idodoyo / **SPIFlash** Public

forked from Marzogh/SPIMemory

Code | Pull requests | Actions | Projects | Wiki | Security | Insights

5 Branches | 16 Tags | Go to file | Go to file | About

This branch is 408 commits behind `Marzogh/SPIMemory:master` .

Marzogh Merge pull request Marzogh#33 from Marzo...

a804cdd · 10 лет назад

examples	merge issues fixed	10 лет назад
extras	Pre-release platform ...	10 лет назад
.gitignore	fixed gitignore	10 лет назад
.travis.yml	Removing Zero from ...	10 лет назад
Изменения.р...	исправлены пробле...	10 лет назад
ЛИЦЕНЗИЯ	v1.3.0 финальная ве...	11 лет назад
README.md	обновил файл read...	10 лет назад
SPIFlash.cpp	исправлены пробле...	10 лет назад
SPIFlash.h	исправлены пробле...	10 лет назад
определяет.h	Готово к выпуску ве...	10 лет назад
keywords.txt	Исправлен getAddr...	11 лет назад
библиотека.с...	Исправлена ошибк...	10 лет назад

Всплеск build unknown DOI

Библиотека Arduino для микросхем флэш-памяти Winbond

Загрузите последнюю стабильную версию (v2.4.0) с здесь. Пожалуйста, сообщайте о любых ошибках в комментариях.

SPI library for Winbond Flash Memory modules

Readme

GPL-3.0 license

Activity

0 stars

1 watching

0 forks

Report repository

Релизы

16 теги

Пакетов

Пакеты не опубликованы

Языки

C++ 94,6% C 5,4%

Эта библиотека Arduino предназначена для работы с микросхемами флэш-памяти Winbond. В текущей версии она поддерживает определение микросхемы флэш-памяти и ее различных функций, автоматическое распределение адресов и управление ими, чтение и запись байтов/символов/целых чисел/длинных чисел/чисел с плавающей запятой/строк из различных ячеек и в них, чтение и запись страниц байтов, непрерывное чтение/запись данных из/в массивы байтов/символов, стирание секторов, блоков и всей микросхемы, а также выключение для работы в режиме пониженного энергопотребления.

- Подробную информацию о микросхемах Winbond Flash, совместимых с этой библиотекой, можно найти в таблице Excel в папке «Дополнительно».

####Совместимость #### Поддерживаемые среды разработки Arduino

- IDE v1.5.x
- IDE v1.6.0-v1.6.5
- IDE v1.6.9-v1.6.11

#####Доски

######Полностью поддерживается

- Arduino Uno
- Arduino Leonardo
- Arduino Micro
- Arduino Fio
- Arduino Mega
- Arduino Due
- Платы ESP8266 (в Arduino IDE)
- ATTiny85

####Установка

#####Вариант 1

- Нажмите кнопку «Скачать zip» справа.
- Распакуйте архив и переименуйте полученную папку в «SPIFlash».
- Переместите папку в папку с библиотеками (~/sketches/libraries)

#####Вариант 2

- Откройте Arduino IDE.
- Перейдите в Sketch > Включить библиотеку > Управление библиотеками.
- Найдите SPIFlash.
- Установите последнюю версию.

Использование

Библиотека вызывается путем объявления конструктора `SPIFLASH flash(csPin)`, где 'flash' можно заменить на пользовательский конструктор по выбору, а 'csPin' — это контакт выбора микросхемы для флэш-модуля. Не забудьте подключить `#include<SPI.H>` при подключении `#include<SPIFlash.h>`. Также не забудьте подключить `flash.begin()` в `void setup()`. Это позволит библиотеке определить тип установленной микросхемы флэш-памяти и загрузить нужные параметры.

Заметки о переполнении буфера и проверке ошибок

- По умолчанию в библиотеке включено переполнение адреса — то есть если последний адрес, считанный/записанный в/из памяти в любой функции, равен 0xFFFFF, то следующий адрес, считанный/записанный в/из памяти, будет равен 0x00000. Это можно отключить, установив для необязательного последнего аргумента 'overflow' в конструкторе значение false. Например, вызовите конструктор `SPIFlash(csPin, false)` вместо `SPIFlash(csPin)`.
- По умолчанию во всех функциях записи включена проверка ошибок, то есть каждый байт, записываемый во флэш-память, будет сверяться с данными, хранящимися на Arduino. Пользователи, которым требуется более высокая скорость записи, могут отключить эту функцию, установив для необязательного последнего аргумента 'errorCheck' в любой функции записи значение false. Например, вызовите функцию `writePage(page_number, *data_buffer, false)` вместо `writePage(page_number, *data_buffer)`.

Библиотека предоставляет следующие функции:

Основные команды ###### begin() Должна вызываться в начале функции setup(). Эта функция определяет тип используемого чипа и соответствующим образом настраивает параметры. ###### error() Возвращает (8-битный) код ошибки, сгенерированный функцией, вызванной непосредственно перед этой функцией. См. раздел [Коды ошибок и отчеты об ошибках] (https://github.com/Marzogh/SPIFlash/wiki/Error-codes-&-Error-reporting) в вики-разделе для получения дополнительной информации. ###### getMANID() Возвращает идентификатор производителя в виде 16-битного значения. ###### getJEDECID() Возвращает идентификатор JEDEC в виде 32-битного значения. ###### getCapacity() Возвращает объем памяти в байтах в виде 32-битного значения. ###### getmaxPage() Возвращает максимальное количество страниц во флэш-памяти в виде 32-битного значения. ###### getChipName() Возвращает идентификационный номер чипа, на основе которого можно сгенерировать его название. (См. функцию getID() в примерах/Diagnostics/Diagnostics.ino) ###### getAddress() Получает следующий доступный для использования адрес. Есть два варианта: * Принимает в качестве аргумента размер данных и возвращает 32-битный адрес * Принимает три переменные: размер данных и две другие переменные для возврата значения номера страницы и смещения.

Все адреса в скетче должны быть получены с помощью этой функции или не получены вовсе.

sizeofStr()

Используйте эту функцию, чтобы узнать размер строки и использовать его в качестве аргумента в getAddress(size). Если использовать size = sizeof(String), функция getAddress(size) не сработает.

size = sizeof(переменная) можно использовать для всех типов данных, кроме строковых объектов.

Все команды чтения/записи могут использовать 32-битную адресную переменную вместо 16-битного номера страницы и 8-битной переменной смещения.

Команды чтения Все команды чтения принимают последний логический аргумент 'fastRead'. По умолчанию этот аргумент имеет значение FALSE, но если задать для него значение TRUE, будет выполнена инструкция Fast Read, позволяющая считывать данные с максимальной частотой памяти.
Все команды чтения могут принимать 32-битную адресную переменную вместо 16-битного номера страницы и 8-битной переменной смещения.

readAnything(номер_страницы, смещение, значение)

Считывает *любой тип переменной/структуры* (значение любого размера) из определенного места на странице. В качестве аргументов принимает номер страницы (от 0 до maxPage), смещение данных на странице (от 0 до 255) и переменную/структуру, в которую нужно записать данные.

readByte(номер_страницы, смещение)

Считывает *байт* (8-битное значение без знака) из определенного места на странице. В качестве аргументов принимает номер страницы (0–maxPage) и смещение байта на странице (0–255).

readChar(номер_страницы, смещение)

Считывает *символ* (8-битное значение со знаком) из определенного места на странице. В качестве аргументов принимает номер страницы (0–maxPage) и смещение символа на странице (0–255).

readWord(номер_страницы, смещение)

Считывает *слово* (16-битное значение без знака) из определенного места на странице. В качестве аргументов принимает номер страницы (0–maxPage) и смещение слова на странице (0–255).

readShort(номер_страницы, смещение)

Считывает *short* (16-битное значение со знаком) из определенного места на странице. В качестве аргументов принимает номер страницы (0–maxPage) и смещение короткого значения на странице (0–255).

readULong(номер_страницы, смещение)

Считывает *беззнаковое длинное* (беззнаковое 32-битное значение) из определенного места на странице. В качестве аргументов принимает номер страницы (0–maxPage) и смещение беззнакового длинного значения на странице (0–255).

readLong(номер_страницы, смещение)

Считывает *long* (32-битное значение со знаком) из определенного места на странице. В качестве аргументов принимает номер страницы (0–maxPage) и смещение long на странице (0–255).

readFloat(номер_страницы, смещение)

Считывает *число с плавающей запятой* (десятичное значение) из определенного места на странице. В качестве аргументов принимает номер страницы (от 0 до maxPage) и смещение числа с плавающей запятой на странице (от 0 до 255).

readStr(номер_страницы, смещение, outputStr)

Считывает *строку* (объект String) из определенного места на странице в переменную outputStr. Принимает в качестве аргументов номер страницы (от 0 до maxPage), смещение строки на странице (от 0 до 255) и строку.

readPage(номер_страницы, *буфер_данных)

Считывает данные размером со страницу в буфер данных для дальнейшего использования. `uint8_t data_buffer[256];` Буфер данных **должен** представлять собой массив из 256 байт.

readAnything(номер_страницы, смещение, значение)

Считывает *любой тип переменной/структуры* (значение любого размера) из определенного места на странице. В качестве аргументов принимает номер страницы (от 0 до maxPage), смещение данных на странице (от 0 до 255) и переменную/структуру, в которую нужно записать данные.

Команды записи Все команды записи принимают последний логический аргумент 'errorCheck'. По умолчанию этот аргумент имеет значение TRUE, но если установить для него значение FALSE, скорость записи увеличится более чем в два раза. Однако за это придется расплачиваться проверкой на наличие ошибок при записи. Используйте с осторожностью.
Все команды записи могут принимать 32-битную адресную переменную вместо 16-битного номера страницы и 8-битной переменной смещения

writeByte(страница, смещение, данные)

Записывает байт данных в определенное место на странице. В качестве аргументов принимает номер страницы (0–maxPage), смещение байта данных на странице (0–255) и один байт данных.

writeChar(номер_страницы, смещение, данные)

Записывает *символ* (8-битное значение со знаком) в определенное место на странице. В качестве аргументов принимает номер страницы (от 0 до maxPage), смещение символа на странице (от 0 до 255) и один символ данных.

writeWord(номер_страницы, смещение, данные)

Записывает *слово* (16-битное значение без знака) в определенное место на странице. В качестве аргументов принимает номер страницы (от 0 до maxPage), смещение слова на странице (от 0 до 255) и одно слово данных.

writeShort(номер_страницы, смещение, данные)

Записывает *short* (16-битное значение со знаком) в определенное место на странице. В качестве аргументов принимает номер страницы (от 0 до maxPage), смещение короткого значения на странице (от 0 до 255) и одно короткое значение данных.

writeULong(номер_страницы, смещение, данные)

Записывает *unsigned long* (32-битное значение без знака) в определенное место на странице. В качестве аргументов принимает номер страницы (от 0 до maxPage), смещение unsigned long внутри страницы (от 0 до 255) и одно значение unsigned long с данными.

writeLong(номер_страницы, смещение, данные)

Записывает *long* (32-битное значение со знаком) в определенное место на странице. В качестве аргументов принимает номер страницы (от 0 до maxPage), смещение long в пределах страницы (от 0 до 255) и один long с данными.

writeFloat(номер_страницы, смещение, данные)

Записывает *число с плавающей запятой* (десятичное значение) в определенное место на странице. В качестве аргументов принимает номер страницы (от 0 до maxPage), смещение числа с плавающей запятой на странице (от 0 до 255) и одно число с плавающей запятой с данными.

writeStr(номер_страницы, смещение, данные)

странице. Принимает в качестве аргументов номер страницы (от 0 до maxPage), смещение строки на странице (от 0 до 255) и строку.

writePage(номер_страницы, *буфер_данных)

Записывает данные размером со страницу в буфер данных для дальнейшего использования. `uint8_t data_buffer[256];` Буфер данных **должен** представлять собой массив из 256 байт.

writeAnything(номер_страницы, смещение, значение)

Записывает *любой тип переменной/структуры* (значение любого размера) из определенного места на странице. В качестве аргументов принимает номер страницы (от 0 до maxPage), смещение данных на странице (от 0 до 255) и переменную/структуру, из которой нужно записать данные.

Непрерывные команды чтения/записи Все команды записи принимают логический последний аргумент 'errorCheck'. По умолчанию этот аргумент имеет значение TRUE, но если установить значение FALSE, скорость записи увеличится более чем в два раза. Однако это происходит за счёт проверки на наличие ошибок записи. Используйте с осторожностью.

readByteArray(номер_страницы, смещение, *буфер_данных, размер_буфера)

Считывает массив байтов, начиная с определенного места на странице. Принимает в качестве аргументов номер страницы (0–maxPage), смещение байта данных на странице (0–255), data_buffer — то есть массив байтов, который нужно считать из флэш-памяти, — и размер массива. `uint8_t data_buffer[n];` Буфер данных **должен** представлять собой массив из n **байтов**. Значение n определяется объемом оперативной памяти на плате Arduino.

writeByteArray(номер_страницы, смещение, *буфер_данных, размер_буфера)

Записывает массив байтов, начиная с определенного места на странице. Принимает в качестве аргументов номер страницы (0–maxPage), смещение байта данных на странице (0–255), data_buffer — то есть массив байтов, который нужно записать во флэш-память, — и размер массива. `uint8_t data_buffer[n];` Буфер данных **должен** быть массивом из 'n' **байтов**. Количество байтов — 'n' — определяется объемом оперативной памяти на плате Arduino.

readCharArray(номер_страницы, смещение, *буфер_данных, размер_буфера)

Считывает массив символов, начиная с определенного места на странице. Принимает в качестве аргументов номер страницы (0–maxPage), смещение байта данных на странице (0–255), data_buffer — то есть массив символов, который нужно считать из флэш-памяти, — и размер массива. `char data_buffer[n];` Буфер данных **должен** представлять собой массив из n **символов**. Значение n определяется объемом оперативной памяти на плате Arduino.

writeCharArray(номер_страницы, смещение, *буфер_данных, размер_буфера)

Записывает массив символов, начиная с определенного места на странице. Принимает в качестве аргументов номер страницы (0–maxPage), смещение байта данных на странице (0–255), data_buffer — то есть массив символов, который нужно записать во флэш-память, — и размер массива. `char data_buffer[n];` Буфер данных **должен** представлять собой массив из 'n' **символов**. Количество символов — 'n' — определяется объемом оперативной памяти на плате Arduino.

Команды стирания Все команды стирания могут использовать 32-битную адресную переменную вместо 16-битного номера страницы и 8-битных переменных смещения ###### eraseSector(page_number, offset) Стирает один сектор размером 4 КБ - 16 страниц, содержащий страницу, подлежащую стиранию. Сектора пронумерованы от 0 до 255 и содержат по 16 страниц в каждом. Страница 0-15 -> Сектор 0; Страница 16-31 -> Сектор 1;..... Страница 4080-4095 -> Сектор 255 и так далее... ###### eraseBlock32K(page_number, смещение) Удаляет один блок размером 32 КБ - 128 страниц, содержащий удаляемую страницу. Блоки пронумерованы от 0 до 31 и содержат по 128 страниц в каждом. Страница 0-127 --> Блок 0; Страница 128-255 --> Блок 1;..... Страница 3968-4095 --> Блок 31 и так далее... ###### eraseBlock64K(page_number, смещение) Удаляет один блок размером 64 КБ - 256 страниц, содержащий удаляемую страницу. Блоки пронумерованы от 0 до 15 и содержат по 256 страниц в каждом. // Страница 0-255 --> Блок 0; Страница 256-511 --> Блок 1;..... Страница 3840-4095 --> Блок 15 и так далее... ###### eraseChip() Удаляет весь чип. Используйте с осторожностью.

Команды приостановки / возобновления ####### suspendProg() Приостанавливает текущую программу стирания блоков / секторов / страниц. Не приостанавливает работу chipErase(). Программа страницы, регистр состояния записи, инструкции по стиранию запрещены. Приостановка стирания разрешена только во время стирания блока / сектора. Приостановка программы разрешена только во время выполнения программы Page / Quad Page ###### resumeProg() Возобновляет ранее приостановленную программу стирания блоков / секторов / Страниц.

Команды управления питанием ###### powerDown() Переводит устройство в режим пониженного энергопотребления. Полезно при работе от аккумулятора. Типичное потребление тока в режиме пониженного энергопотребления составляет 1 мА, максимальное — 5 мА. (Техническое описание, раздел 7.4). В режиме powerDown() микросхема реагирует только на команду powerUp() ###### powerUp() // Выводит микросхему из режима пониженного энергопотребления.

Объяснение кодов ошибок - 0x00 SUCCESS Операция выполнена успешно. - 0x01 CALLBEGIN Пожалуйста, убедитесь, что в функции setup() был вызван метод .begin(). - 0x02 UNKNOWNCHIP Неизвестный производитель чипа. - 0x03 UNKNOWNCAP Неизвестная ёмкость чипа. - 0x04 CHIPBUSY Чип занят. - 0x05 OUTOFBOUNDS Адрес выходит за границы. Пожалуйста, проверьте, был ли вызван метод .begin() в функции setup(). - 0x06 CANTENWRITE Не удалось выполнить _writeEnable. Проверьте проводку/микросхему. - 0x07 OUTOFMEM Номер страницы

превышает максимальное значение. - 0x08 OUTOFPAGE Смещение выходит за пределы страницы.

Устаревшие функции

Следующие функции объявлены устаревшими для обеспечения совместимости с другими чипами AVR. `printPage();` `printAllPages()` `readSerialStr()`

Их можно использовать, раскомментировав в файле SPIFlash.cpp. Однако имейте в виду, что этот блок кода был протестирован только в Arduino IDE (версия 1.6.5) и только на 8-битных платах Arduino на базе ATMega и больше не поддерживается.

printPage(номер_страницы, тип_вывода)

Считывает данные размером со страницу в буфер данных для дальнейшего использования и по умолчанию выводит их в последовательный поток со скоростью 115200 бод. (Скорость передачи данных можно изменить, вызвав `Serial.begin()` с другой скоростью передачи данных в `void setup()`) При значении outputType, равном 1, данные выводятся в шестнадцатеричном формате, а при значении outputType, равном 2, — в десятичном формате в формате CSV через последовательный порт.

printAllPages()

Считывает все страницы с флэш-чипа и выводит их в последовательный поток. Эта функция полезна при извлечении данных с флэш-чипа на компьютер в виде текстового файла.

Как получить данные с флэш-памяти через последовательный порт

(Работает только в операционных системах на базе Unix)

- Убедитесь, что вы добавили код, позволяющий выгружать в последовательный порт всю флэш-память/содержимое определенной страницы (подробнее о том, как это сделать, см. в коде *TestFlash.h*)
- Подключите плату Arduino к компьютеру.
- Откройте Arduino IDE, последовательный монитор IDE и терминал OSX POSIX.
- Введите в окне терминала следующую команду. `% tail -f /dev/tty.usbmodem1411 > FlashDump.txt` Не забудьте заменить */dev/tty.usbmodem1411* на порт, к которому подключена ваша