Например, `PRINT 3 AND 6` даст в результате 2.

Остальные логические операторы дают в результате 0 (ноль), если выражение ложно и 1, если истинно. Например, выражение `PRINT 4>= 5` выведет на экран число 0, а `A=3>2` сохранит 1 в переменной A.

Оператор NOT имеет самый высокий приоритет, он связан со значением, следующим за ним. Для правильной работы, выражение, которое нужно инвертировать, нужно поместить в скобки. Например:

```
IF NOT(A=3 OR A=8) THEN …
```

Строковые операторы:

+	Объединение двух строк
<> < > <= =< >= =>	Неравенство, меньше, больше, меньше или равно, меньше или равно (альтернативная версия), больше или равно, больше или равно (альтернативная версия)
=	равенство

Характеристики реализации MMBasic

Максимальная длина командной строки составляет 255 символов;
Максимальная длина имени переменной или метки составляет 32 символа;
Максимальная размерность массива: 8;
Максимальное количество аргументов команд, которые принимают переменное число аргументов: 50;
Максимальное количество вложенных For ... Next циклов: 20;
Максимальное количество вложенного DO ... LOOP циклов: 20;
Максимальное количество вложенных переходов GOSUB: 100;
Максимальное количество вложенных многострочных IF ... ELSE ... ENDIF команд: 20;
Максимальное количество определяемых пользователем подпрограмм и функций (суммарное): 64;
Числа хранятся и обрабатываются как числа одинарной точности с плавающей точкой;
Максимальное число, которое можно представить это 3.40282347e +38, а минимальное составляет 1.17549435e -38;
Диапазон целых чисел, которыми можно манипулировать без потери точности составляет ± 16777100;
Максимальная длина строки 255 символов;
Максимальное число строк 65000;
Максимальная длина пути доступа к файлу (включая путь к каталогу) составляет 255 символов;
Максимальное количество одновременно открытых файлов: 10 на SD карте и один на внутреннем флеш-накопителе (A:);
Максимальный размер SD-накопителя: 2 ГБ в FAT16 или 2 ТБ с файловой системой FAT32;
Размер внутреннего флеш-накопителя (A:) – 180Кб на монохромном Maximite (и менее в цветной или CAN версии);
Максимальный размер загружаемого шрифта составляет 64 пикселей в высоту, 255 пикселей в ширину, можно использовать до 107 символов.
Максимальное количество файлов библиотеки, которые могут быть загружены одновременно: 8;
Максимальное количество фоновых импульсов, запущенных командой PULSE: 5.

Совместимость

MMBasic реализует большинство команд Microsoft GW-BASIC. Есть многочисленные небольшие различия, связанные с физическими и практическими соображениями, но многие MMBasic команды и функции, по существу аналогичны GW-BASIC. Электронное руководство по GW-BASIC доступно по адресу: http://www.antonis.de/qbebooks/gwbasman/index.html и обеспечивает более подробное описание команд и функций.

MMBasic также реализует ряд современных конструкций программирования, которые описаны в стандарте ANSI для Full BASIC (X3.113-1987) или ISO/IEC 10279:1991. Включая SUB/END SUB, DO WHILE … LOOP и структурированные выражения IF .. THEN … ELSE … ENDIF.

Лицензия

Перевод

Руководство перевел на русский язык Скоморохов Алексей Александрович, аспирант кафедры физики, электротехники и электроники Северо-Кавказского федерального университета, Россия, г. Ставрополь.
По вопросам, касающимся перевода, обращайтесь на почту: askomorohov@mail.ru

Предопределенные переменные (только для чтения)

В центральном столбце указана совместимая платформа (CMM – Colour Maximite, MM – монохромный Maximite и DuinoMite, DOS – версия для Windows).

MM.HRES MM.VRES	CMM MM	Горизонтальное и вертикальное разрешение текущего экрана в пикселях
MM.HPOS MM.VPOS	CMM MM	Текущая горизонтальная и вертикальная позиция (в пикселях), после выполнения последней графической операции или операции вывода текста
MM.VER	CMM MM DOS	Номер версии встроенного ПО в формате aa.bbcc, где aa – главный номер версии, bb – дополнительный номер версии, cc – номер ревизии (обычно нулевой, но A = 01, B = 02, и т.д.).
MM.DEVICE$	CMM MM DOS	Строка, представляющая устройство или платформу, на которой работает MMBasic. В настоящее время эта переменная будет содержать одно из следующих выражений: "Maximite" на стандартном Maximite и совместимых. "Colour Maximite" на цветной версии Maximite и UBW32. "DuinoMite" если запущено на устройстве семейства. "DOS" если запущено в Windows с использованием DOS box. "Generic PIC32" для универсальной версии MMBasic на PIC32.
MM.DRIVE$	CMM MM	Возвращает строку с указанием текущего дискового накопителя, («A:» или «B:»).
MM.FNAME$	CMM MM DOS	Имя файла, которое будет использовано по умолчанию при выполнении команды SAVE. Оно устанавливается командами LOAD, RUN и SAVE.
MM.CMDLINE$	CMM MM DOS	Командная строка, используемая для подразумеваемой команды RUN. См. описание подразумеваемой команды RUN в начале следующей страницы.
MM.ERRNO	CMM MM DOS	Указывает на номер ошибки, имеет нулевое значение, если операция прошла успешно. Зависит от установки OPTION ERROR. Для Maximite (цветного и монохромного) существуют следующие значения MM.ERRNO: 0 = Ошибок нет 1 = Не обнаружена SD карта 2 = SD карта защищена от записи 3 = Не достаточно места 4 = Все записи корневого каталога заняты 5 = Ошибочное имя файла 6 = Невозможно найти файл 7 = Невозможно найти путь 8 = Файл доступен только для чтения 9 = Невозможно открыть файл 10 = Ошибка чтения из файла 11 = Ошибка записи в файл 12 = Не является файлом 13 = Не является папкой 14 = Папка не пуста 15 = Аппаратная ошибка доступа к носителю 16 = Ошибка записи флэш-памяти

Команды

В центральном столбце указана совместимая платформа (CMM – Colour Maximite, MM – монохромный Maximite и DuinoMite, DOS – версия для Windows). Квадратные скобки указывают на то, что параметры или символы опциональны.

Командная строка	CMM MM DOS	**Подразумеваемая команда RUN**. В командной строке можно пропустить ключевое слово RUN и MMBasic будет искать на носителе и в директории по умолчанию программу с совпадающим именем, которая сразу будет запущена. «program» – имя программы. Если разширение не указано, то будет автоматически добавлено расширение .BAS. «command-line» – произвольная последовательность символов, которую можно использовать в новой программе в качестве параметров. Используется переменная MM.CMDLINE$ (только для чтения). Например строка: `SORT INFILE.TXT OUTFILE.TXT` запустит программу SORT.BAS с указанием параметров (именами файлов для обработки), переменная MM.CMDLINE$ будет удерживать строку: "`INFILE.TXT OUTFILE.TXT`" Примечания: - Подразумеваемая команда RUN действует только в командной строке (не внутри запущенной программы). - Если программа с именем «program» уже в оперативной памяти, она будет запускаться напрямую, без перезагрузки с диска (даже если она была отредактирована). - Если имя программы «program» совпадает с внутренней командой MMBasic, внутренняя команда будет выполнена вместо неё. Для предотвращения этого, используйте кавычки. Например: `"PRINT"`. - Если MM.CMDLINE$ возвращает пустую строку, это означает, что командная строка программой не поддерживается и программа запросит параметры, которые ей необходимы. - Чтобы избежать случайной перезаписи программы, с которой Вы работаете, будет сгенерирована ошибка, когда требуется загрузить программу с диска, а программа, находящаяся памяти отредактирована, но не сохранена. Если это произошло (и вы не хотите сохранить программу находящуюся в настоящее время в памяти) используйте команду NEW, чтобы очистить память
‘ (одиночный символ кавычки)	CMM MM DOS	Начинает блок комментария, который игнорируется интерпретатором. Комментарии могут находиться в любом месте строки.
? (символ вопроса)	CMM MM DOS	Сокращенная запись команды PRINT.
AUTO или AUTO start или AUTO start, increment	CMM MM DOS	Запуск режима автоматического ввода строк. Чтобы отключить этот режим, используйте сочетание клавиш Control-C. Без аргументов эта команда принимает строки текста с клавиатуры или USB, и добавляет их в память программы без изменений. Это полезно для добавления строк, которые не имеют номера строк и при вставке программы в эмуляторе терминала. Если указан параметр «start», линии будут начинаться с автоматически генерируемым номером строки. «start» – это номер стартовой строки, а «increment» – размер шага приращения номера строк (по умолчанию 10). Если автоматически сгенерируется число соответствующее уже существующей линии, в памяти он будет начинаться звездочкой (*). В этом случае нажатие клавиши Enter без ввода текста будет сохранять строку в памяти и генерировать следующий номер.

BLIT x1, y1, x2, y2, w, h или BLIT x1, y1, x2, y2, w, h, RGB	CMM MM	Копирует часть видеоэкрана. Координаты источника «x1» и «y1». Координаты области назначения «x2» и «y2». Ширина области экрана, которая будет скопирована «w», высота – «h». Все аргументы команды в пикселях. Исходная область и область назначения могут пересекаться. Если указан дополнительный аргумент «RGB» (только в цветном Maximite), скопированы будут только слои с указанным цветом. Например, при аргументе «GB» будут скопированы только зеленый и синий цветовые слои.
CHAIN file$	CMM MM DOS	Выгружает из памяти текущую программу, загружает новую («file$») в память и запускает её, начиная с первой строки. В отличие от команды RUN, эта команда сохраняет текущее состояние программы (то есть, значение переменных, открытых файлов, загруженных шрифтов, открытых портов COM и т.д.). Единственным исключением является все активные прерывания, которые будут автоматически отключены. Одна программа может перейти в другую, которая затем также может перейти в следующую (или обратно в предыдущую) неограниченное количество раз. То, что программа может быть разбита на модули с помощью этой команды, позволяет писать программы почти неограниченного размера, которые будут выполняться даже с ограниченным объемом памяти. Связь между модулями может быть достигнута путем присвоения значений одной или нескольких переменных, которые затем могут быть прочитаны следующей по цепи программой. Обратите внимание, что еще один способ выполнения большой программы при ограниченном объеме памяти, является использование команды LIBRARY.
CHDIR dir$	CMM MM DOS	Меняет текущую рабочую папку на SD карте card на «dir$» Специальное сочетание «..» вызовет переход на уровень вверх а «.» соответствует текущему каталогу.
CIRCLE (x, y) ,r [,c [, aspect [,F]]]	CMM MM	Выводит на экран окружность с центром в координате «x» и «y», радиусом «r». «c» – цвет (опционально), если не указан, то по умолчанию будет соответствовать цвету заливки экрана. «c» может быть указано как -1, что соответствует инверсии пикселей. Опциональный параметр «aspect» задает соотношение высоты и ширины. Так как пиксели в Maximite прямоугольные, значение параметра: 0.833 даст в результате идеальный круг на большинстве мониторов. Другие значения можно использовать для рисования различных овалов. Если параметр «aspect» не указан, ему будет присвоено значение 1.0, что дает совместимость с предыдущими версиями MMBasic. Опция F в конце аргумента дает заполнение овала, цветом, указанным в параметре «c». См. раздел "Графика и работа с цветом" для получения подробностей по цветам и координатам.
CLEAR	CMM MM DOS	Очищает все переменные и восстанавливает память, используемую ими. См. описание команды ERASE, если требуется удаление определенных массивов переменных.
CLOSE [#]nbr [,[#]nbr] …	CMM MM DOS	Закрывает файл(ы) или последовательный порт/порты, открытые под номером файла «nbr». Символ # добавляется опционально. Также см. описание команды OPEN.
CLOSE CONSOLE	CMM MM	Закрывает последовательный порт, открытый в режиме консоли.
CLS [colour]	CMM MM DOS	Очищает экран и помещает курсор в левый верхний угол экрана. Опция «colour» может использоваться для выбора цвета экрана, после очистки. (Только для цветной версии Maximite).

COLOUR fore [, back] или COLOR fore [, back]	CMM	Устанавливает цвет по умолчанию для команд, которые используются для вывода на экран (PRINT, LINE, и т.д.). Параметр «fore» задает цвет переднего плана, «back» – цвет фона. Параметр для фона задается опционально, если не указан, устанавливается черный. Цвет, который будет отображаться, также зависит от выбранного видеорежима (см. описание команды MODE). См. «Работа с цветом» в начале руководства для получения подробной информации.
CONFIG COMPOSITE NTSC | PAL | DISABLED или CONFIG VIDEO OFF | ON или CONFIG FONT 1 | 2 или CONFIG CASE UPPER | LOWER | TITLE или CONFIG KEYBOARD US | UK | FR | GR | BE | IT | ES или CONFIG TAB 2 | 4 | 8	CMM MM	Перепрограммирование параметров MMBasic. Эта команда отличается от других тем, что изменяет постоянные параметры MMBasic и её достаточно запустить один раз (то есть, настройка сохранится в памяти даже при отключении питания). В большинстве случаев требуется перезапуск для того, чтобы изменения вступили в силу. Опция «COMPOSITE» включает композитный видеовыход и позволяет выбрать соответствующую частоту развертки. По умолчанию «DISABLED» для цветной версии Maximite и DuinoMite. «PAL» для оригинального (монохромного) Maximite. Опция «VIDEO» позволяет включить или выключить видеовыход. Позволяет повысить производительность системы, кроме того, освободившаяся видеопамять становится доступной для использования. Параметр по умолчанию «ON». Опция «FONT» позволяет выбрать шрифт по умолчанию, используемый MMBasic (после выполнения команды требуется перезапуск). По умолчанию – «FONT 1» (размер знакоместа – 10x5 пикселей). Крупный шрифт «FONT 2» (16x11) полезен для небольших дисплеев Опция «CASE» позволяет изменять регистр имен команд при просмотре текста командой «LIST». По умолчанию – «TITLE» (начало команд с заглавной буквы), для возврата к стандарту старых версий MMBasic, можно использовать «CONFIG CASE UPPER» (все буквы команд – заглавные). Опция «KEYBOARD» изменяет раскладку клавиатуры. Доступны: стандартная (US), Великобритания (UK), Франция (FR), Германия (GR), Бельгия (BE), Италия (IT) Испания (ES). По умолчанию «US». Опция «TAB» устанавливает длину табуляции. По умолчанию – 2.
CONTINUE	CMM MM DOS	Продолжает работу программы, которая была остановлена выражением END, ошибкой, или сочетанием клавиш CTRL-C. Программа будет продолжена со следующего выражения, следующего за точкой остановки.
COPY src$ TO dest$	CMM MM DOS	Копирует файл с именем «src$» в другой файл с именем и местом назначения «dest$». «dest$» может указывать место на другом диске (например, A:), что позволяет копировать файлы между дисками.
COPYRIGHT	CMM MM DOS	Выводит список авторов MMBasic и сведения об авторских правах.
DATA constant[,constant]...	CMM MM DOS	Сохраняет численные и строковые константы, которые будут доступны для чтения командой READ. Обычно строковые константы должны быть в кавычках ("). В виде исключения, строки, состоящие из слов и цифр, не воспринимаемых MMBasic как ключевые слова (такие как THEN, WHILE и т.д.), допускается записывать без кавычек. Числовые константы могут содержать выражения, например 5*60.
DATE$ = "DD-MM-YY" или DATE$ = "DD/MM/YY"	CMM MM	Устанавливает дату внутренних часов/календаря. DD, MM и YY – числа, например: `DATE$ = “28-7-2012”` При включении, дата устанавливается в «1-1-2000». В цветной версии Maximite есть часы реального времени, по которым будут синхронизированы внутренние часы при включении питания.

DELETE line DELETE -lastline DELETE startline - DELETE startline - lastline	CMM MM DOS	Удаляет одну или несколько строк программы (если она загружена в оперативную память). Если поставить дефис до номера строки «- lastline», будут удалены строки от начала программы до указанной строки. Если дефис после номера строки «startline -», будут удалены строки от указанной строки до конца программы. Также см. команду NEW.
DIM var(dim) , [var(dim)]... или DIM var$(dim) LENGTH n Примеры: DIM nbr(50) DIM str$(20) DIM a(5,5,5), b(1000) DIM str$(200) LENGTH 20	CMM MM DOS	Задает переменную, представляющую собой массив, который может иметь разную размерность. Переменные могут быть числовыми или строковыми. Можно объявить несколько переменных, разделяя их запятыми. «dim» – список чисел, разделенных запятыми, который помещают в скобки. Числа указывают количество элементов в каждой размерности. Обычно нумерация в каждой размерности начинается с 0, но команда OPTION BASE может изменить номер первого элемента массива на 1. Например: `DIM nbr(10, 20)` задает массив с двумя измерениями с 11 (от 0 до 10) элементами в одном измерении и 21 (от 0 до 20) во втором. Общее число элементов составит 231 и, так как каждое число занимает 4 байта, потребуется 924 байта памяти. Строковым массивам, по умолчанию, выделяется память под 255 символов в каждом элементе, что требует больших объемов памяти. Используйте ключевое слово LENGTH, чтобы указать, сколько требуется памяти для размещения каждого из строковых элементов. Параметр «n» может быть от 1 до 255. Например: `DIM str$(5, 10)` задает массив с 66 элементами, которые займут 16,896 байт памяти, однако если длина строк не будет превышать 20 символов, можно задать массив: `DIM str$(5, 10) LENGTH 20` который займет 1,386 байт памяти. Помните, что часть памяти, выделенной под каждый элемент, будет содержать n + 1 дополнительный байт, используемый для указания действительной длины строки в каждом элементе. Если строка, сохраняемая как элемент массива, содержит больше чем «n» символов, произойдет ошибка. За исключением этого момента, строковые массивы, созданные с использованием ключевого слова LENGTH, можно использовать как обычные массивы строк.
DO <statements> LOOP	CMM MM DOS	Запускает бесконечный цикл, выход из которого возможет только по команде EXIT или с помощью передачи управления за пределы цикла такими командами как GOTO или RETURN (внутри подпрограммы).
DO WHILE expression <statements> LOOP	CMM MM DOS	Цикл с предварительным условием – до тех пор, пока условие «expression» не станет истиннымis (аналогично циклу WHILE-WEND также реализованному в MMBasic). Если условие ложно, выражения не будут выполнены ни разу
DO <statements> LOOP UNTIL expression	CMM MM DOS	Цикл с постусловием. Выполняется до тех пор, пока выражение после UNTIL истинно. Команды внутри цикла будут выполнены однократно даже если условие ложно.
DRIVE drivespec$	CMM MM	Меняет используемый по умолчанию диск. «drivespec$» может быть «A:» или «B:». См. также переменную MM.DRIVE$.
EDIT или EDIT filename или EDIT line-number	CMM MM	Запускает полноэкранный редактор. Можно редактировать программу в оперативной памяти или в файле. Так же можно использовать для просмотра и редактирования текстовых файлов. Если использовать EDIT без параметров, редактор будет работать с памятью. Если указано имя файла «filename», будет отредактирован файл, оперативная память останется без изменений. Курсор будет установлен на строке, которая была последней отредактирована или строке, которая вызвала остановку программы или ошибку.

		Если указан номер строки «line-number», будет запущено редактирование программы в оперативной памяти и курсор будет установлен на указанной строке. Клавиши для редактирования: Курсор влево/вправо – перемещают курсор вдоль строки. Курсор вверх/вниз – перемещают курсор по строкам вверх/вниз. Page Up/Down – переход на предыдущую/следующую страницу. Home/End – курсор в начало/конец строки повторное нажатие перемещает в начало/конец программы. Delete – удаляет символ после курсора, при удалении символа перехода на новую строку, произойдет объединение двух строк Backspace – удаляет символ перед курсором. Insert – переключает режим вставки/перезаписи. Escape – закрывает редактор без сохранения (после подтверждения). F1 – сохраняет текст и закрывает редактор. F2 – сохраняет, закрывает редактор и запускает программу. F3 – запускает функцию поиска. SHIFT F3 – повторяет поиск с использованием введенного текста при поиске по F3. F4 – режим выделения текста (см. ниже). F5 – вставка текста, который был ранее скопирован. CTRL-F – вставка файла в редактируемую программу. В режиме выделения (F4), редактор позволяет, используя курсорные клавиши, выделять текст, который можно удалить, вырезать или скопировать в буфер обмена. Линия статуса в этом режиме будет изменена и будут показаны другие действия, назначенные на функциональные клавиши. В полноэкранном редакторе функциональным клавишам от F1 до F5 назначаются действия отличные от ранее назначенных им действий. После выхода из редактора, функции будут восстановлены. Редактор может работать со строками длинее ширины экрана, но символы за пределами экрана не видны. Вы можете разделить строки на части, сделав переход на новую строку. После Вы можете восстановить строку, удалив символы перехода на новую строку. Все клавиши редактирования работают с эмуляторами терминала VT100, таким образом, редактирование возможно через USB или UART. Редактор был протестирован с программой Tera Term и она рекомендуется для использования. Помните, чтоTera Term должна быть настроена на работу с экраном в 80 столбцов и 36 строк.
ELSE	CMM MM DOS	Вводит условие по умолчанию в многострочную структуру IF. См. описание выражения IF.
ELSEIF expression THEN	CMM MM DOS	Вводит второе условие в многострочную структуру IF. См. описание выражения IF.
ENDIF	CMM MM DOS	Заканчивает многострочную структуру IF. См. описание выражения IF.
END	CMM MM DOS	Заканчивает запущенную программу и возвращает управление командной строке.

END FUNCTION	CMM MM DOS	Метка окончания функции. См. описание команды FUNCTION. Каждая функция должна завершаться выражением END FUNCTION. Используйте выражение EXIT FUNCTION для досрочного выхода. Между END и FUNCTION должно быть не больше одного пробела.
END SUB	CMM MM DOS	Метка окончания подпрограммы. См. описание команды SUB. Каждая подпрограмма должна завершаться выражением END SUB. Используйте выражение EXIT SUB для досрочного выхода. Между END и SUB должно быть не больше одного пробела.
ERASE variable [,variable]...	CMM MM DOS	Удаляет массивы переменных и освобождает память. Используйте команду CLEAR, чтобы удалить все переменные, включая массивы.
ERROR [error_msg$]	CMM MM DOS	Вызывает ошибку и остановку программы. Используется для отладки или отслеживания событий, которые не должны происходить.
EXIT EXIT FOR EXIT FUNCTION EXIT SUB	CMM MM DOS	EXIT вызывает выход из DO...LOOP цикла. EXIT FOR вызывает выход из FOR...NEXT цикла. EXIT FUNCTION вызывает досрочное завершение функции. EXIT SUB вызывает досрочное завершение подпрограммы. Допустимо не более одного пробела между словами.
FILES [fspec$]	CMM MM DOS	Отображает список файлов в текущей директории на SD карте или внутреннем накопителе (A:). Для SD карты (B:) может использоваться опциональный параметр «fspec $»: знак вопроса (?) соответствующий любому символу и звездочка (*) соответствующая любому числу символов. Если параметр не указан, будут отображены все файлы в директории. Например: *.* Найти все записи *.TXT Найти записи с расширением TXT E*.* Найти записи, начинающиеся с E X?X.* Найти записи, чьи имена состоят их трех символов и имеют символ X на первом и последнем месте имени.
FONT [#]nbr или FONT [#]nbr, scale или FONT [#]nbr, scale, reverse	CMM MM	Выбирает шрифт для видеовыхода. «nbr» – номер шрифта в пределах от 1 до 10. Символ # опционален. «scale» – коэффициент умножения размера, в пределах от 1 до 8 (например, 2 – удвоит размер пикселей по вертикали и горизонтали). По умолчанию равен 1. Если параметр «reverse» отличается от нуля, шрифт будет отображаться в инвертированном виде. MMBasic имеет три встроенных шрифта: #1 стандартный шрифт 10x5 пикселей, содержит полный ASCII набор. #2 крупный шрифт 16x11 пикселей, полный набор ASCII. #3 шрифт 30x22 пикселей, содержит цифры от 0 до 9, символы +,-, пробел, запятая и точка. Примеры: `10 FONT #3, 2, 1 ‘` удвоенный размер, инверсия `10 FONT #3, ,0 ‘` переход к нормальному отображению `10 FONT #2 ‘` выбор шрифта #2 Font #1 с масштабом 1, без инверсии установлен по умолчанию при включении и включается каждый раз, когда управление возвращается командной строке. Другой шрифт может быть загружен командой FONT LOAD.
FONT LOAD file$ AS [#]nbr	CMM MM	Загружает файл с именем «file$» и устанавливает его как шрифт «nbr», где номер может быть в приделах от 3 до 10. Символ # опционален. В приложении E описан формат файла с шрифтом.

FONT UNLOAD [#]nbr	CMM MM	Удаляет шрифт «nbr» и освобождает память. Символ # опционален. Вы не можете удалить встроенные шрифты.
FOR counter = start TO finish [STEP increment]	CMM MM DOS	Инициирует цикл FOR-NEXT. Переменной «counter» присваивается значение «start» и она увеличивается на значение «increment» (по умолчанию 1) при каждой итерации steps до тех пор как «counter» достигнет значения «finish». Значение «increment» может быть отрицательным, но обязательно должно быть целым. См. также описание команды NEXT.

FUNCTION xxx (arg1 [,arg2, …])
<statements>
<statements>
xxx = <return value>
END FUNCTION
	CMM
	MM
	DOS
		Определяет вызываемые пользовательские функции.
		«xxx» – имя функции, к которому предъявляются требования аналогичные предъявляемым к именам переменных.
		«arg1», «arg2», и т.д. – аргументы или параметры, передаваемые функции.
		Чтобы возвращать значение функции, в теле функции нужно присвоить это значение имени функции, как обычной переменной.
		Например:
		`FUNCTION SQUARE(a)`
		`  SQUARE = a * a`
		`END FUNCTION`
		Каждое определение должно оканчиваться (только одним!) выражением END FUNCTION. Когда программа достигает этого выражения, она возвращает значение функции выражению, которым она была вызвана Если требуется выход из функции до её окончания, используйте команду EXIT FUNCTION.
		Используйте функцию так же как обычную функцию MMBasic.
		Например:
		`PRINT SQUARE(56.8)`
		При вызове функции, аргументы присваиваются внутренним переменным функции, которые доступны только внутри неё.
		Функция может быть вызвана с различным числом аргументов. Каждый пропущенный аргумент функции будет приравнен нулю или пустой строке. Переменные, указанные в списке аргументов, при вызове функции (т.е., не выражения и не константы) будут передаваться по ссылке. Это означает, что любые изменения соответствующих аргументов внутри функции так же повлияет на указанные при вызове функции переменные.
		Не используйте команды перехода внутрь или из функции, такие как GOTO, GOSUB, и т.д.. В противном случае могут быть непредсказуемые последствия.

GOSUB target	CMM MM DOS	Вызывает переход к подпрограмме, которая обозначена номером строки или меткой. Подпрограмма должна оканчиваться командой RETURN.
GOTO target	CMM MM DOS	Вызывает безусловный переход к строке по её номеру или метке.
IF expr THEN statement или IF expr THEN statement ELSE statement	CMM MM DOS	Оценивает выражение «expr» и выполняет выражение «statement» если «expr» истинно, или переходит к следующей строке если ложно. Опциональное выражение после «ELSE» выполняется, если «expr» ложно. Этот тип выражения IF находится на одной строке. Конструкция «THEN statement» может быть заменена на «GOTO номер строки или метка».

IF expression THEN
<statements>
[ELSE
<statements>]
[ELSEIF expression THEN
<statements>]
ENDIF
	CMM
	MM
	DOS
		Многострочное выражение IF с опциональными ELSE и ELSEIF условиями, обязательно заканчивается выражением ENDIF. Каждый компонент должен находиться на отдельной строке. Выражение «expression» оценивается, и выполняются выражения, следующие за THEN, если выражение истинно, иначе выполняются выражения следующие за ELSE.
		Выражение ELSEIF (если имеется) запускается, когда предыдущее условие не выполнилось, и начинает новую цепь IF с последующими выражениями ELSE и/или ELSEIF, если потребуется. Для завершения многострочной структуры IF, требуется только одно ENDIF.

INPUT ["prompt string$";] list of variables	CMM MM DOS	Позволяет вводить данные с клавиатуры для нескольких переменных. Она вызывает появление в командной строке знака вопроса (?). При вводе необходимо использовать запятые для разделения, если переменных больше чем одна. Например, если используется команда `INPUT a, b, c` Затем, с клавиатуры было введено: `23, 87, 66`, то переменные a = 23, b = 87 и c = 66 Строка «prompt string$» будет выведена перед знаком вопроса, если строка оканчивается запятой (,), если она оканчивается точкой с запятой (;), знак вопроса будет скрыт.
INPUT #nbr, list of variables	CMM MM DOS	Тоже что и выше, за исключением того, что ввод данных выполняется из предварительно открытого для чтения файла с номером «nbr». См. так же описание команды OPEN.
IRETURN	CMM MM	Возвращает из прерывания, которое использовало номер строки или метку. Запускается следующее выражение, которое должно было запуститься, если бы не было вызвано прерывание. Помните, команду IRETURN нельзя использовать, если для обработки прерывания используется пользовательская подпрограмма. В этом случае используйте команды END SUB или EXIT SUB.
KILL file$	CMM MM DOS	Удаляет файл с именем «file$». Должно быть указано и расширение файла. Если «file» – строковая константа, кавычки не обязательны., однако они обязательно должны быть, если команда используется внутри программы. Пример: `KILL "SAMPLE.DAT"`
LET variable = expression	CMM MM DOS	Присваивает переменной значение выражения «expression». LET is Если присвоение не начинается командой LET, она автоматически предполагается.
LIBRARY LOAD $file и LIBRARY UNLOAD $file	CMM MM DOS	Загружает файл библиотеки («file$») в оперативную память. Все пользовательские команды и подпрограммы в файле становятся доступны запущенным программам. Одновременно может быть загружено до 8 файлов. Файл библиотеки аналогичен обычной MMBasic программе, за исключением того, что программный код вне определенных пользователем команд и подпрограмм будет проигнорирован. Библиотека не видима для пользователя (её нельзя просмотреть командой LIST), поэтому она должна быть предварительно отлажена как обычная программа. Обычно файл библиотеки имеет расширение «.LIB» и оно будет автоматически добавлено, если имя файла («$file») не содержит расширения. Библиотеки могут загружаться и выгружаться по желанию. Библиотеки могут загружаться из других библиотек, вложения не ограничены. Любая библиотека может быть выгружена из памяти командой LIBRARY UNLOAD и память станет доступна для использования Нельзя выгружать библиотеку, которая используется текущей программой. Команду можно использовать для загрузки специализированных библиотек, расширяющих функции MMBasic, например драйвера периферийных устройств, которые обеспечивают

		манипуляцию отдельными битами, или особые математические функции. Библиотечный файл загружается только по первой команде загрузки, поэтому допустимо вставлять аналогичные команды загрузки в любую часть программы, где может потребоваться эта библиотека. Другое применение команды LIBRARY – расширение объема доступной памяти путем загрузки необходимых секций кода и выгрузки их, когда задача выполнена и нужно загрузить следующую секцию. Для предотвращения фрагментации памяти, функции, использующие большой объем памяти (как буфер COM порта, массивы), должны быть заданы заранее, до загрузки библиотек, которые нужно будет выгружать.
LINE [(x1 , y1)] - (x2, y2) [,c [,B[F]]]	CMM MM	Выводит на экран линию или прямоугольник. x1,y1 и x2,y2 указывают на начальную и конечную точки линии. Параметр «c» определяет цвет, который по умолчанию соответствует цвету переднего плана. Так же возможна инверсия цвета пикселей, при использовании параметра «-1». (x1, y1) указываются опционально и если они пропущены, вместо них используются последние координаты, использовавшиеся в предыдущей команде. Опция «B» выводит прямоугольник с противоположными углами в координатах (x1,y1) и (x2,y2). Опция «BF» выводит закрашенный прямоугольник. См. раздел «Графика и работа с цветом» для получения подробной информации.
LINE INPUT [prompt$,] string-variable$	CMM MM DOS	Записывает строку, вводимую с клавиатуры, в переменную «string-variable$». Если указана, строка «prompt$» будет выведена перед вводом. В отличие от команды INPUT, LINE INPUT считывает целую строку, вместе с запятыми между элементами данных. Знак вопроса не выводится, если он не является частью строки «prompt$».
LINE INPUT #nbr, string-variable$	CMM MM DOS	Тоже что и команда выше, только ввод осуществляется из ранее открытого файла под номером «nbr». См. описание команды OPEN.
LIST LIST line LIST -lastline LIST startline - LIST startline - lastline	CMM MM DOS	Отображает все строки программы или область строк. Если стоит дефис перед номером строки «– lastline», будут выведены строки от первой до указанной. Если дефис стоит после номера строки «startline –», будут выведены строки от указанной до последней.
LOAD file$	CMM MM DOS	Загружает программу с именем «file$» в оперативную память. Если расширение «.BAS» не указано, оно будет добавлено автоматически.
LOADBMP file$ [, x, y]	CMM MM	Загружает битовое изображение и выводит его на дисплей. «file$» – имя файла, а «x» и «y» – координаты верхнего левого угла изображения. Если координаты не указаны, изображение помещается в левый верхний угол экрана. Если расширение не указано, «.BMP» будет добавлено автоматически. Файл должен быть монохромным или цветным (16 цветов, 4 бита) в несжатом формате BMP. Для создания таких изображений рекомендован Microsoft Paint. См. также команду SAVEBMP.
LOCAL variable [, variables]	CMM MM DOS	Задает список локальных переменных внутри подпрограммы или функции. «variable» может быть и массивом, размерность массива должна быть указана командой DIM. Локальная переменная будет доступна только внутри процедуры и будет удалена, после её завершения. Если локальная переменная имеет имя, совпадающее с именем глобальной переменной (использованной до вызова подпрограммы), она будет защищена от изменения в процессе выполнения процедуры.

LOOP [UNTIL expression]	CMM MM DOS	Метка конца цикла. См. команду DO.

MEMORY	CMM
	MM
	DOS
		Выводит список доступной для использования памяти. Например:
		`15kB (18%) Program (528 lines)`
		`23kB (28%) 52 Variables`
		`17kB (21%) General`
		`28kB (33%) Free`
		Программная память очищается командой NEW, Переменные и общее пространство памяти очищаются многими командами (NEW, RUN, LOAD, и др.), а также особыми командами CLEAR и ERASE.
		Общая память используется шрифтами, буферами ввода/вывода и т.д.

MERGE file$	CMM MM DOS	Добавляет строки программы из файла «file$» в программу, находящуюся в памяти. В отличие от команды LOAD, it does not clear the program currently in memory.
MKDIR dir$	CMM MM DOS	Создает папку «dir$» на SD карте.
MODE mode [, palette]	CMM	Устанавливает число цветов, которые будут отображаться на экране. Параметр «mode» может быть: 1 Монохромный режим. «palette» (от 0 до 7) задает цвет переднего плана. Этот режим дает полную совместимость с программами, написанными для монохромной версии Maximite. 2 Цветной режим, с четырьмя цветами. «palette» (от 1 до 6) позволяет выбрать доступный набор цветов (см. таблицу ниже). 3 Режим с 8 цветами и полным разрешением экрана. Параметр «palette» игнорируется. 4 Режим с 8 цветами и уменьшенным до 240x216 пикселей разрешением. Параметр «palette» игнорируется. Цвета, доступные в режиме 2: palette = 1 Черный, красный, зеленый, желтый palette = 2 Черный, красный, синий, фиолетовый palette = 3 Черный, красный, голубой, белый palette = 4 Черный, зеленый, синий, голубой palette = 5 Черный, зеленый, фиолетовый, белый palette = 6 Черный, синий, желтый, белый Команда MODE позволяет программисту добиться компромисса между числом используемых цветов, разрешением экрана и занятой видеопамятью. Режимы 1 и 4 используются наименьшее число памяти, а режим 3 – наибольшее. См. также «Графика и работа с цветом» в начале руководства.
NAME old$ AS new$	CMM MM DOS	Переименует файл или папку из «old$» в «new$» В отличие от команд, работающих с именами файлов, NAME не воспринимает имя с полным указанием пути к файлу (работает только в текущей директории).
NEW	CMM MM DOS	Удаляет программу из оперативной памяти и очищает все переменные.

NEXT [counter-variable] [, counter-variable], и т.д.	CMM MM DOS	NEXT размещается в конце FOR-NEXT цикла; см. описание FOR. Переменная «counter-variable» указывает, к какому именно циклу относится это выражение. Если «counter-variable» не указана, выражение NEXT будет относиться к ближайшему FOR. Так же возможно указание окончаний нескольких вложенных циклов: `NEXT x, y, z`
ON nbr GOTO | GOSUB target[,target, target,...]	CMM MM DOS	Вызывает переход (GOTO) или вызов подпрограммы (GOSUB) под номером, указанным в «nbr» (число округляется); Если 1, произойдет переход(вызов) по первому адресу, если 2 – по второму и т.д. Адресация может быть по номеру строки или метке.
ON KEY target	CMM MM	Активирует прерывание, которое вызывает переход по адресу «target» (метка или номер строки) с пользовательской подпрограммой, если есть один и более символов в буфере ввода (от клавиатуры или USB консоли). Возврат из прерывания осуществляется выражением IRETURN, за исключением случаев, когда используется пользовательская подпрограмма (в этом случае применимы END SUB или EXIT SUB). Помните, что параметры подпрограммы не могут быть использованы. Все символы, ожидающие в приемном буфере, должны быть считаны в подпрограмме, иначе после выхода из прерывания, следующее прерывание будет запущено автоматически. Для отключения прерывания, используйте: ON KEY 0
OPEN fname$ FOR mode AS [#]fnbr	CMM MM DOS	Открывает файл для чтения или записи. «fname» – имя файла (максимум 8 символов) с опциональным расширением (3 символа) с разделением точкой (.). Оно может начинаться с пути к файлу, например: `"B:DIR1DIR2FILE.EXT"`. «mode» может быть: INPUT, OUTPUT, APPEND or RANDOM. INPUT открывает файл для чтения и выдаст ошибку, если файл не существует. OUTPUT откроет файл для записи и автоматически перезапишет существующий файл с аналогичным именем. APPEND так же открывает файл для записи, но не будет перезаписывать существующий файл; Вместо этого, новые записи будут добавляться в конце файл. Если файла не существует, он будет создан и открыт для записи. Помните, команда APPEND не поддерживается файловой системой на флеш-накопителе (A:). RANDOM откроет файл для полного доступа (чтение и запись), также возможен произвольный доступ с использованием команды SEEK. При открытии, указатель записи/чтения устанавливается в конце файла. См. Приложение I для получения подробной информации «fnbr» – номер файла (от 1 до 10). Символ # опционален. Одновременно может быть открыто до 10 файлов. Команды INPUT, LINE INPUT, PRINT, WRITE и CLOSE так же как функции EOF() и INPUT$() используют «fnbr» для определения номера файла, с которым требуется совершить действие. См. также OPTION ERROR и MM.ERRNO.
OPEN comspec$ AS [#]fnbr или OPEN comspec$ AS console	CMM MM	Откроет последовательный порт для чтения или записи. Доступно два порта (COM1: и COM2:), которые могут быть открыты одновременно. Для полного описания с примерами, см. Приложение A. «comspec$» – параметры последовательного порта, имеющие форму: `"COMn: baud, buf, int, intlevel"` с опциональными «,FC» или «,OC» или «,DE» or «,S2» в конце. COM1: использует pin 15 для приема данных и pin 16 для передачи, если требуется контроль потока, pin 17 используется для сигнала RTS, а pin 18 для CTS. COM2: использует pin 19 для приема данных и pin 20 для передачи в монохромном Maximite, а, в цветной версии Maximite, D0 для приема и D1 для передачи (Разъем Arduino).

		Для DuinoMite см. документ "DuinoMite MMBasic ReadMe.pdf" для получения подробной информации. Если порт открыт с указанием номера «fnbr», он может быть записан или считан любыми командами или функциями, которые используют номер файла. Последовательный порт может быть открыт с параметром «AS CONSOLE». В этом случае, принимаемые данные будут восприниматься аналогично нажатиям клавиш клавиатуры, а все символы, отправляемые в видеовыход, будут передаваться через последовательный порт. Это позволяет управлять MMBasic через последовательный интерфейс.
OPTION BASE 0 или OPTION BASE 1	CMM MM DOS	Устанавливает начальный номер значения в массиве (0 или 1). По умолчанию 0. Команда должна быть использована до объявления массивов.
OPTION BREAK nn	CMM MM	Устанавливает значение для клавиши остановки программы в «nn». По умолчанию, выбирается сочетание CTRL-C, но оно может быть изменено на любую клавишу (например, `OPTION BREAK 156` переназначит клавишу F12 на выполнение остановки программы). Установка недопустимого числа (например, 255) отключит функцию остановки программы.
OPTION ERROR CONTINUE или OPTION ERROR ABORT	CMM MM DOS	Устанавливает реакцию на ошибки при файловом вводе/выводе. Опция CONTINUE позволит MMBasic игнорировать ошибки, связанные с файлами. Программа должна выполнять проверку константы MM.ERRNO для обнаружения ошибок. Опция ABORT возвращает нормальное поведение (т.е., остановка программы и выдача сообщения об ошибке). По умолчанию –ABORT. Помните, эти опции влияют на ошибки, связанные с чтением или записью на диски A: и B:, но не изменяют реакции на другие ошибки.

OPTION PROMPT string$	CMM
	MM
	DOS
		Заменяет символ начала командной строки на указанный в «string$» (она может быть выражением, которое вычисляется при выводе командной строки).
		Например:
		`OPTION PROMPT “Ok “`
		или `OPTION PROMPT TIME$ + “: “`
		или `OPTION PROMPT CWD$ + “: “`
		Максимальная длина командной строки 48 символов. Командная строка сбрасывается на «>» при включении питания, но Вы можете автоматически изменять её, сохранив следующую программу как «AUTORUN.BAS» во внутреннюю память (A:):
		`10 OPTION PROMPT “My prompt:”`
		`20 NEW`

OPTION Fnn string$	CMM
	MM
		Присваивает программируемой клавише «Fnn» функцию, указанную в «string$».
		«Fnn» – клавиша от F1 до F12. Максимальная длина выражения – 12 символов.
		«string$» может быть выражением, которое будет вычисляться во время выполнения команды OPTION. Это часто используется для прибавления ENTER (chr$(13)), или двойных кавычек (chr$(34)).
		Например:
		`OPTION F1 "RUN" + CHR$(13)`
		`OPTION F6 "SAVE " + CHR$(34)`
		`OPTION F10 "ENDIF"`
		Обычно эти команды добавляют в файл AUTORUN.BAS file (см. описание OPTION PROMPT для примера).

OPTION USB OFF или OPTION USB ON	CMM MM	Включает или выключает вывод через USB, который осуществляется командой PRINT. Команда не влияет на прием данных по USB. Обычно используется, когда нужно выводить информацию на экран и через USB раздельно. Опция сбрасывается в ON при возврате управления командной строке.
OPTION VIDEO OFF или OPTION VIDEO ON	CMM MM	VIDEO OFF предотвращает вывод информации на экран (VGA или композитный) при выполнении команды PRINT. Опция VIDEO ON возвращает нормальное функционирование. Обычно используется, когда нужно выводить информацию на экран и через USB раздельно. Опция сбрасывается в ON при возврате управления командной строке.
PAUSE delay	CMM MM DOS	Прерывает выполнение программы на «delay» мс. Число может быть дробным, например 0.2 равнозначно 200 мкс. Максимальная задержка 4294967295 мс (49 дней).
PIN(pin) = value	CMM MM	Для «pin» настроенного как цифровой выход, установит вывод в состояние с низким лог. уровнем, если «value» равно нулю и с высоким, если «value» не равно нулю. Вы можете устанавливать на выходах высокий или низкий уровень до конфигурирования выводов в режим цифрового выхода. Изменения произойдут сразу после выполнения команды SETPIN. Нулевой «pin» подключен к светодиоду на передней панели Maximite, желтому светодиоду у UBW32 и зеленому у DuinoMite. «value» отличное от нуля погасит светодиод, нулевое значение включит его. См. функцию PIN() для чтения состояния вывода и команду SETPIN для его конфигурирования.
PIXEL(x,y) = colour	CMM MM	Установит цвет пикселя в VGA или композитном видеовыходе (или инвертирует его, если указано значение -1). См. раздел «Графика и работа с цветом» для получения сведений о координатах и цветах. См. функцию PIXEL(x,y) для считывания состояния пикселя.
PLAYMOD file [, dur] или PLAYMOD STOP	CMM MM	Воспроизводит синтезированную музыку или звуковые эффекты. «file» указывает на имя файла, который должен находиться на внутреннем накопителе A: и быть в формате.MOD format. «dur» – длительность воспроизведения в миллисекундах. Если не указана, файл будет воспроизводиться циклически. Синтез звука происходит в фоновом режиме и не прерывается работой программ. Команда PLAYMOD STOP вызывает остановку воспроизведения. Помните: Файлы должны находиться во внутренней памяти A: и не будут воспроизводиться с SD карты.
POKE hiword, loword, val или POKE keyword, offset, val	CMM MM DOS	Установит байт в виртуальной памяти PIC32. Адрес задается группами по 16 бит: «hiword» – старшая половина адреса, «loword» – младшая. Адресация может быть в виде базового адреса в виде ключевого слова «keyword» и смещения «offset» (может быть + и -) относительно него. Ключевое слово может быть: VIDEO (видеобуфер монохромного Maximite) или: RVIDEO, GVIDEO, BVIDEO (видеобуферы красного, синего и зеленого цвета для цветной версии Maximite), PROGMEM (память программ) или VARTBL (таблица переменных). Ключевое слово для буфера ввода клавиатуры – KBUF, позиция в начале буфера – KHEAD, в конце – KTAIL (длина буфера 256 байт). Эта команда только для опытных программистов. PIC32 дает доступ ко всем регистрам управления, внутренней памяти программ и оперативной памяти, таким образом отпадает необходимость в командах INP или OUT. См. PIC32MX5XX/6XX/7XX Family Data Sheet для получения подробной информации об адресном пространстве.

		Помните, что MMBasic сохраняет большинство данных (в т.ч. видео) в виде 32-х битных целых чисел, а PIC32 использует формат с обратным порядком байтов. ПРЕДУПРЕЖДЕНИЕ: Проверка параметров доступа не выполняется, и если Вы, используя эту команду, обратились по неверному адресу, будет выведено сообщение «internal error» и будет выполнена перезагрузка процессора и очистка памяти.
PORT(start, nbr) = value	CMM MM	Устанавливает несколько идущих подряд выводов I/O одновременно (т.е. одной командой). «start» – первый вывод I/O, который будет соответствовать младшему биту в «value» (бит 0). Бит 1 будет управлять выводом, следующим за тем, который указан в «start» и т.д., до номера указанного в «nbr». Он обозначает число используемых выводов I/O, которыми будет управлять команда. Используемые выводы должны быть пронумерованы последовательно, а недоступный или ненастроенный вывод I/O будет пропущен. Например; `PORT(4, 8) = &B10000011` Установит 8 идущих подряд выводов, начиная с четвертого. Выводы 4, 5 и 11 будут иметь высокий лог. уровень, остальные от 6 до 10 – низкий. Эта команда может использоваться для удобного управления параллельными устройствами, такими как LCD дисплеи. Может использовано любое число выводов I/O с номерами от 1 до 23. См. описание функции PORT, для параллельного.
PRINT expression [[,;]expression] … и т.д.	CMM MM DOS	Выводит текст на экран. Может быть выведено несколько выражений, которые могут разделяться: - Запятой (,) для вывода через табуляцию; - Точкой с запятой (;) которая не выводит никаких символов, а только разделяет выводимые выражения; - Ничем или пробелом, что даст результат аналогичный точке с запятой (результаты будут выведены без разделителей). Точка с запятой (;) в конце выражения блокирует автоматический переход на следующую строку. При печати, перед положительным числом добавляется пробел, а перед отрицательным числом пробел отсутствует, но отображается знак минус (-). Целые числа отображаются без десятичной точки, тогда как дробные имеют точку и значащие десятичные цифры после неё. Длинные числа (больше 6 знаков) выводятся в научном формате (мантисса и порядок). Для форматирования чисел используется функция FORMAT$(). Функция TAB() может использоваться для форматирования строк, например для указания табуляции определенного столбца.
PRINT @(x, y) expression или PRINT @(x, y, m) expression	CMM MM	Аналогична команде PRINT, только курсор устанавливается в позиции заданной координатами x и y. Пример: `PRINT @(150, 45) "Hello World"` Функция @ может быть использована в любом месте команды PRINT. Пример:`PRINT @(150, 45) "Hello" @(150, 55) "World"` Функция @(x,y) перемещает курсор в любое место на экране или за его пределами. Например, `PRINT @(-10, 0) "Hello"` покажет только «llo», так как первые два символа будут за пределами экрана. Функция @(x,y) автоматически блокирует переход на новую строку, который обычно происходит, когда курсор достигает правого края экрана. Параметр «m» задает режим вывода текста: m = 0 Обычный текст (белые буквы, черный фон) m = 1 Фон под буквами не отображается (т.е. прозрачный) m = 2 Инверсия видео вывода (черные буквы, белый фон) m = 5 Текущие пикселей будут обращены (прозрачный фон)

PRINT #nbr, expression [[,;]expression] … и т.д.	CMM MM DOS	Аналогична командам, описанным выше, только вывод осуществляется в файл под номером «nbr», открытый в режиме OUTPUT или APPEND. См. команду OPEN.
PULSE pin, width	CMM MM	Запустит генерацию импульсов на выводе «pin» с периодом «width» в миллисекундах. «width» может быть дробной, например, 0.01 будет равно 10 мкс, что позволяет получать очень узкие импульсы. Импульсы отрицательной полярности, относительно постоянного состояния вывода, могут быть сгенерированы, если на выводе предварительно установить высокий лог. уровень. Команда PULSE запустит генерацию отрицательных импульсов. Примечание: вывод «pin» должен быть сконфигурирован как выход. Для импульсов короче 3 мс точность составляет ± 1 мкс. Для импульсов от 3 мс и длиннее точность ± 0.5 мс. Импульсы длиннее 3 мс могут быть запущены в фоновом режиме. До пяти различных импульсов могут работать параллельно, в фоновом режиме. Период фоновых импульсов может быть изменен повторным вызовом команды PULSE с другим параметром. Для остановки генерации, команда PULSE используется с нулевым «width».
PWM freq, ch1, ch2 или PWM STOP	CMM MM	Генерирует импульсы с широтно-импульсной модуляцией (ШИМ) для управления аналоговыми схемами. «freq» – выходная частота (от 20 Гц до 1 МГц) . Частота может быть изменена в любой момент повторным выполнением команды PWM. Вывод будет работать постоянно, в фоновом режиме и может быть остановлен командой PWM STOP. «ch1» and «ch2» – выходные коэффициенты заполнения каналов 1 и 2 в процентах. Если процент близок к нулю на выходе будет узкий положительный импульс, если близок к 100, на выходе будет широкий положительный импульс. Если коэффициент заполнения для одного из каналов не указан, будет использоваться значение заданное ранее. ШИМ сигнал генерируется на звуковом выходе. Частота сигнала синхронизирована с кварцевым генератором PIC32 и генерируется с высокой точностью. Для частот до 50 кГц, коэффициент заполнения имеет точность до 0.1%. Монохромный Maximite оснащен только одним выходом ШИМ/звука, поэтому для него доступен только параметр – «ch1»
QUIT	DOS	Закрывает MMBasic и возвращает управление операционной системе.
RANDOMIZE nbr	CMM MM DOS	Загружает генератор случайных чисел числом «nbr». При включении питания, в генератор случайных чисел загружается число 0, и он будет выдавать некоторую последовательность чисел при каждом обращении. Чтобы изменить эту последовательность, в генератор каждый раз нужно загрузить другое значение «nbr». Хороший способ – использование функции TIMER, например: `100 RANDOMIZE TIMER`
READ variable[, variable]...	CMM MM DOS	Считывает константы заданные выражением DATA и присваивает их указанным именованным переменным. Типы переменных, указанных в выражении READ должны совпадать с типом констант записанных при помощи выражения DATA statements as they are read. См. также DATA и RESTORE.
REM string	CMM MM DOS	Позволяет добавлять комментарии. Помните, что используемая для добавления комментариев в Microsoft BASIC одиночная кавычка, так же поддерживается и является предпочтительной.
RESTORE	CMM MM DOS	Сбрасывает счетчики строки и позиции для выражений DATA и READ на начало программного файла.

RETURN	CMM MM DOS	RETURN заканчивает подпрограмму, вызванную командой GOSUB, и выполняет переход на выражение, следующее за GOSUB.
RMDIR dir$	CMM MM DOS	Удаляет папку «dir$» с SD карты.
RUN [line] [file$]	CMM MM DOS	Запускает программу в памяти. Если указан номер строки, «line», выполнение программы начнется с этой линии. Если указано имя файла «file$», текущая программа будет удалена из памяти, а указанная будет загружена и запущена. Это так же позволяет одной программе загружать и запускать другие. Пример: `RUN "TEST.BAS"` Если расширение не указано, будет автоматически добавлено «.BAS»
SAVE [file$]	CMM MM DOS	Сохраняет программу в текущей рабочей директории под именем «file$». Имя файла опционально и если оно не указано, будет задействовано использовавшееся предыдущими командами SAVE, LOAD или RUN. Пример: `SAVE "TEST.BAS"` Если расширение не указано, будет автоматически добавлено «.BAS»
SAVEBMP file$	CMM MM	Сохраняет текущее изображение, выводимое через VGA или композитный видеовыход в виде BMP файла в текущей директории. Цветная версия Maximite будет сохранять файл в 16-ти цветном формате (4 бита на пиксель). Пример: `SAVEBMP "IMAGE.BMP"` Если расширение не указано, «.BMP» будет добавлено автоматически. Помните, Windows 7 Paint неверно отображает монохромные изображения. Другие графические редакторы работают с монохромными файлами корректно. См. также команду LOADBMP.
SCANLINE colour, start [, end]	CMM	Эта команда может быть использована для изменения цвета одной или нескольких строк пикселей, выводимых на экране монитора в режиме MODE 1,7 (монохромный, с белым передним планом). Команда влияет на все изображение, выведенное и до и после команды SCANLINE. «colour» (от 0 до 7) задает цвет строк. «start» и «end» указывают область линий, на которые будет воздействовать команда. Если «end» не указан, будет изменена только одна линия. Несколько вызовов команды SCANLINE можно использовать для изменения цвета строк, которые изменялись ранее (т.е., изменения накапливаются). Все изменения вызванные командой SCANLINE автоматически отменяются при вызове команды MODE или возврате управления командной строке. См. раздел «Графика и работа с цветом» для получения дополнительной информации.
SEEK [#]fnbr, pos	CMM MM DOS	Устанавливает позицию указателя чтения/записи в файле открытом для произвольного доступа на байте «pos». Первый байт в файле пронумерован единицей, таким образом, `SEEK #5,1` установит указатель чтения/записи в начале пятого файла. См. Приложение I для получения подробной информации по произвольному доступу к данным.

SETPIN pin, cfg	CMM MM	Переключает вывод I/O «pin» в режим «cfg». Монохромный Maximite имеет 20 выводов I/O, пронумерованных от 1 до 20, цветной Maximite, в дополнение к первым двадцати, имеет 20 выводов I/O на разъеме Arduino. Они обозначены от D0 до D13 и от A0 до A5. Доступны режимы: 0 Не активен 1 Аналоговый вход (выводы от 1 до 10, и от A0 до A5) 2 Цифровой вход (для всех выводов) 3 Вход измерения частоты (выводы от 11 до 14) 4 Вход измерения периода (выводы от 11 до 14) 5 Вход счета импульсов (выводы от 11 до 14) 8 Цифровой выход (для всех выводов) 9 Цифровой выход с открытым коллектором (выводы от 11 до 20, и от D0 до D13) В этом режиме функция PIN() также будет возвращать цифровое значение на выводе. Выводы от 11 до 20 и от D0 до D13 могут воспринимать сигналы с уровнем 5 В и устанавливать высокий уровень 5 В через «подтягивающие» резисторы в режиме с открытым коллектором. Остальные выводы имеют максимальное входное напряжение 3.3 В. Для DuinoMite см. «DuinoMite MMBasic ReadMe.pdf» См. функцию PIN() для считывания состояния выводов и выражение PIN()= для управления цифровым выходом. См. команду ниже для настройки прерываний.
SETPIN pin, cfg , target	CMM MM	Настраивает вывод «pin» для вызова прерывания при выполнения условия, указанного в «cfg»: 0 Не активен 6 Прерывание при переходе от низкого уровня к высокому (выводы от 1 до 20, и от D0 до D8) 7 Прерывание при переходе от высокого уровня к низкому (выводы от 1 до 20, и от D0 до D8) «target» – подпрограмма обработки прерывания, которая может быть указана номером строки, меткой или пользовательской подпрограммой. Эти режимы также настраивают вывод как цифровой вход, таким образом можно считывать его состояние функцией PIN(). Выход из подпрограммы обработки прерывания осуществляется выражением IRETURN за исключением случаев использования обычных подпрограмм (В этом случае используются END SUB или EXIT SUB). Помните, что при этом невозможно указать параметры для подпрограммы.
SETTICK period, target [, nbr]	CMM MM	Настраивает периодическое прерывание. Доступно 4 независимых таймера («nbr» = 1, 2, 3 или 4). «nbr» указывается опционально, по умолчанию используется первый таймер. Время между прерываниями – «period» миллисекунд. «target» – номер строки или метка подпрограммы обработки прерывания. См. также IRETURN для выхода из прерывания. Обычная подпрограмма также может быть использована для обработки прерывания, в этом случае выход будет осуществляться командами EXIT SUB или END SUB. Период может задаваться в пределах от 1 до 4294967295 мс (49 дней). Прерывание будет отключено, если параметру «period» присвоить нулевое значение (т.е., `SETTICK 0, 0, 3` отключит прерывание по третьему таймеру).

SPRITE LOAD file или SPRITE ON n, x, y [, colour] или SPRITE MOVE n, x, y [, colour] или SPRITE COPY n1 TO n2 или SPRITE OFF n [, n [, …]] или SPRITE OFF ALL или SPRITE PASTE n, x, y или SPRITE UNLOAD	CMM MM	Загружает или манипулирует спрайтами на экране. Спрайты размером 16x16 пикселей могут перемещаться по экрану без стирания или нарушения фоновой графики. См. Приложение H «Спрайты» для получения подробной информации. В основном эта команда используется для анимации в играх. «n» – номер спрайта, «x» и «y» – координаты спрайта на экране. SPRITE LOAD загрузит файл со спрайтами в оперативную память. Этот файл содержит графические изображения одного или нескольких спрайтов. SPRITE ON отобразит отдельный спрайт, содержащийся в файле. SPRITE MOVE переметит спрайт и восстановит фон на его предыдущем месте. Для ON и MOVE параметр «colour» опционально задает цвет, используемый для фона. Когда используется сплошная заливка фона, использование этого параметра ускоряет работу и не требует особой обработки для перекрытия спрайтов (См. Приложение H). SPRITE COPY копирует битовое изображение спрайта n1 в изображение спрайта n2. Если спрайт назначения («n2») уже включен (т.е. отображается), то эта команда выполняться как последовательность SPRITE OFF, COPY затем снова ON. Эта команда может использоваться для анимации нескольких спрайтов, использующих одинаковые изображения. См. библиотеку MMBasic (http://geoffg.net/maximite.html#Downloads) для примера. SPRITE OFF удалит спрайт с экрана и восстановит фон, который был им закрыт. Несколько спрайтов могут быть быстро удалены одной командой, если их перечислить через запятую т.е: `SPRITE OFF 4, 8, 2, 3.` SPRITE OFF ALL удалит все активные спрайты. SPRITE PASTE простое копирование изображения спрайта на экран. Нет необходимости включать или выключать спрайты, фон под ними не будет сохранен, и они станут его частью. SPRITE UNLOAD отключит все спрайты, выгрузит файл и вернет отводившуюся под них память. Файл может содержать много отдельных спрайтов, которые могут одновременно отображаться на экране и независимо друг от друга перемещаться по нему. Число спрайтов ограничено только доступной памятью. См. также функцию COLLISION() для обнаружения «столкновений» спрайтов.
SUB xxx (arg1 [,arg2, …]) <statements> <statements> END SUB	CMM MM DOS	Вводит подпрограмму, доступную для вызова, аналогично добавлению новой команды в MMBasic на время работы Вашей программы. «xxx» – имя подпрограммы (требования к имени аналогичны требованиям к именам переменных). «arg1», «arg2», и т.д. – аргументы или параметры для подпрограммы. Каждая подпрограмма должна оканчиваться выражением END SUB, при выполнении которого произойдет переход на строку, следующую за той, которая вызвала подпрограмму. Команда EXIT SUB может быть использована для досрочного выхода из подпрограммы. Вы запускаете подпрограмму, используя её имя и аргументы так же, как и обычную команду. Например: `MySub a1, a2` При вызове подпрограммы, каждый аргумент в строке вызова будет присвоен аргументам подпрограммы. Эти аргументы будут доступны только внутри подпрограммы. Подпрограмма может вызываться с различным числом переменных. Пропущенные аргументы будут приравнены к нулю или пустой строке. Переменные, указанные при вызове подпрограммы в качестве аргументов будут переданы по ссылке в подпрограмму. Это означает, что любые изменения аргументов в подпрограмме вызовут соответствующие изменения в этих переменных. Скобки при указании аргументов не обязательны.

SYSTEM command$	DOS	Передает команду «command$» операционной системе. Это может быть любая команда, распознаваемая командной строкой Windows XP/Vista/7. Доступные команды перечислены тут: http://ss64.com/nt Например, чтобы переключить окно в режим отображения синих символов на желтом фоне: `SYSTEM "COLOR 1E"` Помните, что команды запускаются в разных версиях командного процессора и некоторые из них (такие как «CD ..») не будут действовать.
TIME$ = “HH:MM:SS” или TIME$ = “HH:MM” или TIME$ = “HH”	CMM MM	Устанавливает время внутренних часов. MM и SS опциональны и будут установлены в 00, если не указаны. Например: `TIME$ = “14:30”` установит часы на 14:30 с нулевыми секундами. Значение при включении: «00:00:00». В цветной версии Maximite установлены часы реального времени, по ним синхронизируются внутренние часы при каждом включении.
TIMER = мс	CMM MM DOS	Переустанавливает таймер на заданное количество миллисекунд. Обычно используется для сброса таймера, но Вы можете установить любое положительное целое число. См. функцию TIMER для получения подробной информации.
TONE left [, right [, dur]] или TONE STOP	CMM MM	Генерирует непрерывный синусоидальный сигнал на выходе звука. «left» и «right» – частоты, которые устанавливаются отдельно для левого и правого канала. Сигнал генерируется в фоновом режиме (программа продолжает работать после запуска этой команды). «dur» – длительность воспроизведения. Если она не указана, сигнал будет вырабатываться пока не будет остановлен командой TONE STOP или завершением программы. Частота может задаваться в пределах от 1 Гц до 20 кГц с высокой точностью (синхронизация с кварцевым резонатором PIC32). Частота может быть изменена в любой момент, выполнением новой команды TONE. В монохромном Maximite доступна только одна частота («left»), для правого канала указывается произвольное значение или оставляется пустое место.
TROFF	CMM MM DOS	Отключает средство трассировки; См. TRON.
TRON	CMM MM DOS	Включает средство трассировки. Выводит номера строк, начиная с начала программы, в квадратных скобках, при запуске программы. Полезно дня отладки программ.
WATCHDOG timeout или WATCHDOG OFF	CMM MM	Запускает сторожевой таймер, который используется для перезапуска Maximite при возникновении событий или ошибок останавливающих программу. Это важно когда система оказывается без присмотра. «timeout» – время в миллисекундах до принудительной перезагрузки. Эта команда должна размещаться в стратегических местах программы для постоянного перезапуска таймера до того, как он достигнет нуля. Если таймер достиг нуля (по тому, что BASIC программа прекратила работу), PIC32 будет автоматически перезапущен. После перезапуска, MMBasic начнет поиск и автоматический запуск программы с именем «RESTART.BAS», которая может использоваться для восстановления ситуации перед запуском или переходом к основной программе. Команда WATCHDOG OFF используется для отключения сторожевого таймера, который отключен по умолчанию после перезагрузки или включения питания.

XMODEM SEND file$ или XMODEM RECEIVE file$	CMM MM	Передает или принимает файл от удаленного компьютера через протокол XModem. Передача может осуществляться через USB подключение или через последовательный порт, открытый в режиме консоли. «file$» – имя файла (на SD карте или во внутренней памяти) для приема или передачи. Протокол XModem требует наличие взаимодействующей программы на персональном компьютере, подключенной к соответствующему последовательному порту. Передача файлов была протестирована с программой Tera Term, на ОС Windows и она является рекомендуемой. После запуска команды XMODEM в MMBasic, выберете: `File -> Transfer -> XMODEM -> Receive/Send` в меню Tera Term для запуска передачи. Передача может занять до 15 секунд, и если команда XMODEM не будет выполнена, MMBasic вернет управление командой строке через 60 секунд. Скачать Tera Term можно по адресу: http://ttssh2.sourceforge.jp/

Функции

В центральном столбце указана совместимая платформа (CMM – Colour Maximite, MM – монохромный Maximite и DuinoMite, DOS – версия для Windows). Квадратные скобки указывают на то, что параметры или символы опциональны.

ABS(number)	CMM MM DOS	Возвращает модуль числа «number» (т.е. отбрасывается знак минуса).
ASC(string$)	CMM MM DOS	Возвращает ASCII код первого символа в строке «string$».
ATN(number)	CMM MM DOS	Возвращает арктангенс числа «number» в радианах.
BIN$(number)	CMM MM DOS	Возвращает в строковом виде двоичное представление числа «number».
CHR$(number)	CMM MM DOS	Возвращает строковый символ, соответствующий ASCII коду аргумента «number».
CINT(number)	CMM MM DOS	Округляет до ближайшего целого числа (не по правилам арифметики!). Например, 45.47 будет округлено до 45 45.57 – до 46 -34.45 – до -34 -34.55 – до -35 См. также INT() и FIX().
CLR$( ) или CLR$(fg) или CLR$(fg, bg)	CMM	Возвращает строку, содержащую встроенные коды, для выбора цвета в строке. «fg» – цвет переднего плана, «bg» – цвет фона. Если параметры не указаны, оба цвета устанавливаются по умолчанию, которые были заданы последней командой COLOUR. Например, для отображения желтых букв на красном фоне: `PRINT CLR$(YELLOW,RED) " ALARM "` Эта функция генерирует два строковых символа, где первый – символ номер 128 + номер цвета переднего плана, а второй – символ 192+ номер цвета фона.
COLLISION(n, EDGE) или COLLISION(n, SPRITE)	CMM MM	Проверяет наличие «столкновений» между спрайтом «n» и краем экрана или другим спрайтом. См. Приложение H для получения подробной информации. Возвращает: &B0001 «Столкновение» слева от спрайта &B0010 «Столкновение» справа от спрайта &B0100 «Столкновение» сверху &B1000 «Столкновение» снизу Также возможна комбинация результатов. Например, результат &B0101 сообщает о «столкновении» сверху и слева (например, с левым верхним углом экрана).

		Если спрайт накладывается на другой (т.е. 1 или более непрозрачных пикселя одного спрайта оказались поверх пикселей другого), бит &B10000 будет установлен в дополнение к описанным выше.
COS(number)	CMM MM DOS	Возвращает косинус аргумента «number», заданному в радианах.
CWD$	CMM MM DOS	Возвращает текущую рабочую директорию в виде строки.
DATE$	CMM MM DOS	Возвращает текущую дату внутренних часов MMBasic в виде строки в формате «DD-MM-YYYY». Например, “28-07-2012”. Внутренние часы/календарь отслеживают время и дату, учитывая високосные годы. Для установки даты, используйте команду DATE$ =. \
DEG(radians)	CMM MM DOS	Преобразует число «radians» в радианах в градусы.

DIR$(fspec, type)
или
DIR$(fspec)
или
DIR$( )
	CMM
	MM
		Находит на SD карте файлы и возвращает имена найденных.
		«fspec» – спецификация файла с использованием групповых символов, аналогичных используемым в команде FILES. Например «*.*» вернет имена всех найденных файлов, «*.TXT» вернет имена всех найденных текстовых файлов.
		«type» указывает на тип объекта для поиска:
		VOL – поиск только меток тома
		DIR – поиск только имен папок
		FILE – поиск только файлов (по умолчанию, если «type» не указан)
		Функция возвращает первую найденную запись. Для поиска последующих записей, используйте функцию без аргументов, т.е. DIR$( ). Возврат функцией пустой строки значит, что записей больше нет.
		Этот пример будет выводить список всех файлов в папке:
		`f$ = DIR$("*.*", FILE)`
		`DO WHILE f$ <> ""`
		`PRINT f$`
		`f$ = DIR$()`
		`LOOP`
		Эта функция работает только с SD картой, и Вы должны перейти в требуемую папку перед использованием этой функции.

EOF([#]nbr)	CMM MM DOS	Возвращает истинное значение, если позиция чтения в файле «nbr» открытом в режиме INPUT находится в конце файла. Если используется номер «nbr» последовательного порта открытого для чтения, функция возвращает истинное значение, если в буфере нет символов, ожидающих считывания. Символ # опционален. См. также команды OPEN, INPUT и LINE INPUT и функцию INPUT$.
EXP(number)	CMM MM DOS	Возвращает экспоненту степени «number».

FIX(number)	CMM MM DOS	Отбрасывает дробную часть числа «number». Например, на аргумент 9.89 будет возвращено 9 и на -2.11 будет возвращено -2. Главное отличие функции FIX от INT в том, что FIX не возвращает следующее меньшее значение для отрицательных чисел, как это делает INT(). Такое поведение обеспечивает совместимость с Microsoft BASIC. См. также CINT().
FORMAT$(nbr [, fmt$])	CMM MM DOS	Возвращает строку соответствующую числу «nbr» в формате указанном в строке «fmt$». Спецификация формата начинается символом % и заканчивается буквой. Остальные символы вне этой конструкции копируются в неизменном виде. Структура спецификации формата: % [flags] [width] [.precision] type Где «flags» может быть: - Выравнивание по левой границе внутри поля заданной ширины 0 Использование «0» в качестве заполняющего символа + Добавление символа «+» для положительных чисел пробел Вызывает отображение пробела на месте знака для положительных чисел. Отрицательные выводятся, как обычно, выводятся со знаком «-». «width» – минимальное количество символов для вывода. Меньшие числа будут дополнены, для больших чисел, это количество будет увеличено. «precision» показывает количество знаков в дробной части числа, которое будет отображаться при выбранном типе e или f или максимальное число значащих цифр при выбранном типе g. Число знаков в дробной части должно быть отделено точкой (.). «type» может быть: g Автоматически форматирует число для лучшего отображения. f Форматирует число с десятичной точкой и последующими цифрами дробной части e Форматирует число в экспоненциальную форму Если использовать заглавные G или E, при экспоненциальном выводе будет отображаться заглавная E. Если спецификация формата не указана, предполагается «%g». Примеры: format$(45) возвращает 45 format$(45, “%g”) возвращает 45 format$(24.1, “%g”) возвращает 24.1 format$(24.1,”%f”) возвращает 24.100000 format$(24.1, “%e”) возвращает 2.410000e+01 format$(24.1,"%09.3f") возвращает 00024.100 format$(24.1,"%+.3f") возвращает +24.100 format$(24.1,"**%-9.3f**") возвращает **24.100 **
HEX$(number)	CMM MM DOS	Возвращает строку в шестнадцатеричном виде, соответствующую числу «number».
INKEY$	CMM MM DOS	Проверяет буферы ввода клавиатуры и USB и если в них один или больше символов ожидают очереди, удаляет первый символ из буфера и возвращает его как первый символ строки. Если буфер ввода пуст, функция возвращает пустую строку (т.е., «»).

INPUT$(nbr, [#]fnbr)	CMM MM DOS	Возвращает строку, состоящую из «nbr» символов, считываемых из файла «fnbr» предварительно открыто в режиме INPUT. Функция будет считывать все символы, включая символ возврата каретки и символ новой строки, без перевода. При чтении из буфера последовательного порта, считываются все ожидающие символы до номера «nbr». Если в буфере нет символов, будет возвращена пустая строка. Символ # опционален. См. также команду OPEN.
INSTR([start-position,] string-searched$, string-pattern$)	CMM MM DOS	Возвращает позицию, в которой строка- шаблон «string-pattern$» совпадает со строкой «string-searched$», начиная с «start-position».
INT(number)	CMM MM DOS	Округляет число до меньшего или равного значения. Например, 9.89 будет округлено до 9, а -2.11 до -3. Такое поведение обеспечивает совместимость с Microsoft BASIC. Функция FIX() округляет отрицательные числа в большею сторону. См. также CINT().
KEYDOWN	CMM MM	Возвращает десятичное ASCII значение клавиши нажатой на PS/2 клавиатуре, ноль будет возвращен, если клавиши не были нажаты. Десятичные значения функциональных и курсорных клавиш приведены в Приложении F. Помните, эта функция работает только с подключенной PS/2 клавиатурой, и использование этой функции удаляет все символы, находящиеся в приемном буфере клавиатуры.
LEFT$(string$, number-of-chars)	CMM MM DOS	Возвращает часть строки «string$» с количеством символов «number-of-chars» начиная слева (начала) строки.
LEN(string$)	CMM MM DOS	Возвращает число символов в строке «string$».
LOC([#]fnbr)	CMM MM DOS	Возвращает текущую позицию указателя чтения/записи в файле «fnbr», открытом в режиме RANDOM. Первый байт файла имеет номер 1. См. Приложение I для получения подробной информации по произвольному доступу к файлу. Для последовательного порта функция будет возвращать количество принятых и ожидающих в приемном буфере байтов. Символ # опционален.
LOF([#]fnbr)	CMM MM DOS	Возвращает длину файла «fnbr» в байтах. Для последовательного порта, возвращает количество свободного места (в символах) остающееся в буфере отправки. Помните, что MMBasic приостановит работу при попытке добавить символ в заполненный буфер отправки и будет ожидать освобождения места для него. Символ # опционален.
LOG(number)	CMM MM DOS	Возвращает натуральный логарифм аргумента «number».

LCASE$(string$)	CMM MM DOS	Возвращает строку «string$», все символы в которой заменены строчными.
MID$(string$, start-position-in-string[, number- of-chars])	CMM MM DOS	Возвращает часть строки «string$» начиная с «start-position-in-string» и длиной «number-of-chars» байт. Если число «number-of-chars» не указано, будут возвращены все символы до конца «string$».
OCT$(number)	CMM MM DOS	Возвращает строку, дающую восьмеричное представление числа «number».
PEEK(hiword, loword) или PEEK(keyword, ±offset)	CMM MM DOS	Возвращает байт из виртуальной памяти PIC32. Адрес задается группами по 16 бит: «hiword» – старшая половина адреса, «loword» – младшая. Адресация может быть в виде базового адреса в виде ключевого слова «keyword» и смещения «offset» (может быть + и -) относительно него. Ключевое слово может быть: VIDEO (видеобуфер монохромного Maximite) или: RVIDEO, GVIDEO, BVIDEO (видеобуферы красного, синего и зеленого цвета для цветной версии Maximite), PROGMEM (память программ) или VARTBL (таблица переменных). Ключевое слово для буфера ввода клавиатуры – KBUF, позиция в начале буфера – KHEAD, в конце – KTAIL (длина буфера 256 байт). Эта команда только для опытных программистов. PIC32 дает доступ ко всем регистрам управления, внутренней памяти программ и оперативной памяти, таким образом отпадает необходимость в командах INP или OUT. См. описание команды POKE для получения дополнительной информации и предупреждений, связанных с доступом к памяти.
PI	CMM MM DOS	Возвращает значение числа π.
PIN(pin)	CMM MM	Возвращает значение, соответствующее логическому уровню на выводе I/O «pin». Ноль означает низкий, а 1 – высокий логический уровень. Аналоговые входы будут возвращать значение измеренного напряжения в формате с плавающей точкой. Входы измерения частоты возвращают частоту в Гц (максимум 200 кГц). Входы измерения периода возвращают период в миллисекундах, а счетный вход возвращает количество подсчитанных импульсов с момента сброса (счет выполняется по нарастающему фронту). Сброс счетного входа выполняется повторным выбором режима счета (даже если он уже настроен). Нулевой вывод (т.е., = PIN(0)) возвращает состояние кнопки на плате (которой выполняется запуск обновления прошивки). Ненулевое значение означает то, что кнопка нажата. См. также команды SETPIN и PIN() =.
PORT(start, nbr)	CMM MM	Возвращает значения нескольких соседних выводов I/O одновременно. «start» – номер I/O вывода, значение которого будет возвращено как нулевой бит двоичного числа. Вывод «start»+1 будет соответствовать первому биту, а «start»+2 – второму и т.д. до «nbr» числа бит. Используемые выводы I/O должны быть пронумерованы последовательно, а любой из них, ненастроенный или настроенный не как цифровой вход, будет возвращать нулевое значение. Эта команда может использоваться для простой организации параллельного обмена данными, например с микросхемами памяти. Может использоваться любое количество выводов I/O от первого до 23-го. См. команду PORT для параллельного вывода данных.

POS	CMM MM DOS	Возвращает текущее положение курсора в строке (в символах).
PIXEL(x, y)	CMM MM	Возвращает цвет пикселя на VGA или композитном видеовыходе. См. раздел «Графика и работа с цветом» для получения описания координат и цветов. См. выражение PIXEL(x,y) = для установки цвета пикселя.
RAD(degrees)	CMM MM DOS	Преобразует градусы «degrees» в радианы.
RIGHT$(string$, number-of-chars)	CMM MM DOS	Возвращает «number-of-chars» символов строки «string$», от её конца.
RND(number)	CMM MM DOS	Возвращает псевдослучайное число в диапазоне от 0 до 0.99999. Число «number» игнорируется. Команда RANDOMIZE используется для перезагрузки генератора случайных чисел.
SGN(number)	CMM MM DOS	Возвращает знак аргумента «number», +1 для положительных чисел, 0 для 0, и -1 для отрицательных чисел.
SIN(number)	CMM MM DOS	Возвращает синус числа «number» (в радианах).
SPACE$(number)	CMM MM DOS	Возвращает строку, состоящую из «number» пробелов.
SPI(rx, tx, clk[, dat[, speed]])	CMM MM	Отправляет и принимает байт информации через интерфейс SPI в режиме ведущего (т.е. MMBasic генерирует тактовые импульсы). «rx» номер вывода для входа данных (MISO) «tx» номер вывода для выхода данных (MOSI) «clk» номер вывода для тактовых импульсов (CLK) «dat» опционально, целочисленное представление 1 байта данных, который будет отправлен через «tx» выход. Если не указано, на выводе «tx» будет удерживаться низкий уровень. «speed» – указываемая опционально частота тактовых импульсов. Задается в виде символов H, M или L, где H – 3 МГц, M – 500 кГц и L – 50 кГц. По умолчанию – H. См. Приложение D для получения полного описания.
SQR(number)	CMM MM DOS	Возвращает квадратный корень числа «number».
STR$(number)	CMM MM DOS	Возвращает строку с десятичным представлением числа «number».

STRING$(number, ascii-value\|string$)	CMM MM DOS	Возвращает строку длиной «number» байт, состоящую из символов, соответствующих первому символу строки «string$» или символов указанных по ASCII коду в «ascii-value».
TAB(number)	CMM MM DOS	Выводит пробелы до тех пор, пока не будет достигнут столбец, обозначенный через «number».
TAN(number)	CMM MM DOS	Возвращает тангенс числа «number» (в радианах).
TIME$	CMM MM DOS	Возвращает в виде строки текущее время из внутренних часов MMBasic в формате “HH:MM:SS” в 24-ти часовом представлении.in 24 hour notation. Например, “14:30:00”. Для установки текущего времени, используйте команду TIME$ = .
TIMER	CMM MM DOS	Возвращает время, прошедшее с момента сброса таймера, в миллисекундах (т.е., 1/1000 сек.). Если сброс не выполнялся, после переполнения, таймер начнет отсчет сначала (через 49 дней). Сброс таймера осуществляется при включении питания и при использовании ключевого слова TIMER как команды.
UCASE$(string$)	CMM MM DOS	Возвращает строку «string$», все символы которой заменены на заглавные.
VAL(string$)	CMM MM DOS	Возвращает значение числа, указанного в виде строки «string$». Если строка «string$» задана неверно, функция будет возвращать 0. Функция распознает префикс &H для шестнадцатеричных, &O для восьмеричных и &B для двоичных чисел.

Устаревшие команды и функции

Эти команды и функции включены для обеспечения совместимости с программами, написанными для Microsoft BASIC. Для написания новых программ, рекомендуется использовать соответствующие команды MMBasic.

В центральном столбце указана совместимая платформа (CMM – Colour Maximite, MM – монохромный Maximite и DuinoMite, DOS – версия для Windows). Квадратные скобки указывают на то, что параметры или символы опциональны.

IF condition THEN linenbr	CMM MM DOS	Для совместимости с Microsoft, в выражении THEN предполагается GOTO для перехода по номеру строки. Метка в этой структуре вызовет ошибку. Для новых рекомендуется: IF condition THEN GOTO linenbr | label
LOCATE x, y	CMM MM	Задает позицию курсора в пикселях, для последующего выполнения команды PRINT, которая будет выводить текст с этого места. В новых программах рекомендуется команда PRINT @(x,y) (См. описание команды PRINT).
PRESET (x, y) PSET (x, y)	CMM MM	Выключает (PRESET) или включает (PSET) пиксель на видеоэкране. В новых программах рекомендуется использовать выражение PIXEL(x,y) = .
SOUND freq или SOUND freq, dur	CMM MM	Генерирует тональный сигнал с частотой «freq» (от 20 Hz до 1 МГц) в течении «dur» миллисекунд. Звук воспроизводится в фоне и не останавливает выполнение программы. Если длительность «dur» не указана, звук будет воспроизводиться до отключения. Если задать нулевое значение «dur», все активные SOUND будут выключены. Эта команда заменена на TONE, которая генерирует синусоидальный сигнал на выходе (вместо прямоугольных импульсов, генерируемых по команде SOUND).
SPC(number)	CMM MM DOS	Команда возвращает строку, содержащую «number» пробелов. Аналогична новой команде SPACE$(), используется только для совместимости с Microsoft.
WHILE expression WEND	CMM MM DOS	WHILE обозначает начало WHILE-WEND цикла. Структура оканчивается WEND, и цикл повторяется до тех пор, пока выражение «expression» истинно. Для новых программ рекомендуется использовать структуру DO WHILE … LOOP.
WRITE [#nbr,] expression [,expression] …	CMM MM DOS	Выводит значения выражений «expression», разделенных запятыми (,). Если «expression» – число, оно вводится без пробелов до и перед ним. Если это строка, она помещается в кавычки (″). Список заканчивается переходом на новую строку Если указан номер «#nbr», вывод будет осуществляться в соответствующий файл, предварительно открытый в режиме OUTPUT или APPEND. См. команду OPEN. Вместо WRITE рекомендуется использовать PRINT.

Приложение A
Использование последовательных портов

Доступно два последовательных порта для асинхронной передачи данных (четыре в DuinoMite). Они обозначаются как COM1:, COM2:, и т.д., и могут быть открыты аналогично открытию файла. После открытия, порту присваивается номер (аналогично открытому файлу на диске) и Вы можете использовать все команды, которые выполняют действия с файлами для чтения/записи информации. Последовательный порт также закрывается командой CLOSE. Например:

```
OPEN "COM1:4800" AS #5     ' открывает первый порт с заданной скоростью 4800 бод
PRINT #5, "Hello"          ' отправляет строку "Hello" через порт
dat$ = INPUT$(20, #5)      ' получает до 20 символов из порта
CLOSE #5                   ' закрывает последовательный порт
```

Команда OPEN

Последовательный порт открывается при использовании следующего выражения:

```
OPEN comspec$ AS #fnbr
```

Формат передачи – 8 бит данных, без контроля четности, один стоп-бит (возможно также два стоп-бита). «fnbr» номер файла, который будет использоваться для обращения. Он должен быть задан в пределах от 1 до 10. Символ # опционален.

«comspec$» – параметры порта, указываются в виде строки (можно использовать строковую переменную). Строка имеет вид:

`"COMn: baud, buf, int, intlevel, FC, DE, OC, S2"`, где:

- «n» – номер порта, COM1: или COM2: (плюс COM3: и COM4: в DuinoMite).
- «baud» – скорость передачи данных, 19200, 9600, 4800, 2400, 1200, 600 или 300 бод. Для COM3 и COM4 (только DuinoMite) может задаваться в пределах от 300 до 460800. По умолчанию – 9600.
- «buf» – размер буфера в байтах. Два таких буфера будут размещены в памяти – для приема и для передачи. Размер по умолчанию – 256 байт.
- «int» – номер строки или метка пользовательской подпрограммы, которая будет вызвана прерыванием, когда последовательным портом будет получена какая-либо информация. По умолчанию прерывания отключены.
- «intlevel» – число символов, которые должны ожидать в приемном буфере для вызова подпрограммы обработки прерывания. По умолчанию – 1 символ.

Все параметры, кроме имени порта (COMn:) опциональны. Если один параметр не указан, все последующие за ним так же должны отсутствовать, для них будут установлены значения по умолчанию.

В конце строки «comspec$» могут быть добавлены четыре опции, FC, DE, OC и S2:

- «FC» включает аппаратный контроль потока данных для RS232. Может использоваться только с портом COM1: и добавляет два дополнительных сигнала – запрос отправки (RTS) и разрешение (CTS), которые используются приемным и передающим устройством для предотвращения потери данных при переполнении буфера.
- «DE» активирует сигнал включения выхода данных (DE) для RS485. Может использоваться только с портом COM1:. Сигнал DE переключается на высокий уровень перед передачей байта и возвращается в состояние с низким уровнем после передачи последнего байта из буфера отправки. Опции DE и FC не могут использоваться одновременно.
- «OC» переключает выходы (Tx и, опционально RTS или DE) в режим с открытым коллектором to be open collector. Опция может использоваться с COM1: и COM2:. По умолчанию используется нормальное напряжение (от 0 до 3.3 В).
- «S2» Включает выдачу двух стоп-битов после передачи каждого символа.

Примеры

Открытие последовательного порта с параметрами по умолчанию:

```
OPEN "COM2:" AS #2
```

Открытие последовательного порта, только с указанием скорости передачи (4800 бод):

```
OPEN "COM2:4800" AS #1
```

Открытие порта с заданной скоростью (9600 бод), объемом буфера 1 кБ, без контроля потока:

```
OPEN "COM1:9600, 1024" AS #8
```

Тоже, что выше, но с включением контроля потока:

```
OPEN "COM1:9600, 1024, FC" AS #8
```

Пример выражения включающего все параметры, в том числе метку программы обработки прерываний, уровень прерывания, контроль потока и активацию открытых коллекторов для выходов:

```
OPEN "COM1:19200, 1024, ComIntLabel, 256, FC, OC" AS #5
```

Распределение выводов I/O

COM1: использует вывод 15 для приема данных (RX), вывод 16 для передачи данных (TX). Если активен контроль потока данных, вывод 17 будет использован для выхода запроса передачи (RTS), а вывод 18 – для входа разрешения передачи (CTS). Если активен сигнал активации выхода данных (DE) для RS485, он будет задействовать вывод 17.
COM2: использует вывод 19 для приема данных (RX) и вывод 20 для передачи (TX) в монохромной версии Maximite, и D0 для приема (RX), D1 для передачи (TX) в цветной версии Maximite.
Для подробностей по DuinoMite см. документ "DuinoMite MMBasic ReadMe.pdf".
При открытии последовательных портов, используемые выводы устанавливаются как входы и выходы автоматически, команды SETPIN и PIN на эти выводы не оказывают влияния до закрытия порта. После закрытия порта (командой CLOSE), все выводы, которые были им задействованы переводятся в неустановленное состояние. Требуется команда SETPIN для переключения их режима.
Полярность сигнала стандартна для устройств работающих с TTL сигналами (не RS232). Уровень в режиме ожидания – высокий, старт-бит имеет низкий уровень, для передачи лог. 1 используется высокий уровень, стоп-биты также имеют высокий уровень. Выводы контроля потока (RTS и CTS) используют низкий уровень для остановки передачи данных и высокий для её разрешения. Эти сигналы позволяют Вам напрямую подключать устройства, такие как GPS приемник EM-408 (в которых используются TTL логические уровни).

Чтение и запись

После открытия последовательного порта, Вы можете использовать любую команду или функцию, которые используют номер файла для записи или чтения. Команда PRINT обычно используется для передачи, а функция INPUT$() – для считывания полученных данных. В функции INPUT$() указывается максимальное число возвращаемых символов, однако в приемном буфере их может быть и меньше. Функция INPUT$() будет возвращать пустую строку, если буфер приема пуст.
Функция LOC() возвращает число символов, ожидающих в буфере приема (т.е. число символов, которое может быть считано функцией INPUT$()). Функция EOF() возвращает истинное значение, если приемный буфер пуст. Функция LOF() возвращает свободное место (число символов) в буфере передачи. При передаче через последовательный порт (используя команду `PRINT #n, dat`), выполнение программы будет приостановлено до появления свободного пространства в буфере передачи, если он был переполнен. Если буфер приема переполнен входящими данными, старая информация будет автоматически удаляться, для размещения новой.
Последовательный порт может быть закрыт командой CLOSE. Это вызовет удаление всех символах в буферах приема и передачи и освобождение памяти, также происходит отмена прерываний (если были включены) и установка выводов порта в неустановленное состояние. Последовательные порты также закрываются командами RUN и NEW.

Прерывания

Подпрограмма прерывания будет работать аналогично подпрограммам, запускаемым по изменению состояния I/O выводов (см. описание на странице 7). Возврат из прерывания осуществляется командой IRETURN, кроме случая использования пользовательских подпрограмм для обработки прерывания (в этом случае используются END SUB или EXIT SUB). Помните, что при вызове подпрограммы для обработки прерывания, невозможно передавать ей параметры.
При использовании прерываний, нужно знать, что MMBasic требуется некоторое время для реакции на них и за это время в буфер приема может поступить еще некоторое количество символов (особенно при высоких скоростях передачи). Например, если Вы используете уровень прерывания в 200 символов, а объем буфера 256 символов, то за время выполнения прерывания, буфер может быть переполнен. Для устранения проблемы, объем буфера должен быть увеличен, например до 512 символов или больше.

Открытие последовательного порта в режиме консоли

Порт в режиме консоли может быть открыт командой:
```
OPEN comspec AS CONSOLE
```
В этом случае, символы, получаемые из порта, будут восприниматься так же, как полученные из клавиатуры, а все символы, выводимые на экран, будут также передаваться через последовательный порт. Это позволяет пользователю использовать терминал на другом конце линии для дистанционного управления MMBasic. Например, через модем.
Только один порт может быть одновременно открыт в режиме `"AS CONSOLE"`, а закрыть его можно только командой CLOSE CONSOLE. Консоль не могут закрыть такие команды как NEW и RUN.

Приложение B
Интерфейс I^2C

Только для Maximite (не доступно в DOS и минимальной версии для PIC32).
Межмикросхемная шина IIC (I^2C) была разработана электронным гигантом Philips для передачи данных между интегральными микросхемами. Эта шина используется многими производителями электронных компонентов и может применяться для объединения различных устройств, таких как память, часы, дисплеи, голосовые модули и т.д. Использование шины I^2C может вызвать затруднения, поэтому, если Вы не собираетесь применять устройства на этой шине, можете спокойно пропустить этот раздел.
Эту реализацию шины разработал Gerard Sexton. Она полностью поддерживает режимы ведущего и ведомого, 10-ти битную адресацию, маскирование адреса, команду общего вызова, а также арбитраж шины (для предотвращения конфликтов шины при наличии нескольких ведущих устройств).
В режиме ведущего доступно два режима: с прерываниями и нормальный. В нормальном режиме, команды передачи и чтения I2C останавливают работу MMBasic до их выполнения или срабатывания таймера (если была использована такая опция). В режиме с прерываниями, отправка или прием данных по шине выполняются в фоновом режиме, позволяя другим командам MMBasic продолжить работу. Когда прием/передача будут окончены, произойдет переход к программе обработки прерывания. Это позволяет Вам устанавливать флаги или выполнять другие действия.
При включении функции I^2C, будут задействованы внешние выводы I/O под номерами 12 и 13, а команды SETPIN и PIN() для этих выводов будут заблокированы. Вывод 12 становится линией данных шины I^2C (SDA), а вывод 13 – линией тактирования (SCL). Оба этих вывода должны быть подключены к «подтягивающим» резисторам (типовое значение 10 кОм для 100 кГц или 2 кОм для 400 кГц). Для получения информации по DuinoMite см. документ "DuinoMite MMBasic ReadMe.pdf".
Нужно понимать, что при работе на частотах шины I^2C выше 150 кГц, проводные соединения между устройствами становятся важны. В идеальном случае, они должны быть насколько это возможно короткими (для уменьшения паразитной емкости), кроме того, линии данных и тактирования не должны проходить рядом, между ними требуется проведение общего провода (для ослабления взаимных помех). Если линия данных нестабильна при высокой частоте или линия тактирования сильно зашумлена, периферийные устройства начинают блокировать шину (обычно, удерживая низкий уровень на линии тактирования). Если Вам не нужны высокие скорости обмена, более безопасной будет частота 100 кГц.
В режиме ведущего доступно четыре команды: I2CEN, I2CDIS, I2CSEND и I2CRCV. Для режима ведомого: I2CSEN, I2CSDIS, I2CSSEND и I2CSRCV. Режимы ведущего и ведомого могут быть запущены одновременно. Когда выполняется команда ведущего, функции ведомого находятся в дежурном режиме, ожидая освобождения шины ведущим. Аналогично, во время выполнения команд ведомого, функции ведущего станут доступны после освобождения шины.
Ведущий и ведомый могут использовать прерывания MMBasic для сообщений об изменении своего состояния. Эти подпрограммы обработки прерываний работают так же как при прерываниях вызванных изменением состояния выводов I/O (См. описание на странице 4). Возврат из прерывания осуществляется по команде IRETURN, за исключением случаев, когда для обработки прерываний используются обычные пользовательские подпрограммы (для выхода из них используются END SUB или EXIT SUB). Помните, подпрограммам, используемым для обработки прерываний невозможно передать параметры. Прерывания, связанные с шиной I^2C должны быть отключены перед использованием команды CHAIN.
Автоматическая переменная MM.I2C содержит результат команд и действий.

Команды I2C в режиме ведущего

I2CEN speed, timeout [, int]	Включает модуль I^2C в режиме ведущего. «speed» – значение от 10 до 400 (для скорости от 10 кГц до 400 кГц). «timeout» – время, в миллисекундах, по истечении которого, выполнение команд отправки/приема данных будет прервано, если оно не будет завершены. Минимальное значение – 100. Нулевое значение отключит таймер (не рекомендуется). «int» – опциональный номер строки или метка подпрограммы для обработки прерывания, которая будет запущена после завершения передачи или приема данных. При этом процесс обмена данными будет происходить в фоновом режиме. Если прерывание не используется, команды приема или передачи будут ожидать завершения операции или срабатывания таймера.

I2CDIS	Отключает ведущий I^2C модуль и возвращает 12 и 13 выводы I/O в неустановленное состояние. Затем они могут быть использованы для обычного применения командой SETPIN. Также будет отправлено стоп-состояние, если шина была занята.

I2CSEND addr, option, sendlen, senddata [,sendata ....]	Отправляет данные ведомому I^2C устройству.
	«addr» – адрес ведомого I^2C устройства.
	«option» – число от 0 до 3:
	1 = удерживать шину после выполнения команды (стоп-условие не будет сформировано на шине)
	2 = обрабатывать адрес как 10-ти битный
	3 = комбинация 1 и 2 (удерживает шину и использует 10-ти битный адрес).
	«sendlen» – число байтов для отправки.
	«senddata» – данные для отправки. Могут быть заданы разными путями (все значения должны находиться в пределах от 0 до 255):
	• Данные могут передаваться индивидуальными байтами. Пример:
	`I2CSEND &H6F, 1, 3, &H23, &H43, &H25`
	• Данные могут быть в виде одномерного массива. Обязательно должен быть указан индекс, так как передается элемент массива, а не весь массив; также будет выполнена проверка пределов значений. Пример:
	`I2CSEND &H6F, 1, 3, ARRAY(0)`
	• Данные могут быть в виде строковой переменной (не константой). Пример:
	`I2CSEND &H6F, 1, 3, STRING$`
	Автоматическая переменная MM.I2C будет содержать результат операции.

I2CRCV addr, bus_hold, rcvlen, rcvbuf [,sendlen, senddata [,sendata ....]]	Получает данные от ведомого I^2C устройства с опциональными возможностями передавать некоторые данные перед началом приема. «addr» – адрес ведомого устройства (10-ти битовая адресация не поддерживается). «bus_hold» – число от 0 до 3 1 = удерживать шину после выполнения команды (стоп-условие не будет сформировано на шине) 2 = обрабатывать адрес как 10-ти битный 3 = комбинация 1 и 2 (удерживает шину и использует 10-ти битный адрес). «rcvlen» – число байтов для приема. «rcvbuf» – переменная для приема данных – одномерный массив или обычная переменная, если для «rcvlen» выбрано значение 1. Индекс массива должен быть указан, также будет выполняться проверка пределов значений. Опционально, Вы можете указать данные, которые будут отправлены перед началом приема. Параметры «sendlen» и «senddata» используются как в команде I2CSEND (т.е., «senddata» может быть константой, массивом или строковой переменной). Примеры: I2CRCV &h6f, 1, 1, avar I2CRCV &h6f, 1, 5, anarray(0) I2CRCV &h6f, 1, 4, anarray(2), 3, &h12, &h34, &h89 I2CRCV &h6f, 1, 3, anarray(0), 4, anotherarray(0) Автоматическая переменная MM.I2C будет содержать результат операции.

Команды I2C в режиме ведомого

I2CSEN addr, mask, option, send_int, rcv_int	Включает модуль I2C в режиме ведомого. «addr» – адрес в режиме ведомого устройства «mask» – маска адреса (биты установленные в 1 всегда будут совпадать) «option» – число от 0 до 3 1 = позволяет MMBasic реагировать на сигнал общего вызова. Если этот сигнал обнаружен, переменная MM.I2C получит значение 4. 2 = обрабатывать адрес как 10-ти битный 3 = комбинация 1 и 2 (воспринимать сигнал общего вызова и использовать 10-ти битный адрес). «send_int» – номер строки или метка подпрограммы обработки прерывания, которое будет вызвано, когда модуль обнаружит, что ведомый ожидает получения данных «rcv_int» – номер строки или метка подпрограммы обработки прерывания, которое будет вызвано, когда модуль получит данные от ведущего.
I2CSDIS	Отключает ведомый I^2C модуль и возвращает 12 и 13 выводы I/O в неустановленное состояние. Затем они могут быть использованы для обычного применения командой SETPIN.
I2CSSEND sendlen, senddata [,sendata ....]	Отправляет данные I^2C ведущему. Эта команда должна использоваться в подпрограмме обработки прерывания (т.е. в «send_int_line», когда ведущий запрашивает данные). В качестве альтернативы, при обработке прерывания может быть установлен соответствующий флаг, а команда будет вызвана из основного цикла программы, при обнаружении флага. «sendlen» – число байтов для отправки. «senddata» – данные для отправки. Они могут быть представлены разными способами, см. описание команды I2CSEND.
I2CSRCV rcvlen, rcvbuf, rcvd	Получает данные от I^2C ведущего. Эта команда должна использоваться в подпрограмме обработки прерывания (т.е. в «rcv_int_line», когда ведущий передает данные). В качестве альтернативы, при обработке прерывания может быть установлен соответствующий флаг, а команда будет вызвана из основного цикла программы, при обнаружении флага. «rcvlen» – максимальное число байтов для приема. «rcvbuf» – переменная для приема данных – одномерный массив или обычная переменная, если для «rcvlen» выбрано значение 1. Индекс массива должен быть указан, также будет выполняться проверка пределов значений. «rcvd» – переменная, которая будет содержать число полученных байт.

I2C автоматическая переменная

MM.I2C	Отображает результат I2C операции: 0 = Команда выполнена без ошибок. 1 = Получен бит отсутствия подтверждения – NACK 2 = Вышло время на выполнение команды 4 = Получен сигнал общего вызова (в режиме ведомого)

Вспомогательные команды I^2C

NUM2BYTE number, array(x) или NUM2BYTE number, variable1, variable2, variable3, variable4	Преобразует число «number» в 4 числа, содержащих отдельные байты числа «number» (MMBasic сохраняет числа в формате с плавающей точкой, длиной в 4 байта). Байты могут возвращаться как отдельные переменные и как четыре элемента массива «array», начиная с индекса «x». См. описание функции BYTE2NUM(), которая выполняет обратное действие.

Вспомогательные функции I^2C

BYTE2NUM(array(x)) или BYTE2NUM(arg1, arg2, arg3, arg4)	Возвращает число, составленное из четырех аргументов, путем последовательного объединения байтов. Байты могут быть взяты из четырех отдельных чисел (arg1 - arg4) или из четырех элементов массива «array», начиная с «x». См. описание команды NUM2BYTE, которая выполняет обратное действие.

Приложение C
Интерфейс 1-Wire

Только для Maximite (не доступно в DOS и минимальной версии для PIC32).
Протокол 1-Wire разработан Dallas Semiconductor для объединения разнообразных устройств на одной сигнальной линии. В основном используется для подключения различных датчиков, например температурных DS18B20 и DS18S20. Эту реализацию протокола разработал Gerard Sexton.

Доступно 4 команды:
OWRESET pin [,presence]
OWWRITE pin, flag, length, data [, data…]
OWREAD pin, flag, length, data [, data…]
OWSEARCH pin, srchflag, ser [,ser…]

Где:
«pin» - вывод I/O используемый для передачи данных.
«presence» - опциональная переменная для определения присутствия устройств на линии
(1 = устройство обнаружено, 0 = устройств не обнаружено)

«flag» - комбинация следующих опций:
1 - отправить сигнал сброса перед командой
2 - отправить сигнал сброса после команды
4 - только отправит/получит бит вместо байта данных
8 - вызывает усиление «подтягивания» вывода после выполнения команды (на выводе устанавливается высокий уровень вместо открытого стока(коллектора))

«length» - длина данных для передачи.
«data» - данные для отправки или получения.
«srchflag» - комбинация следующих опций:
1 - запуск нового поиска
2 - возвращать только устройства в состоянии вызова
4 - поиск устройств запрошенного семейства (первый байт «ser»)
8 - пропустить текущее семейство устройств и возвратить следующее устройство
16 – проверяет, что устройство с серийным номером в «ser» доступно
Если «srchflag» = 0 (или 2), будет возвращен серийный номер следующего найденного устройства
«ser» - серийный номер, состоящий (8 байт), который будет возвращен в результате поиска («srchflag» 4 и 16 используют значения в «ser».

После запуска команды, вывод будет переключен в неустановленное состояние, кроме случая выбора 8 флага опций (усиление «подтягивания»).
Аргументы «data» и «ser» могут быть строками, массивами или набором переменных.
Команды OWRESET или OWSEARCH (и команды OWREAD и OWWRITE) устанавливают переменную MM.OW в 1, если команда выполнена успешно (устройство обнаружено, поиск успешен) или 0 в случае ошибок (устройства не обнаружены, поиск не успешен).

Доступно две вспомогательные функции:
OWCRC8(len, cdata [, cdata…]) Обрабатывает данные в «cdata» и возвращает 8 бит кода CRC
OWCRC16(len, cdata [, cdata…]) Обрабатывает данные в «cdata» и возвращает 16 bit кода CRC
Где:
«len» - длина данных для обработки
«cdata» - данные для обработки
«cdata» может быть строкой, массивом или набором переменных.

Приложение D
Интерфейс SPI

Только для Maximite (не доступно в DOS и минимальной версии для PIC32).
Последовательный периферийный интерфейс (SPI) используется для передачи данных между интегральными схемами.

Интерфейс SPI в MMBasic работает в режиме ведущего (т.е., MMBasic генерирует тактовые импульсы).

Синтаксис функции:

```
received_data = SPI( rx, tx, clk, data_to_send, speed, mode, bits )
```

Где:

- «rx» – вывод для приема данных (MISO)
- «tx» – вывод для отправки данных (MOSI)
- «clk» – вывод для тактовых импульсов, генерируемых MMBasic (CLK)
- «data_to_send» опциональное целочисленное представление данных, передаваемых через вывод «tx». Если не указано, на выводе «tx» будет низкий уровень.
- «speed» опциональная частота тактовых импульсов, которая задается в виде одиночного символа H, M или L, где H соответствует 3 МГц, M – 500 кГц и L – 50 кГц. По умолчанию – H.
- «mode» – опциональное число, соответствующее формату передачи (см. таблицу ниже) По умолчанию – 3.
- «bits» опциональное число, обозначающее количество бит для приема/передачи (от 1 до 23) (ограничено числом бит, которые могут быть сохранены в формате с плавающей запятой). По умолчанию – 8.

Параметры «data_to_send», «speed», «mode» и «bits» опциональны. Если не требуются, их можно пропустить (вставить пробелы между запятыми или отбросить конец списка).

Функция SPI возвращает полученное, в результате выполнения операции, число в целочисленном формате. Помните, что одновременно с передачей, будет выполняться и прием данных из ведомого (которые часто отбрасываются).

Примеры

Использование параметров по умолчанию:

```
A = SPI(11, 12, 13)
```

Указание данных для отправки:

```
A = SPI(11, 12, 13, &HE4)
```

Установка формата передачи данных, но с параметрами «data to send» и «speed» по умолчанию:

```
A = SPI(11, 12, 13, , , 2)
```

Использование всех параметров, включая передачу 12 бит данных:

```
A = SPI(11, 12, 13, &HE4, M, 2, 12)
```

Формат передачи

Старшие значащие биты передаются и принимаются первыми. Формат передачи выбирается параметром «mode», как показано:

«mode»	Описание	CPOL	CPHA
0	Активный высокий уровень сигнала тактирования. Данные считываются по нарастающему фронту, установка выходных данных по спадающему фронту.	0	0
1	Активный высокий уровень сигнала тактирования. Данные считываются по спадающему фронту, установка выходных данных по нарастающему фронту.	0	1
2	Активный низкий уровень сигнала тактирования. Данные считываются по нарастающему фронту, установка выходных данных по спадающему фронту	1	0
3	Активный низкий уровень сигнала тактирования. Данные считываются по спадающему фронту, установка выходных данных по нарастающему фронту.	1	1

Для более подробного описания см.: http://en.wikipedia.org/wiki/Serial_Peripheral_Interface_Bus

Выводы I/O

Перед вызовом этой функции, вывод «rx» должен быть сконфигурирован как цифровой вход, а выводы «tx» и «clk» настраиваются как выходы (обычные или с открытым коллектором) с помощью команды SETPIN. Вывод тактирования должен иметь правильный начальный уровень (функция PIN) перед применением команды SETPIN.

Сигнал активации ведомого SPI устройства этой функцией не генерируется и (если требуется) должен быть сгенерирован командой PIN на одном из незанятых выводов, который будет использован в качестве выхода CS (Chip Select). Функция SPI не «перехватывает контроль» над выводами I/O как последовательные порты и I^2C, таким образом, команда PIN будет оказывать на них влияние. Так как номера выводов I/O могут меняться в каждом случае вызова функции, возможно управление несколькими ведомыми SPI устройствами, подключенными к разным выводам I/O.

Пример

Показана процедура передачи команды 80 (hex) и приема двух байт данных от SPI устройства. Так как режим, скорость и количество бит не указаны, будут использованы значения по умолчанию.

```
SETPIN 18, 2                    ' устанавливает вывод «rx» как цифровой вход
SETPIN 19, 8                    ' устанавливает вывод «tx» как выход
PIN(20) = 1 : SETPIN 20, 8      ' высокий уровень на «clk», затем устанавливает его как выход
PIN(11) = 1 : SETPIN 11, 8      ' вывод 11 будет использован как сигнал активации
PIN(11) = 0                     ' активация ведомого (активный низкий уровень)
junk = SPI(18, 19, 20, &H80)    ' передача команды, принятое игнорируется
byte1 = SPI(18, 19, 20)         ' получение первого байта от ведомого
byte2 = SPI(18, 19, 20)         ' получение второго байта от ведомого
PIN(11) = 1                     ' снятие активации
```

Приложение E
Загружаемые шрифты

Только для Maximite (не доступно в DOS и минимальной версии для PIC32).

Этот раздел описывает формат шрифта, который может быть загружен командой FONT LOAD.

Файл шрифта – простой текстовый файл, содержащий обычные символы, которые вписываются строка за строкой для построения изображения каждого символа. Каждый символ может иметь до 64 пикселей в высоту и 255 пикселей в ширину. Первая незакомментированная строка содержит спецификацию шрифта в виде:

```
height, width, start, end
```

Где «height» и «width» – высота и ширина каждого символа в пикселях, «start» – номер в таблице ASCII, где будет расположен первый символ, и «end» – номер последнего символа. Номера символов должны быть в пределах от 20 до 126 (в десятичном формате). Параметры должны быть разделены запятыми. Например: `16, 11, 48, 57` – спецификация шрифта высотой 16 пикселей и шириной – 11. Первый символ в таблице занимает номер 48 (символ «0»), а последний – 57 (символ «9»).

Последующие строки содержат изображения каждого символа.

Каждая строка определяет горизонтальный ряд пикселей. Пробелы обозначают невключенные пиксели, любые другие символы обозначают включенные пиксели. Если шрифт имеет ширину 11 пикселей, строки должны содержать по 11 символов. Отсутствующие в конце строк символы воспринимаются как пробелы. Первая строка соответствует верхнему ряду пикселей и т.д. Если высота символа составляет 16 пикселей, на каждый символ должно отводиться 16 строк. Для указанного в примере шрифта 16x11 с 10 символами, в файле должно быть 160 строк по 11 символов каждая (вдобавок к строке спецификации вверху).

Строка комментария имеет апостроф (') в качестве первого символа, и может встречаться в любом месте. Строки с комментариями полностью игнорируются, все остальные строки имеют значение.

В следующем примере показано как создать два символа в виде смайлов. Каждый из них имеет размер 11x11 пикселей, первый из них располагается в таблице ASCII на месте символа «0», а второй на месте «1». Для отображения смайла, Ваша программа должна содержать строки:

```
40 FONT LOAD "FACES.FNT" AS #6 ' загрузка шрифта
50 FONT #6                     ' выбор шрифта
60 PRINT "0"                   ' вывод смайла на экран
```

```
'example
'FACES.FNT
11,11,48,49

    XXX
  XX   XX
 XX     XX
XX  X X  XX
X         X
XX X   X XX
 X  XXX  X
  XX   XX
    XXX
'
    XXX
  XX   XX
 XX     XX
XX  X X  XX
X         X
XX  XXX  XX
 X X   X X
  XX   XX
    XXX
```

Приложение F
Особые клавиши

Только для Maximite (не доступно в DOS и минимальной версии для PIC32).

MMBasic генерирует одиночные уникальные символы при нажатии специальных или функциональных символов на клавиатуре. Они показаны в таблице в шестнадцатеричном и десятичном формате:

Клавиша клавиатуры	Код символа (Hex)	Код символа (Dec)
Курсор вверх	80	128
Курсор вниз	81	129
Курсор влево	82	130
Курсор вправо	83	131
Insert	84	132
Home	86	134
End	87	135
Page Up	88	136
Page Down	89	137
Alt	8B	139
F1	91	145
F2	92	146
F3	93	147
F4	94	148
F5	95	149
F6	96	150
F7	97	151
F8	98	152
F9	99	153
F10	9A	154
F11	9B	155
F12	9C	156

Если одновременно с одной из клавиш был нажат «control», к коду символа будет прибавлено число 20 (hex) (это эквивалентно установке бита 5 в коде). Если был зажат «shift», будет добавлено число 40 (hex) (эквивалентно установке бита 6). Если обе эти клавиши были зажаты, будет добавлено 60 (hex). Например, сочетание «Control-PageDown» будет генерировать код A9 (hex). Модификатор «shift» работает только с функциональными клавишами F1 – F12 и будет игнорироваться для других клавиш.

MMBasic транслирует большинство VT100 кодов, генерируемых эмуляторами терминала, такими как Tera Term и Putty в эти коды (исключая модификаторы «shift» и «control»). Это означает, что эмулятор терминала, работающий через USB или последовательный порт, будет генерировать те же коды, что и напрямую получаемые от клавиатуры. Это особенно полезно при использовании команды EDIT.

Приложение G
Настройка эмулятора терминала Tera Term

Только для Maximite (не доступно в DOS и минимальной версии для PIC32).

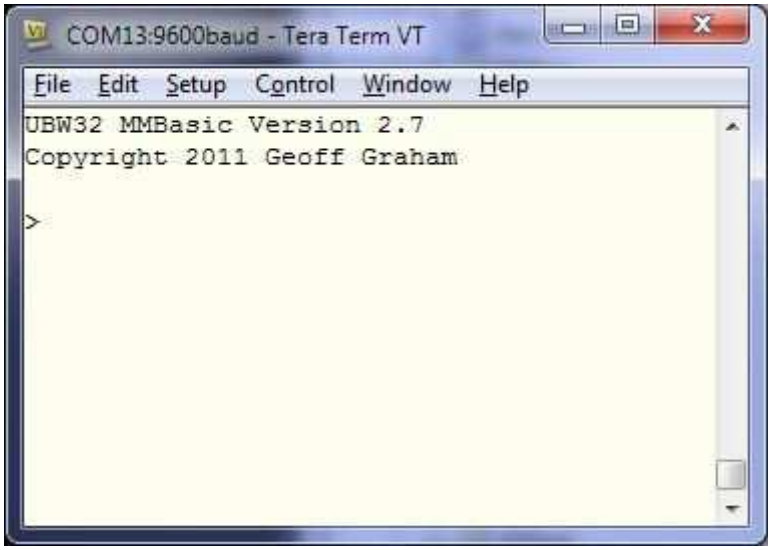
MMBasic создает виртуальный последовательный порт через USB, что дает Вам возможность подключиться к нему через компьютер на Windows, Linux или Macintosh, не используя ничего кроме USB порта.

Используемый протокол связи – CDC (Communication Device Class), стандартный для Linux (драйвер cdc-acm) и Apple OS/X. Пользователи Macintosh могут ознакомиться с документом «Using Serial Over USB on the Macintosh» по ссылке: http://geoffg.net/maximite.html. Остальная часть этого раздела предполагает, что Вы используете Windows XP, Vista или 7.

Для начала, Вам необходимо установить драйвер «Windows Serial Port Driver» (доступен по адресу: http://geoffg.net/maximite.html). Полная инструкция по установке включена в архив с драйвером. Когда Вы выполните все действия, в Диспетчере устройств должно появиться устройство с обозначением последовательного порта (например, COM13).

Для связи с MMBasic через этот виртуальный последовательный порт, Вам нужно использовать эмулятор терминала. Это программа, которая эмулирует терминал стандарта VT100. Существует достаточно много бесплатных эмуляторов, но рекомендуется использовать Tera Term.

1. Вам нужно скачать Tera Term по адресу http://en.sourceforge.jp/projects/ttssh2/releases/ и установить. Эта инструкция основана на версии 4.71.
2. Убедитесь, что USB кабель подключен к компьютеру, и Вы знаете номер виртуального порта.
3. Когда Вы запустите Tera Term, первый раз, появится диалоговое окно, в котором надо указать тип подключения и номер COM порта. Затем Вы увидите окно с приглашением MMBasic:



4. Перед началом использования Tera Term, нужно сделать некоторые изменения в настройках:

   Зайдите в **Setup -> Terminal…**

   Выберите размер окна 80 x 36.

   Отключить опцию «term size = win size».

   Включить «auto window resize».

   В **Setup -> Serial Port…**

   Убедитесь, что номер порта совпадает с предоставленным портом для Maximite.

   В поле «transmit delay msec/line» введите 50. В остальных полях впишите нули.

   Не беспокойтесь о скорости передачи и других настройках.

   Выберете **Setup -> Save Setup…**

   Сохраните настройки как TERATERM.INI в папке, куда был установлен Tera Term, перезаписав уже имеющийся файл.

Приложение H
Спрайты

Только для Maximite (не доступно в DOS и минимальной версии для PIC32).

Спрайт – это графическое битовое изображение размера 16x16, которое может перемещаться по экрану независимо от фона. При отображении спрайта, MMBasic автоматически сохраняет текст и графику на фоне под спрайтом и, при выключении или перемещении спрайта, восстанавливает их.

Спрайты записываются в файлы, которые загружаются в память командой SPRITE LOAD, количество спрайтов в файле ограничено только доступной памятью (в ОЗУ, куда они будут загружаться). Каждый спрайт в файле может содержать пиксели любого цвета (для цветного Maximite) и может также иметь прозрачные пиксели, через которые будет видно фон.

Управление спрайтами

Для управления спрайтами, Вы можете использовать команду SPRITE ON, которая включит отображения определенного спрайта в определенном месте экрана. SPRITE MOVE перемещает спрайт в новое месте и восстанавливает фон на прежнем месте. SPRITE OFF удаляет спрайт с экрана и восстанавливает фон.

Спрайты не должны пересекаться, но если это произошло, вы должны выключить их в обратной, относительно их включения, последовательности, прежде чем снова включите их в их новом месте. Это позволит правильно восстановить фоновое изображение.

Например, при наложении двух спрайтов:

```
SPRITE ON 1, 100, 150      ' спрайт 1 отображается в x = 100, y = 150
SPRITE ON 2, 110, 160      ' спрайт 2 накладывается
```

Для перемещения спрайтов, необходимо выключить их в обратном порядке:

```
SPRITE OFF 2
SPRITE OFF 1
```

Затем они могут быть перемещены в другое место:

```
SPRITE ON 1, 104, 154      ' спрайт 1 отображается в x = 104, y = 154
SPRITE ON 2, 116, 166      ' спрайт 2 все также накладывается
```

Так как спрайты выключаются и включаются очень быстро, пользователь этого не заметит.

Установка фонового цвета

Альтернативой последовательного выключения спрайтов при наложении является указание фонового цвета в командах SPRITE ON или SPRITE MOVE. Фоновый цвет опционален и указывается в конце команд. Например: `SPRITE ON 1, 100, 100, BLUE`

Использование однотонного фона позволяет выполнять операции гораздо быстрее, так как не требуется сохранять фрагменты фона под спрайтами в буфер. Это также позволяет правильно восстанавливать фон при наложении спрайтов.

Обнаружение столкновений

Вы можете использовать функцию COLLISION() для обнаружения столкновения спрайта с другим спрайтом или краями экрана. Столкновение фиксируется, если непрозрачная часть спрайта соприкоснулась (т.е. непрозрачные пиксели оказались соседними) или стали наложены поверх непрозрачной части другого спрайта или края экрана.

Для определения столкновения с другим спрайтом, используйте: `R = COLLISION (n, SPRITE)`

Для обнаружения столкновения с краем экрана, используйте: `R = COLLISION (n, EDGE)`

Где «n» – номер спрайта для проверки.

В обоих случаях, возвращенное значение показывает в каком месте произошло столкновение – слева, справа, снизу или сверху от спрайта. Функция COLLISION () будет возвращать:

&B0001 «Столкновение» слева от спрайта
&B0010 «Столкновение» справа от спрайта
&B0100 «Столкновение» сверху
&B1000 «Столкновение» снизу

Также возможна комбинация результатов. Например, результат &B0101 сообщает о «столкновении» сверху и слева (например, с левым верхним углом экрана). При тестировании на столкновение с другими спрайтами, возможно получение значение &B1111, что сообщает о столкновении со всех сторон. Это может случиться, когда спрайт окружен другими спрайтами.

Если спрайт накладывается на другой (т.е. 1 или более непрозрачных пикселя одного спрайта оказались поверх пикселей другого), бит &B10000 будет установлен в дополнение к описанным выше.

Формат файла для спрайтов

Формат файла со спрайтами похож на формат файла со шрифтом, за исключением того, что спрайты являются битовыми графическими объектами размером 16x16 пикселей. Файл со спрайтами – простой текстовый файл, содержащий обычные символы, которые записываются строка за строкой для построения изображения каждого спрайта. В данный момент, размер каждого спрайта имеет фиксированное значение 16x16, в будущем планируется добавление альтернативных размеров.

Первая незакомментированная строка содержит спецификацию спрайта в виде:

```
dimension, number
```

Где «dimension» – высота и ширина спрайта в пикселях. В данный момент должно быть задано число 16. «number» – количество спрайтов в файле, которое ограничено количеством доступной памяти (в ОЗУ, куда они будут загружаться). Остальные строки формируют растровое изображение каждого спрайта.

Каждая строка соответствует горизонтальному ряду пикселей, где каждый символ указывает на цвет пикселя. Символы – цифры в пределах от 0 to 7 обозначающие цвета от черного до белого, символ пробела означает, что соответствующий пиксель будет прозрачным (т.е., через него будет видно фон). Монохромный Maximite распознает 0 как черный пиксель, а любое другое число – как белый.

Каждый спрайт в файле должен следовать непосредственно за предыдущим и иметь 16 строк, состоящих из 16 символов (Отсутствующие в конце строк символы воспринимаются как пробелы и будут соответствовать прозрачным пикселям).

Строка комментария имеет апостроф (') в качестве первого символа, и может встречаться в любом месте. Строки с комментариями полностью игнорируются, все остальные строки имеют значение.

В следующем примере показан файл, содержащий одиночный спрайт в виде красного круга с белой каймой и синей точкой в центре. Монохромный Maximite будет отображать белый круг:

```
'  example sprite
'  TEST.SPR
16, 1
      7777
    74444447
  744444444447
 74444444444447
 74444444444447
7444444444444447
7444444114444447
7444441111444447
7444441111444447
7444444114444447
7444444444444447
 74444444444447
 74444444444447
  744444444447
    74444447
      7777
```

Приложение I
Произвольный доступ к данным в файле

Приложение описывает осуществление произвольного доступа к данным в файле с фрагментами фиксированной длины. Это возможно при открытии файла в режиме RANDOM, при этом команда SEEK позволяет устанавливать позицию указателя чтения/записи внутри файла.
Для открытия файла используется ключевое слово RANDOM. Например:

```
OPEN "filename" FOR RANDOM AS #1
```

Для поиска фрагмента, используйте команду SEEK, которая перемещает указатель чтения/записи на заданный байт. Первый байт имеет номер 1, таким образом, например, пятый фрагмент в файле, содержащем фрагменты длиной 64 байта каждый, будет начинаться с байта номер 257. Для установки указателя, используйте выражение:

```
SEEK #1, 257
```

При чтении из файла с произвольным доступом, функция INPUT$() должна содержать указание на количество считываемых байтов (например, чтения фрагмента целиком). Например, для чтения фрагмента длиной 64 байта, используйте выражение:

```
dat$ = INPUT$(64, #1)
```

При записи в файл фрагмента, его размер должен быть фиксирован, это условие легко выполняется заполнением свободного места символами (обычно пробелами) перед записью данных. Например:

```
PRINT #1, dat$ + SPACE$(64 – LEN(dat$);
```

Функция SPACE$() используется для добавления достаточного количества пробелов, что обеспечивает требуемую длину фрагмента (64 байта в этом примере). Точка с запятой в конце команды предотвращает добавление символов возврата каретки и перевода строки, которые делают запись длиннее, чем это необходимо.
Две функции позволяют облегчить произвольный доступ. Функция LOC() текущую позицию указателя чтения/записи, а функция LOF() возвращает общую длину файла в байтах.

Следующая программа демонстрирует произвольный доступ к данным в файле. Используя эту программу, Вы можете добавить данные в начале файла, затем считывать/записывать фрагменты, используя произвольные номера фрагментов. Первый фрагмент в файле находится под номером 1, второй – 2 и т.д.

```
RecLen = 64
OPEN "test.dat" FOR RANDOM AS #1
DO
   abort: PRINT
   PRINT "Number of records in the file =" LOF(#1)/RecLen
   INPUT "Command (r = read,w = write, a = append, q = quit): ", cmd$
   IF cmd$ = "q" THEN CLOSE #1 : END
   IF cmd$ = "a" THEN
      SEEK #1, LOF(#1) + 1
   ELSE
      INPUT "Record Number: ", nbr
      IF nbr < 1 or nbr > LOF(#1)/RecLen THEN PRINT "Invalid record" : GOTO abort
      SEEK #1, RecLen * (nbr - 1) + 1
   ENDIF
   IF cmd$ = "r" THEN
      PRINT "The record = " INPUT$(RecLen, #1)
   ELSE
      LINE INPUT "Enter the data to be written: ", dat$
      PRINT #1,dat$ + SPACE$(RecLen - LEN(dat$));
   ENDIF
LOOP
```

Эту программу можно найти в библиотеке MMBasic (http://geoffg.net/maximite.html#Downloads).

Произвольный доступ также можно использовать с обычными текстовыми файлами. Например, эта программа будет переворачивать запись в файле:

```
OPEN "file.txt" FOR RANDOM AS #1
FOR i = LOF(#1) TO 1 STEP -1
   SEEK #1, i
   PRINT INPUT$(1, #1);
NEXT i
CLOSE #1
```