**Инструмент для работы с флеш-памятью ESP32 и ESP8266**

Ссылка на гит:

<https://github.com/espressif/esptool/tree/0ba556f042bd428e62e96891601b1940bc1c0972>

установка

pip install esptool

Чтобы просмотреть сводку всех доступных команд и параметров командной строки, используйте esptool.py -h.

Чтобы увидеть все параметры для конкретной команды, добавьте -h к имени команды. т.е. esptool.py write\_flash -h.

**Общие параметры**

**Серийный порт**

Последовательный порт выбирается с помощью параметра -p, например -p -p COM1 (Windows).

Последовательный порт по умолчанию можно указать, задав переменную среды ESPTOOL\_PORT.

Если не указана опция -p или значение ESPTOOL\_PORT, esptool.py перечислит все подключенные последовательные порты и попробует каждый из них, пока не обнаружит подключенное устройство Espressif (новое поведение в v2.4.0).

**Скорость передачи**

Скорость передачи esptool.py по умолчанию составляет 115200 бит / с. Различные скорости могут быть установлены с помощью -b 921600 (или другой скорости передачи по вашему выбору). Скорость передачи по умолчанию также можно указать с помощью переменной среды ESPTOOL\_BAUD. Это может ускорить операции write\_flash и read\_flash.

Скорость передачи ограничена 115200, когда esptool.py устанавливает начальное соединение, более высокие скорости используются только для передачи данных.

Большинство конфигураций оборудования будут работать с -b 230400, некоторые с -b 460800, -b 921600 и / или -b 1500000 или выше.

Если у вас есть проблемы с подключением, вы также можете установить скорость передачи ниже 115200. Вы также можете выбрать 74880, которая является обычной скоростью, используемой ESP8266 для вывода информации журнала загрузки.

**Команды**

Записать двоичные данные во флеш-память: write\_flash

Двоичные данные могут быть записаны во флеш-чип ESP с помощью команды serial write\_flash:

esptool.py --port COM4 write\_flash 0x1000 my\_app-0x01000.bin

В одной командной строке можно указать несколько адресов флэш-памяти и имена файлов:

esptool.py --port COM4 write\_flash 0x00000 my\_app.elf-0x00000.bin 0x40000 my\_app.elf-0x40000.bin

Аргумент --chip является необязательным при записи во флэш-память, esptool определит тип микросхемы при подключении к последовательному порту.

Аргумент --port задокументирован в разделе Последовательный порт.

Следующими аргументами write\_flash являются *одна или несколько пар* *смещения* (адреса) и *имени файла*. При создании образов ESP8266 версии 1 имена файлов, созданные elf2image, включают смещения флеши как части имени файла. Для других типов образов обратитесь к документации вашего SDK, чтобы определить с каким смещением прошивать файлы.

Числовые значения, передаваемые во write\_flash (и другие команды), могут быть указаны либо в шестнадцатеричном формате (например, 0x1000), либо в десятичном (например, 4096).

Если команда write\_flash завершается ошибкой или прошитый модуль не загружается, см. Раздел «Устранение неполадок»,.

**Установка режима и размера флеш-памяти**

Вам также может потребоваться указать аргументы для *режима* флэш-памяти и *размера* флэш-памяти, если вы хотите изменить значения по умолчанию. Например:

esptool.py --port COM5 write\_flash --flash\_mode qio --flash\_size 32m 0x0 bootloader.bin 0x1000 my\_app.bin

Начиная с esptool v2.0, эти параметры не часто нужны, так как по умолчанию режим и размер флэш-памяти сохраняются из файла образа .bin. См. Раздел «Режимы вспышки» для получения более подробной информации.

**Сжатие**

По умолчанию данные последовательной передачи сжимаются для повышения производительности. Параметр -u/--no-compress отключает это поведение.

**Чтение содержимого флэш-памяти: read\_flash**

Команда read\_flash позволяет считывать содержимое флэш-памяти. Аргументами команды являются адрес, размер и имя файла, куда выводится вывод. Например, чтобы прочитать полные 2 МБ прикрепленной флеш-памяти:

esptool.py -p COM5 -b 460800 read\_flash 0 0x200000 flash\_contents.bin

прочитать всю память из LyraTD-DSPG (16 Mb) с нулевого адреса:

esptool.py -p COM5 -b 460800 read\_flash 0 0x1000000 factory\_lyratd\_dspg.bin

esptool.py -p COM9 -b 460800 read\_flash 0 0x400000 factory\_lyratd\_msc.bin

|  |  |
| --- | --- |
| Двоичная | HEX |
| 16b | 0x10 |
| 32b | 0x20 |
| 64b | 0x40 |
| 128b | 0x80 |
| 256b | 0x100 |
| 512b | 0x200 |
| 1Kb | 0x400 |
| 2Kb | 0x800 |
| 4Kb | 0x1000 |
| 8Kb | 0x2000 |
| 16Kb | 0x4000 |
| 32Kb | 0x8000 |
| 64Kb | 0x10000 |
| 128Kb | 0x20000 |
| 256Kb | 0x40000 |
| 512Kb | 0x80000 |
| 1Mb | 0x100000 |
| 2Mb | 0x200000 |
| 4Mb | 0x400000 |
| 8Mb | 0x800000 |
| 16Mb | 0x1000000 |
|  |  |

(Обратите внимание, что если write\_flash ***обновил*** режим флэш-памяти загрузочного образа и ***размер флэш-памяти во время прошивки***, эти байты могут отличаться при обратном чтении!)

**Стирание флэш-памяти и стирание региона: erase\_flash**

Чтобы стереть весь флеш-чип (все данные заменяются байтами 0xFF):

esptool.py erase\_flash

Чтобы стереть область флеш-памяти, начиная с адреса 0x20000 длиной 0x4000 байт (16 КБ):

esptool.py erase\_region 0x20000 0x4000

*!!Адрес и длина* ***должны быть кратны размеру сектора*** *стирания флэш-памяти SPI. Это 0x1000 (4096) байт для поддерживаемых микросхем флэш-памяти.*

**Чтение встроенного MAC-адреса: read\_mac**

esptool.py read\_mac

**Прочитать идентификатор флэш-памяти SPI: flash\_id**

esptool.py flash\_id

Пример вывода:

Производитель: e0

Устройство: 4016

Обнаруженный размер флэш-памяти: 4 МБ

**Преобразование ELF в двоичный: elf2image**

Команда elf2image преобразует файл ELF (из вывода компилятора / компоновщика) в двоичные исполняемые изображения, которые можно прошить, а затем загрузить в:

esptool.py --chip esp8266 elf2image my\_app.elf

Эта команда не требует последовательного подключения.

elf2image также принимает аргументы режимов Flash --flash\_freq и --flash\_mode, которые можно использовать для установки значений по умолчанию в заголовке изображения. Это важно при генерации любого изображения, которое будет загружаться непосредственно чипом. Эти значения также можно перезаписать с помощью команды write\_flash, подробности см. В команде write\_flash.

По умолчанию elf2image использует разделы в файле ELF для генерации каждого сегмента в двоичном исполняемом файле. Чтобы использовать вместо этого сегменты (PHDR), передайте параметр --use\_segments.

elf2image для ESP8266

По умолчанию команда выводит два двоичных файла: my\_app.elf-0x00000.bin и my\_app.elf-0x40000.bin. Вы можете изменить префикс имени файла прошивки, используя параметр --output / -o.

elf2image может также создать файл образа "версии 2", подходящий для использования с заглушкой загрузчика программного обеспечения, такой как rboot или программа загрузчика Espressif. Вы не можете прошить образ "версии 2", не прошив также подходящий загрузчик.

esptool.py --chip esp8266 elf2image --version = 2 -o my\_app-ota.bin my\_app.elf

elf2image для ESP32

Для ESP32 elf2image создает единственный выходной двоичный «файл изображения». По умолчанию он имеет то же имя, что и файл .elf, с расширением .bin. то есть:

esptool.py --chip esp32 elf2image my\_esp32\_app.elf

В приведенном выше примере файл выходного изображения будет называться my\_esp32\_app.bin.

**Детальный вывод .bin образа: image\_info**

Команда image\_info выводит некоторую информацию (адреса загрузки, размеры и т. Д.) о .bin файле, созданном elf2image.

esptool.py --chip esp32 image\_info my\_esp32\_app.bin

*Обратите внимание, что при чтении образов ESP32* ***требуется --chip esp32****. В противном случае по умолчанию используется --chip esp8266, и изображение будет интерпретировано как недопустимое изображение ESP8266.*

**Расширенные команды**

Следующие команды используются реже или представляют интерес только для опытных пользователей. Они задокументированы в вики:

verify\_flash

dump\_mem

load\_ram

read\_mem & write\_mem

read\_flash\_status

write\_flash\_status

chip\_id

make\_image

run

**Дополнительные инструменты ESP32**

Следующие инструменты для ESP32 в комплекте с esptool.py задокументированы в вики:

espefuse.py - для чтения / записи региона efuse ESP32

espsecure.py - для работы с функциями безопасности ESP32

**Вход в загрузчик**

И ESP8266, и ESP32 должны быть сброшены определенным образом, чтобы запустить последовательный загрузчик.

На некоторых платах разработки (включая NodeMCU, WeMOS, HUZZAH Feather, Core Board, ESP32-WROVER-KIT) esptool.py может автоматически запускать сброс последовательного загрузчика - в этом случае вам не нужно читать этот раздел.

Для всех остальных, чтобы войти в последовательный загрузчик, должны произойти три вещи - сброс, правильная установка требуемых контактов и низкий уровень GPIO0.

**Режим загрузки**

И ESP8266, и ESP32 выбирают режим загрузки при каждом сбросе. Событие сброса может произойти одним из нескольких способов:

На микросхему подается питание.

На выводе nRESET был низкий уровень, и он подтянут вверх (только на ESP8266).

На выводе CH\_PD / EN ("разрешение") был низкий уровень, и он подтянут к высокому уровню.

На ESP8266 контакты nRESET и CH\_PD должны быть подтянуты высоко, чтобы микросхема начала работать.

Дополнительные сведения о выборе режима загрузки см. На следующих страницах Wiki:

Выбор режима загрузки ESP8266

Выбор режима загрузки ESP32

**Режимы прошивки**

write\_flash и некоторые другие команды принимают аргументы командной строки для установки режима флэш-памяти загрузчика, размера флэш-памяти и тактовой частоты флэш-памяти. Для правильной работы микросхеме необходимы правильные настройки режима, частоты и размера, хотя есть некоторая гибкость. Заголовок в начале загрузочного образа содержит эти значения.

Чтобы переопределить эти значения, параметры --flash\_mode, --flash\_size и / или --flash\_freq должны быть указаны после write\_flash в командной строке, например:

esptool.py --port COM5 write\_flash --flash\_mode dio --flash\_size 4MB 0x0 bootloader.bin

Эти параметры используются только при прошивке загрузочного образа на ESP8266 со смещением 0x0 или ESP32 со смещением 0x1000. Это адреса, используемые загрузчиком ПЗУ для загрузки из флеш-памяти. При перепрошивке на всех остальных смещениях эти аргументы не используются.

**Режим прошивки (--flash\_mode, -fm)**

Они устанавливают режимы ввода / вывода Quad Flash или Dual Flash I / O. Допустимые значения: keep, qio, qout, dio, dout. По умолчанию используется keep, при котором сохраняется любое значение, уже имеющееся в файле изображения. Этот параметр также можно указать с помощью переменной среды ESPTOOL\_FM.

Большинство плат используют режим qio. Некоторые модули ESP8266, в том числе модули ESP-12E на некоторых (не на всех) платах NodeMCU, имеют двойной ввод-вывод, и микропрограмма загружается только при установке --flash\_mode dio. Большинство модулей ESP32 также имеют двойной ввод / вывод.

В режиме qio два дополнительных GPIO (9 и 10) используются для флэш-связи со SPI. Если режим прошивки установлен на dio, эти контакты доступны для других целей.

Полное описание этих режимов см. На вики-странице SPI Flash Modes.

**Выбор тактовой частоты при прошивке (--flash\_freq, -ff)**

Тактовая частота для взаимодействий со SPI flash. Допустимые значения: 40м, 26м, 20м, 80м (МГц). По умолчанию используется keep, при котором сохраняется любое значение, уже имеющееся в файле изображения. Этот параметр также можно указать с помощью переменной среды ESPTOOL\_FF.

Флэш-чип, подключенный к большинству микросхем, работает с тактовой частотой 40 МГц, но вы можете попробовать более низкие значения, если устройство не загружается. Максимальная тактовая частота флеш-памяти 80 МГц обеспечит наилучшую производительность, но может вызвать сбой, если конструкция флеши или платы не поддерживает такую скорость.

**Размер флеш-памяти (--flash\_size, -fs)**

Размер флэш-памяти SPI в мегабайтах. Допустимые значения зависят от типа микросхемы:

|  |  |
| --- | --- |
| Чип | flash\_size - значения |
| ESP32 | keep, detect, 1 МБ, 2 МБ, 4 МБ, 8 МБ, 16 МБ |
| ESP8266 | keep, detect, 256 КБ, 512 КБ, 1 МБ, 2 МБ, 4 МБ, 2 МБ-c1, 4 МБ-c1, 8 МБ, 16 МБ |

Для ESP8266 доступны некоторые дополнительные размеры и макеты для "слотов прошивки" OTA.

Параметр по умолчанию --flash\_size - keep. Это означает, что если при прошивке загрузчика не передается аргумент --flash\_size, значение в заголовке файла .bin загрузчика сохраняется вместо определения фактического размера флеш-памяти и обновления заголовка.

Чтобы включить автоматическое определение размера флэш-памяти на основе идентификатора флэш-памяти SPI, добавьте аргумент esptool.py [...] write\_flash [...] -fs detect. Если обнаружение не удается, выводится предупреждение и используется значение по умолчанию 4 МБ (4 мегабайта).

Если размер флэш-памяти не обнаружен, вы можете узнать размер флэш-памяти с помощью команды flash\_id, а затем найти идентификатор в выходных данных (см. Чтение идентификатора флэш-памяти SPI). В качестве альтернативы, прочтите маркировку шелкографии на флеш-чипе и найдите ее техническое описание.

Параметр flash\_size по умолчанию также можно переопределить с помощью переменной среды ESPTOOL\_FS.

**ESP8266 и размер флэш-памяти**

ESP8266 SDK хранит конфигурацию WiFi в «конце» флеш-памяти и находит конец, используя этот размер. Однако нет проблемы в указании меньшего размера флэш-памяти, чем вам надо на самом деле, если вам не нужно записывать изображение больше этого размера.

Модули ESP-12, ESP-12E и ESP-12F (и платы, которые их используют, такие как NodeMCU, HUZZAH и т. Д.) Обычно имеют не менее 4 мегабайт / 4 МБ (иногда обозначаемых 32 мегабитами) флэш-памяти.

При использовании OTA доступны некоторые дополнительные размеры и макеты для «слотов прошивки» OTA. Если вы не используете обновления OTA, вы можете игнорировать эти дополнительные размеры:

|  |  |  |  |
| --- | --- | --- | --- |
| flash\_size arg | Number of OTA slots | OTA Slot Size | Non-OTA Space |
| 256KB | 1 (no OTA) | 256KB | N/A |
| 512KB | 1 (no OTA) | 512KB | N/A |
| 1MB | 2 | 512KB | 0KB |
| 2MB | 2 | 512KB | 1024KB |
| 4MB | 2 | 512KB | 3072KB |
| 2MB-c1 | 2 | 1024KB | 0KB |
| 4MB-c1 | 2 | 1024KB | 2048KB |
| 8MB [^] | 2 | 1024KB | 6144KB |
| 16MB [^] | 2 | 1024KB | 14336KB |

*[^] Поддержка флэш-памяти размером 8 МБ и 16 МБ отсутствует во всех SDK ESP8266. Если ваш SDK не поддерживает эти размеры флэш-памяти, используйте --flash\_size 4MB.*

**ESP32 и размер** **флэш-памяти**

ESP-IDF записывает таблицу разделов во флэш-память со смещением 0x8000. Все разделы в этой таблице ***должны соответствовать настроенному размеру флэш-памяти, иначе ESP32 не будет работать правильно***.

**Слияние двоичных файлов**

Команда merge\_bin объединит несколько двоичных файлов (любого типа) в один файл, который позже можно будет записать на устройство. Любые промежутки между входными файлами заполняются байтами 0xFF (так же, как незаписанное содержимое флэш-памяти).

Например:

esptool.py --chip esp32 merge\_bin -o merged-flash.bin --flash\_mode dio --flash\_size 4MB 0x1000 bootloader.bin 0x8000 partition-table.bin 0x10000 app.bin

Будет создан файл merged-flash.bin с содержимым остальных 3 файлов. Позднее этот файл можно записать во флэш-память с помощью esptool.py write\_flash 0x0 merged-flash.bin.

***Примечание.*** Поскольку промежутки между входными файлами заполняются байтами 0xFF, при записи объединенного двоичного файла все флеш-секторы между отдельными файлами ***будут стерты***. Чтобы этого избежать, ***записывайте файлы по отдельности***.

**Опции**

Команда merge\_bin поддерживает те же параметры --flash\_mode, --flash\_size и --flash\_speed, что и команда write\_flash, для переопределения заголовка флэш-памяти загрузчика (подробности см. Выше). Эти параметры применяются к содержимому выходного файла так же, как и при записи во флэш-память. Обязательно передайте параметр --chip, если используете эти параметры, поскольку поддерживаемые значения и смещение загрузчика зависят от чипа.

Параметр --target-offset 0xNNN создаст объединенный двоичный файл, который должен быть прошит с указанным смещением, а не со смещением 0x0.

Параметр --fill-flash-size SIZE дополняет объединенный двоичный файл байтами 0xFF до указанного размера полной флэш-памяти, например --fill-flash-size 4MB создаст двоичный файл размером 4 МБ.