Контроллер дисплея ST7565

13 июля 2012 г.

На этой странице представлена теория использования графического ЖК-дисплея на базе контроллера ST7565. Они широко доступны с популярными размерами 132x32 и 128x64, стоимость номера ниже £ 10.

(Изображение © Electronic Assembly, lcd-module.de)

Учебное пособие идет по тому же пути, который я выбрал при разработке простой библиотеки для экрана, который я купил. Я поставил перед собой задачу сделать это с нуля, а не использовать код из Интернета.

Хотя в этом посте в качестве примера используется экран 128x64, он в равной степени применим к разным размерам.

Плюсы:

- Простой интерфейс с использованием меньшего количества контактов, чем у других графических ЖК-дисплеев
- Бортовой усилитель напряжения для жидкокристаллического дисплея
- Контролируемый программно контраст и вращение
- Работает на более высоких тактовых частотах
- Хороший выбор цветов подсветки

Минусы:

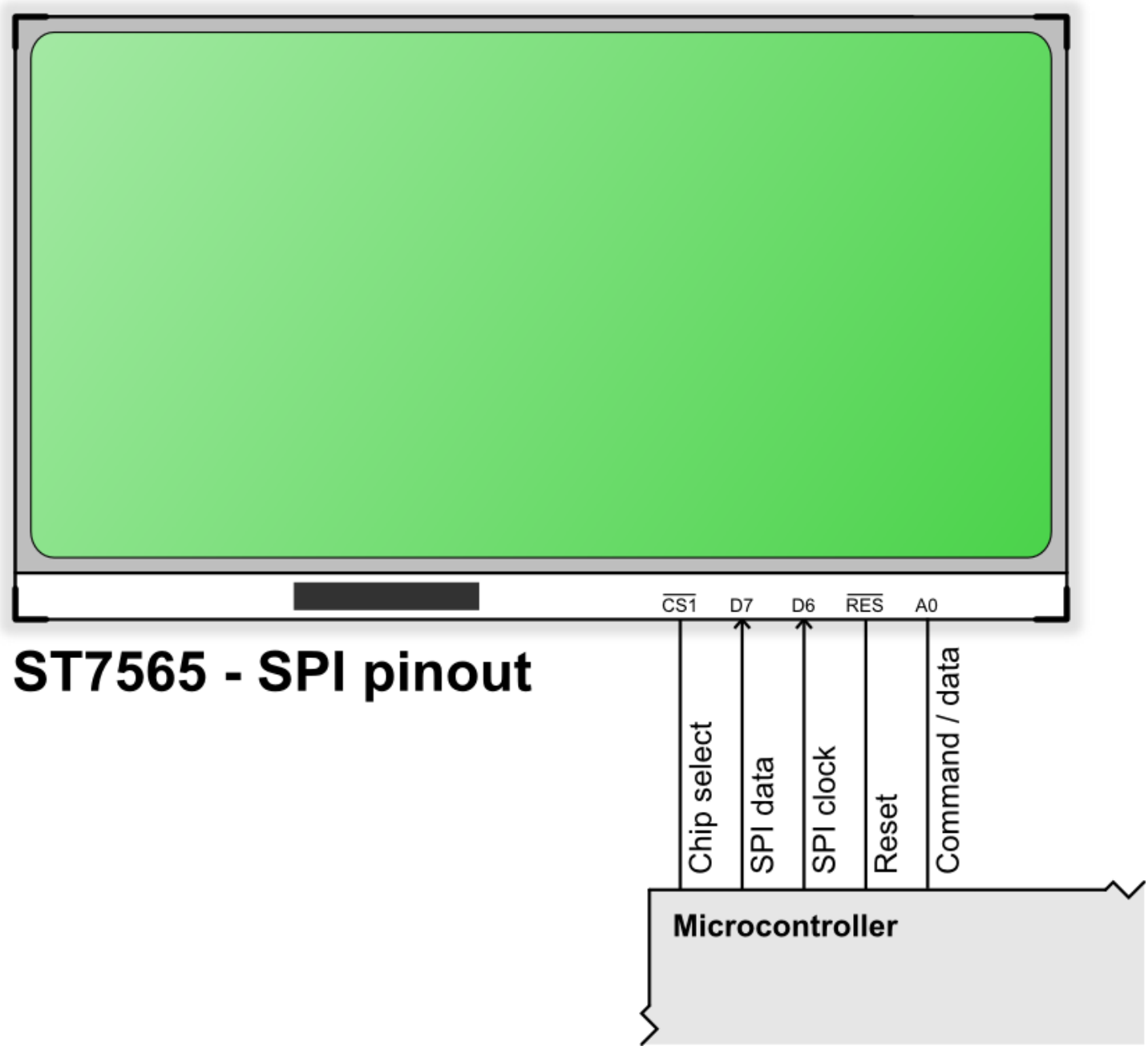
- Только логика 3.3v (больше не проблема)
- В режиме SPI в локальной памяти требуется большой буфер

Начиная

Самая важная вещь, необходимая при разработке библиотеки - это хорошая таблица данных. Информация, предоставляемая производителями экранов, часто не хватает деталей и качества. Таблицу, которую я использовал, можно найти здесь .

Основные операции (аппаратные)

Основная распиновка для ST7565 показана ниже. Для работы с микроконтроллером в режиме SPI требуется 5 контактов. Все эти выводы - это выходы микроконтроллера и входы на экран. Поскольку невозможно читать с экрана, требования к входным данным отсутствуют.



Необходимо подключить RESET к микроконтроллеру, поскольку процедура инициализации включает переключение этого контакта. Этот вывод не должен быть привязан к верхнему или нижнему значению, иначе может быть невозможно правильно сбросить экран.

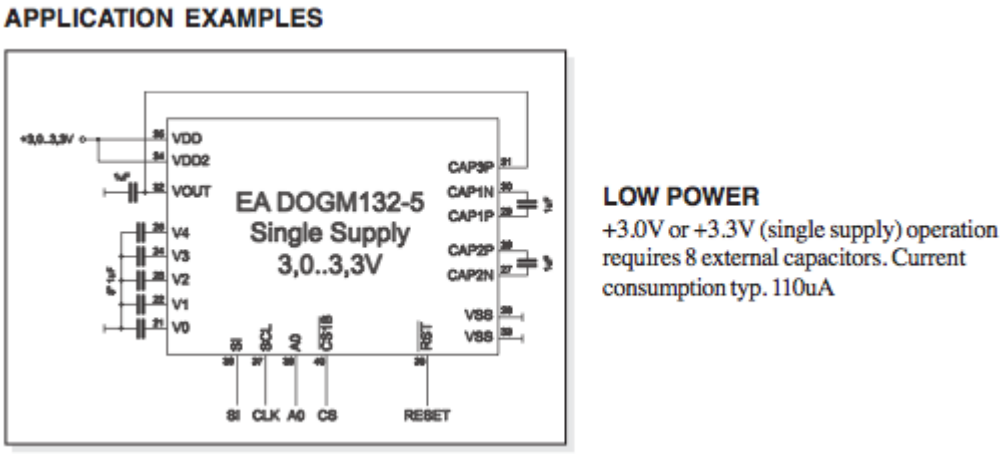
В зависимости от вашего модуля GLCD может потребоваться подключить несколько конденсаторов, это обсуждается в следующем разделе.

Повышение напряжения и регулирование

ST7565 имеет встроенную схему повышения напряжения (подкачки заряда), которая может обеспечивать высокое напряжение, необходимое для жидкокристаллического дисплея. Если в цепи есть только один источник питания, например, 3,3 В, то зарядный насос является самым простым способом питания экрана. Если цепь содержит источник напряжения между 9 В и 12 В, то можно уменьшить количество необходимых внешних компонентов, используя это напрямую.

Внутренний усилитель может обеспечить от 2x до 6x входного напряжения в зависимости от того, как конденсаторы подключены между клеммами CAPxx. Полная информация представлена на стр. 31-34 таблицы выше. Единственным ограничением является то, что максимальное напряжение после повышения не должно превышать Vout, что составляет 13,5 В.

Многие технические характеристики ЖК-дисплея содержат точные конденсаторы, необходимые для конкретной модели. Для некоторых дисплейных модулей эти конденсаторы включены в монтажную плату и не требуют добавления. Внимательно проверьте таблицу! Например, изображение ниже взято из таблицы данных для недорогой модели, доступной от Mouser, EA DOGM132-5:



(© Электронная Ассамблея, lcd-module.de)

В дополнение к усилителю напряжения ST7565 содержит встроенный регулятор напряжения. Это имеет 8 шагов и регулирует опорное напряжение, используемое для привода жидкокристаллического дисплея. Необходимо найти значение от 0 до 7, которое хорошо работает с конкретным оборудованием, поскольку оно используется с функцией динамического контраста. Обычно «среднее» значение, такое как 3, будет работать с любым экраном.

Основные операции (программное обеспечение)

Режим SPI обеспечивает простой механизм управления экраном, используя несколько контактов. Одиночный бит записывается на вывод данных, и тактовый импульс получает высокий, а затем низкий уровень. Реализация SPI с помощью ST7565 является довольно стандартной, более подробную информацию можно найти на странице Wikipedia SPI Bus .

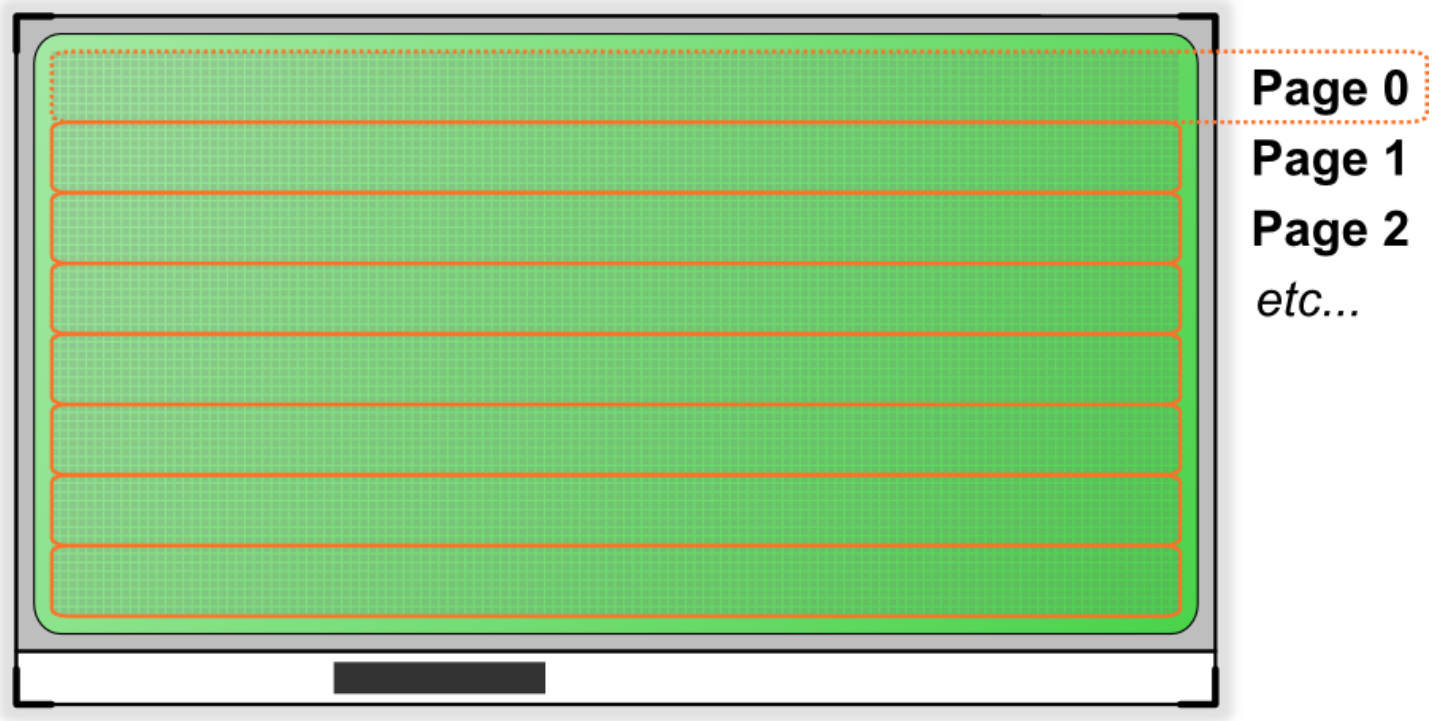
Одним из ограничений использования экрана в режиме SPI является то, что он поддерживает только написание команд или данных. Чтение с экрана невозможно, поэтому необходимо сохранить копию данных текущего экрана в памяти на микроконтроллере. Размер этого программного буфера может быть легко рассчитан, есть 128 столбцов и 64 строки, в общей сложности 8192 бит (1 бит на пиксель для черного и белого). Поэтому для хранения копии экрана требуется всего 1024 байта, что может быть большим для многих небольших микроконтроллеров.

С программным («битым») SPI можно использовать экран на очень высоких тактовых частотах. На PIC 18F я использовал свой экран с внутренним генератором 64 МГц, и он работает абсолютно нормально. Можно рассчитать (или смоделировать / измерить), может ли микроконтроллер превысить минимальный тактовый период ST7565.

Макет экрана

В отличие от дисплеев, использующих KS0108, ST7565 имеет единый контроллер для всего экрана. Нет необходимости переключаться между левым и правым для отправки команд.

Экран 128x64 состоит из 8 страниц, каждая из которых имеет высоту 8 пикселей и ширину 132 дюйма. Это хорошо отображается в программном буфере, необходимом для кэширования содержимого экрана, потому что каждая страница имеет высоту байта.



ST7565 - Pages

Легче всего представить верхнюю левую часть экрана как 0,0, а нижнюю правую как 127,63. Это значительно упрощает некоторые математические вычисления, необходимые в программном обеспечении. Для записи данных в (0,0) необходимо выбрать страницу 0, столбец 0, а затем установить отдельный бит. В отличие от этого место (32,32) находится на странице 5. Самый простой способ записи данных на экран - отправка всей страницы за раз, начиная слева в столбце 0 и продолжая вправо в столбце 127.

Отправка команды на экран

Выполнение одной команды на экране является самым основным строительным блоком кода интерфейса. Необходимые шаги:

- Установите низкий вывод A0, чтобы указать, что данные команды отправляются
- Установите низкий уровень контакта выбора чипа (CS)
- Отправьте каждый бит команды, начиная с *самого старшего* бита и заканчивая самым *младшим*
- Установите высокий уровень контакта выбора микросхемы (CS), чтобы освободить шину

Поэтому команду «display on» (которая 0xAF или 0b10101111) следует отправлять как: 1, 0, 1, 0, 1, 1, 1, 1.

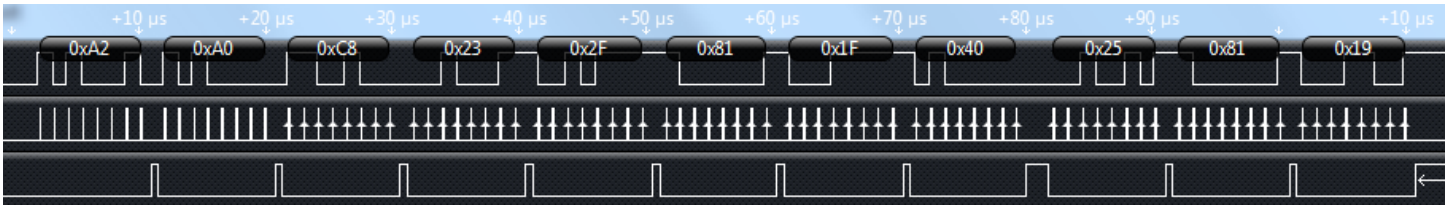
Пока синхронизация SPI не нарушена, нет необходимости вводить какие-либо задержки, так как экран никогда не будет «занят».

Инициализация экрана

Теперь, когда мы можем отправить одну команду, нам нужно выполнить ряд шагов для настройки экрана.

- Строб стека RESET низкий, а затем высокий, чтобы инициировать аппаратный сброс
- Установите рабочий цикл ($^{1}/_{7}$ или $^{1}/_{9}$) в зависимости от физического ЖК-дисплея
- Установите горизонтальную и вертикальную ориентацию в известное состояние
- Настройте внутренний резисторный делитель, который используется регулятором напряжения
- Включите внутренний усилитель напряжения для подачи питания на ЖК-стекло
- Инициализируйте динамический контраст со значением по умолчанию
- Сбросить текущее положение дисплея вверху слева

Команды для них представляют собой один байт, за исключением динамического контраста, который будет объяснен позже. Ниже приведено изображение, полученное из логического анализатора во время выполнения кода инициализации.



В этом следе можно увидеть следующие вещи:

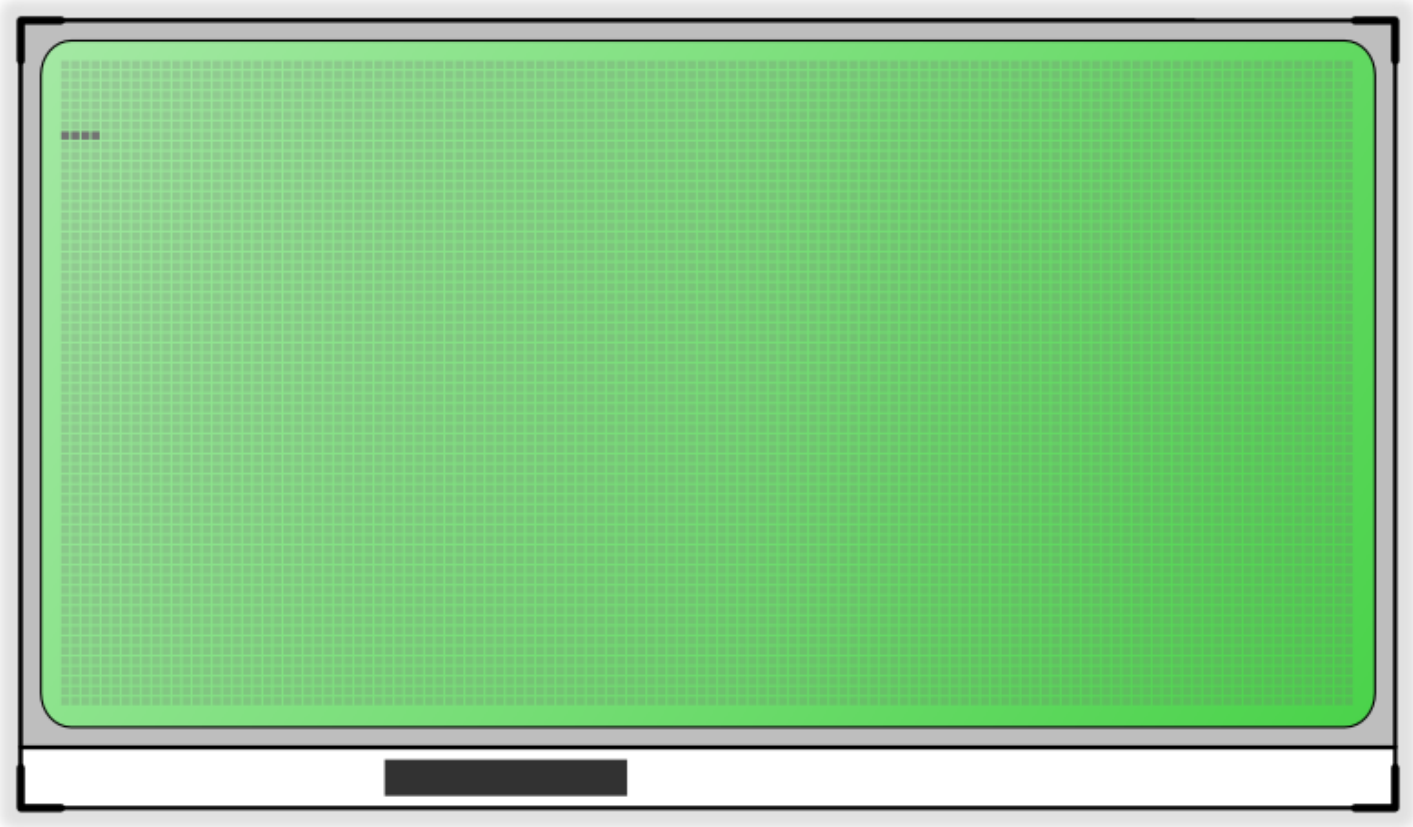
- Команда `0xA2`: установите LCD смещение на 1/9
- Команда `0xA0`: горизонтально «нормальная» (не перевернутая)
- Команда `0xC8`: вертикально «перевернутая» (дополняет команду выше)
- Команда `0x23`: делитель внутреннего резистора установлен на 3 (от 0..7)
- Команда `0x2F`: управление питанием, все внутренние блоки включены
- Команда `0x81`: войти в режим динамического контраста
- Данные для режима динамического контраста, установлены на 31 (от 0..63)
- Команда `0x40`: вернуться в верхний левый угол дисплея

При первом включении питания на экран также важно очистить встроенную экранную память. Поскольку это энергонезависимое ОЗУ, оно запустится в неизвестном состоянии.

Отправка данных на экран

Единственная разница между отправкой команды и данных - это вывод A0. При отправке данных вывод A0 должен быть высоким, чтобы отличать данные шины от команды.

Для начала отправляем команду сброса позиции в верхний левый угол (0,0). После этого быстрый трюк - написать один байт и посмотреть, как он отображается. Важно знать, в каком направлении на экране будут отображаться данные, которые он получает, отображается ли «старший значащий бит» или «наименее значимый бит» вверху. На рисунке ниже показан результат, если байт `0x80`отправляется на экран четыре раза:



Из этого видно, что *самый* значимый бит отображается внизу страницы. Первый бит, который мы посылаем, находится внизу, затем последующие биты работают вверх к верхней части страницы. Это будет важно позже, когда мы попытаемся установить отдельный пиксель.

Отображение базового тестового шаблона

Самый простой способ проверить, работает ли код драйвера, - отправить базовый тестовый шаблон. Быстрая проверка заключается в отправке постоянных байтов `0xFF`на весь экран. Если это работает, то можно попробовать варианты, такие как `0xF0`или `0xAA`.

В конце каждой строки, после каждых 128 байтов, необходимо выбрать следующую страницу и перейти к началу строки. Следующее изображение является основной тестовой картой, которую я использовал, чтобы продемонстрировать, что мой дисплей был правильно подключен.

Запись в определенные пиксели

Написание определенных пикселей открывает целый мир возможностей из линий, кругов, графики 1 бит на пиксель и шрифтов переменной высоты. Поскольку эти графические функции не зависят от конкретного контроллера или экрана, лучше всего разделить их на отдельную библиотеку, которая переносима на любой экран. Это, пожалуй, самая сложная часть при разработке библиотеки для работы с любым экраном.

Чтобы установить определенный пиксель, необходимо разбить процесс на несколько этапов:

- Найти соответствующий байт в программном буфере
- Вычислить бит в этом байте
- Используйте логический XOR (для установки бита) или AND (для сброса бита)

Найти правильный байт просто. Первая позиция в нашем массиве находится в левом верхнем углу экрана. 128-й байт в массиве (`Array[127]`) находится на первой странице в правой части экрана. Следующий бит «переносится» на следующую страницу, на странице 1, столбец 1.

ST7565 - Relation to software buffer

Чтобы вычислить, какой байт нам нужно использовать, нам нужно разделить позицию Y на 8 и умножить результат на 128. К этому мы добавляем позицию X, затем вычитаем 1 (так как массивы C начинаются с 0). Уравнение $ X + ({\ frac {Y} {8} \ times 128}) - 1 $ хорошо работает в C. Чтобы убедиться, что это правильно:

- (1,1) на экране даст $ 1 + (\ frac {1} {8} \ times 128) - 1 = 0 $
- (61,52) на экране даст $ 61 + (\ frac {52} {8} \ times 128) - 1 = 828 $

Чтобы вычислить бит внутри этого байта, мы должны помнить, что 8-й (самый значимый) бит появляется внизу страницы. Таким образом, чтобы записать в Y расположение 8, нам нужно установить 8-й бит байта на первой странице. Чтобы записать в местоположение Y 16, нам нужно записать 8-й бит соответствующего байта на второй странице. Поэтому нам нужен остаток после того, как позиция Y будет разделена на 8, что можно найти с помощью оператора по модулю . Это так же просто, как Y% 8:

- (1,1) означает установку 1% 8 = 1-й бит
- (61,49) означает установку 49% 8 = 1-й бит
- (61,53) означает установку 53% 8 = 5-й бит

Теперь можно установить правильный бит внутри байта, на микроконтроллерах PIC это может быть достигнуто с помощью очень эффективных битовых операторов, таких как bsf или bcf.

Отображение текста

Как и многие другие графические ЖК-дисплеи, ST7565 не имеет встроенного шрифта. Самый быстрый и простой способ отображения текста на этом экране - определить шрифт высотой 8 пикселей. Это отображается непосредственно на одной странице (или «строке») дисплея, и поэтому мы можем использовать очень быстрые инструкции для копирования данных из предопределенного шрифта в правую область экрана.

Однако это ограничивает нас фиксированной высотой и только 8 строками экрана. Теперь, когда мы знаем, как устанавливать отдельные пиксели, есть гораздо более приятные способы отображения текста. Рассмотрим следующее изображение:

Большие шрифты занимают больше места, но могут быть использованы для обеспечения более приятного внешнего вида пользовательского интерфейса для проекта. Зная, что мы можем установить определенный пиксель, легко создавать расширенные графические функции. Однако они не относятся к самому экрану ST7565, поэтому я расскажу о них в следующем посте. Нетерпеливые читатели могут проверить мою графическую библиотеку(документацию по Doxygen можно найти здесь).

Использование расширенных функций

Пока что все функции являются общими для всех стандартных графических ЖК-дисплеев. Однако есть ряд дополнительных приемов, которые могут предложить дисплеи на базе ST7565.

Инвертирование экрана

Одной командой ST7565 может быть дано указание инвертировать дисплей. Это может быть полезно для выделения или облегчения чтения экрана в определенных ситуациях. Команда «отображение нормальной или обратная» является `0b1010011X`, где `X`находится `1`на обратном или `0`для нормального.

На рисунке ниже показана та же демонстрация шрифта, что и выше, но с включенной обратной функцией.

Чернение экрана

ST7565 способен устанавливать все точки на дисплее черным, не влияя на ОЗУ экрана. Хотя трудно представить себе ситуацию, в которой это может быть полезно, команда «показать все точки» представляет собой один байт и может использоваться, если необходимо, чтобы дисплей стал черным.

Динамический контроль контраста

В отличие от других обычных графических ЖК-дисплеев контраст для ST7565 устанавливается внутренним резисторным делителем. Это означает, что он может быть установлен программно и изменен динамически. Чтобы изменить контраст, нам нужно отправить две команды. Первый вводит «режим громкости установлен», а второй устанавливает контраст от 0..63.

Количество полезных шагов может быть меньше 64, на моем тестовом экране видимые шаги от 12 (очень светлые) до 40 (очень темные). Более низкие значения не видны, так как напряжение слишком низкое, более высокие значения делают экран полностью черным, поскольку нет различий между включенными или выключенными пикселями.

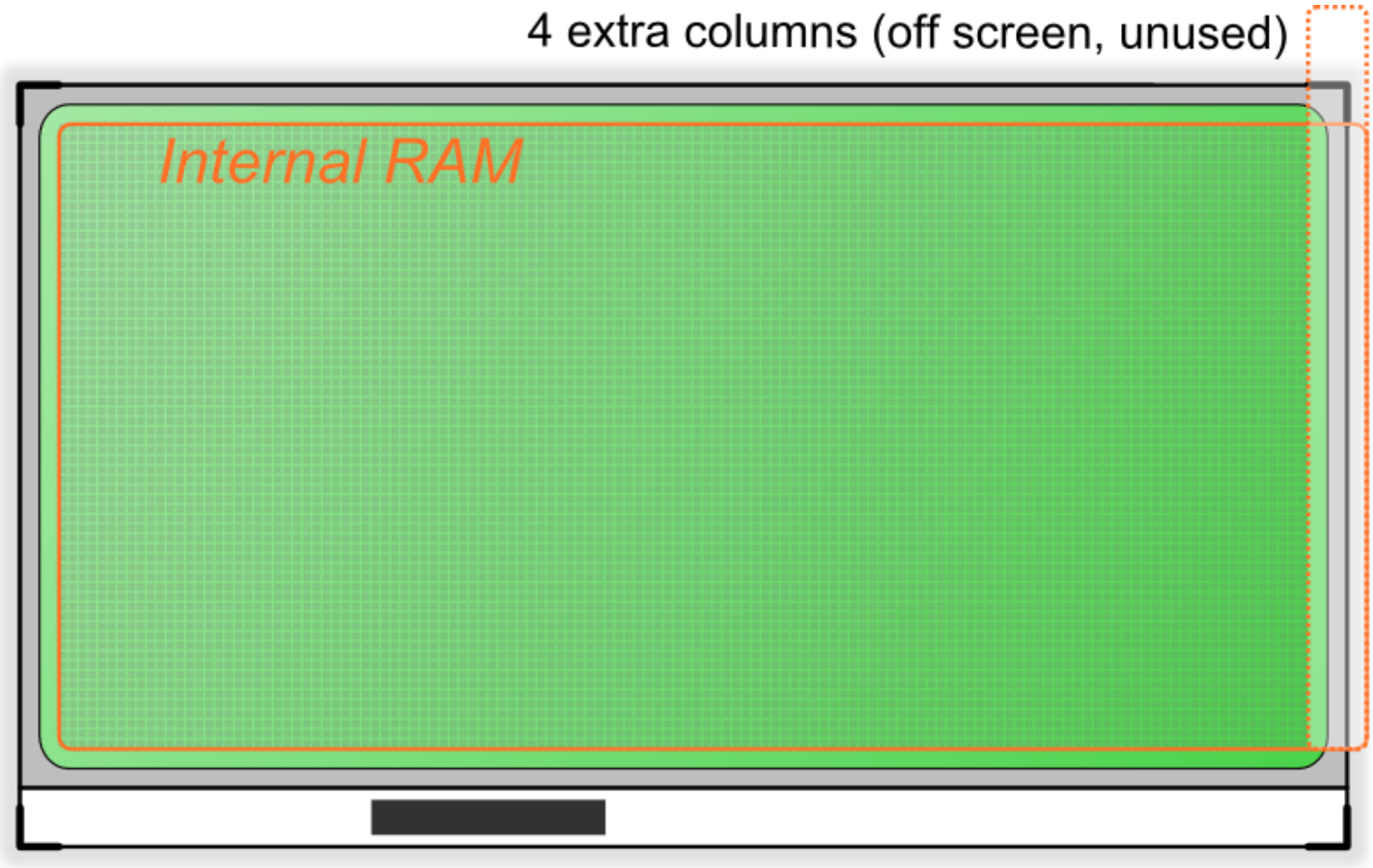
Эта функция может использоваться с датчиком освещенности для затемнения экрана в зависимости от условий окружающей среды. Он также может быть использован для уменьшения или уменьшения яркости экрана.

Поворот экрана и зеркалирование

Таким же образом, что экран может быть инвертирован без влияния на данные дисплея, можно перевернуть экран в горизонтальном или вертикальном направлении. Это означает, что экран можно использовать в любом случае, без каких-либо изменений программного обеспечения, за исключением отправки на экран дополнительных двух команд.

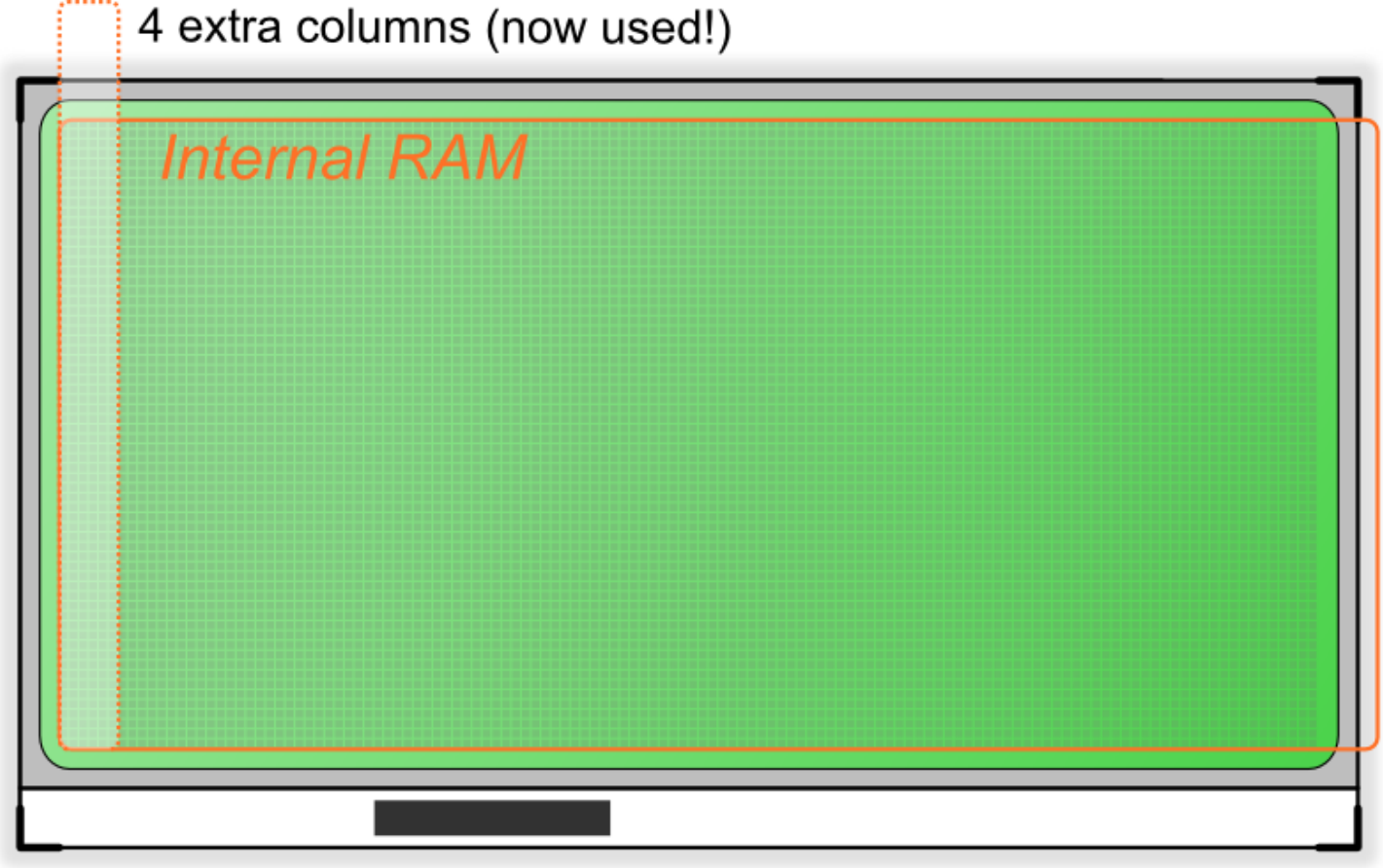
Перевернуть экран по горизонтали (слева направо) можно с помощью запутанно названной команды «Выбор АЦП». Перевернуть экран по вертикали (сверху вниз) можно с помощью команды «Выбор режима вывода COM». Если вы не хотите просматривать экран в зеркале, необходимо использовать обе команды одновременно!

Как упоминалось ранее, память на экране фактически имеет ширину 132 столбца, чтобы поддерживать максимальный размер экрана 132x64. На моделях шириной 128 пикселей важно вращать дисплей. Если экран не поворачивается, так как дополнительные 4 ряда находятся справа и не видны, как показано ниже:



ST7565 - Normal orientation

Когда экран поворачивается, контроллер выводит эти столбцы в обратном порядке, *начиная с последнего столбца* . Поэтому неиспользуемые 4 столбца из экранной памяти выводятся первыми, оставляя пустое пространство слева.



ST7565 - Rotated

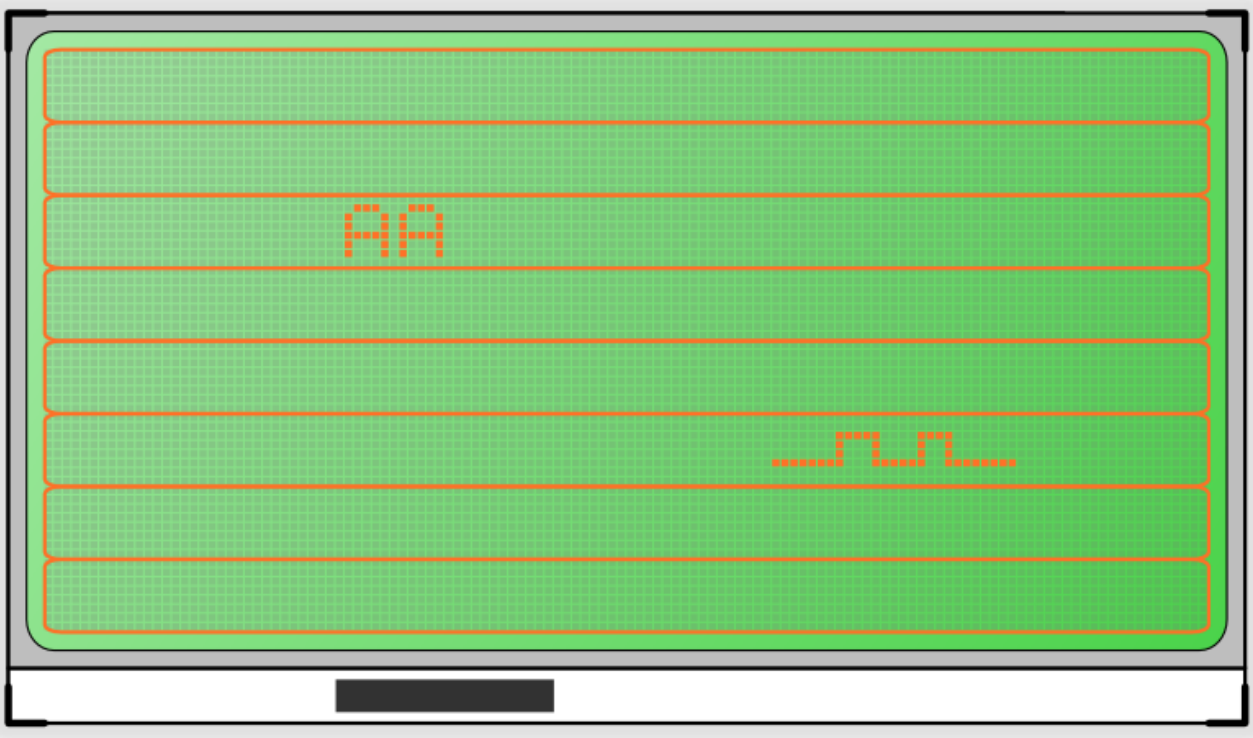
Чтобы исправить это, необходимо отслеживать в программном обеспечении, поворачивается экран или нет. Когда он поворачивается, адрес столбца должен увеличиваться на 4 каждый раз, когда на экран отправляется новая строка, пропуская дополнительные столбцы, которые в противном случае отображались бы.

Повышение эффективности

Обновление экрана включает в себя отправку всего локального буфера в экранную память. Если было сделано всего несколько изменений, то это очень неэффективный способ обновления экрана.

Можно было бы отправить каждую отдельную запись на экран, просто скопировав один байт, который изменялся каждый раз при записи пикселя. Однако это неэффективно, если в последующих операциях в одном байте установлено более одного бита (или пикселя). Изменения на экране также будут очень заметны на более медленных скоростях.

Поскольку на экране 8 страниц, самый эффективный метод на небольшом микроконтроллере - отправлять только те страницы, которые были изменены. Каждый раз, когда пиксель устанавливается или очищается, соответствующая страница в локальном буфере помечается как «грязная». Когда дело доходит до обновления экрана, переносятся только грязные страницы.



ST7565 - Improving efficiency

В приведенном выше примере изменения были внесены только на страницах 2 и 5. Поэтому они будут отправлены на экран, который занимает $^{1}/_{4}$ времени, как отправка всех 8 страниц.

Экран можно разделить на еще более мелкие области, например, на 16 половинок страниц. Компромисс дополнительного кода каждый раз, когда устанавливается пиксель, и сокращенное время отправки данных по шине должно быть тщательно продумано.

Проверка скорости

Выше я прокомментировал, что было возможно запустить PIC на частоте 64 МГц и не нарушать время SPI. Важные значения приведены в таблице 28 таблицы:

Table 28

(V_{DD} = 3.3V, Ta = –30 to 85°C)

Item	Signal	Symbol	Condition	Rating		Units
				Min.	Max.	
4-line SPI Clock Period	SCL	T_{scyc}		50	—	ns
SCL “H” pulse width		T_{shw}		25	—	
SCL “L” pulse width		T_{SLW}		25	—	
Address setup time	A0	T_{SAS}		20	—	
Address hold time		T_{sah}		10	—	
Data setup time	SI	T_{sds}		20	—	
Data hold time		T_{SDH}		10	—	
CS-SCL time	CS	T_{css}		20	—	
CS-SCL time		T_{csh}		40	—	

Нас интересуют Tscyc, Tshw и Tslw. Они относятся к общему такту, ВЫСОКОЙ части цикла и НИЗКОЙ части соответственно. Общий тактовый импульс должен составлять *не менее* 50 нс, а верхняя и нижняя части должны быть *не менее 25* нс каждая.

Хруст номера

Предположим, что код SPI включает тактовый вывод, а затем снова немедленно выключается в 2 инструкциях. Контакт будет ВЫСОКИМ в конце первой инструкции и снова НИЗКИМ после второй, поэтому он будет ВЫСОКИМ в течение периода 1 инструкции.

В семействе микроконтроллеров PIC 12/16/18 каждая инструкция занимает 4 цикла. При 64 МГц это означает, что каждая инструкция занимает 1 / ($^{64000000}/_{4}$) секунд или 62,5 нс. Исходя из этого, мы можем ожидать, что период ВЫСОКОГО тактового импульса должен составлять около 62,5 нс, что легко в течение требуемого времени.

Проверка с помощью логического анализатора

Также возможно измерить и посмотреть, как долго длится импульс на реальном оборудовании. На снимке экрана ниже показана ВЫСОКАЯ часть измеряемого тактового импульса, это самая короткая часть цикла. Используется PIC 18F25K22, работающий на частоте 64 МГц. Код включает и выключает контакт в двух инструкциях.

Помните, что он должен быть длиной не менее 25 нс, чтобы быть приемлемым. Измерение показывает 41,7 нс, что меньше, чем рассчитано, но все же приемлемо. Ряд причин может привести к разнице между нашими расчетами и физическими измерениями:

- Разрешение логического анализатора (в данном примере 24 МГц) может быть недостаточным для измерения.
- Установка или очистка битов в оборудовании может занять не все 4 цикла.
- PIC может не работать на частоте 64 МГц из-за нестабильности генератора или ФАПЧ.

Смотрите также

Если вы заинтересованы в использовании экрана на основе ST7565 в проекте Arduino, вам следует ознакомиться с руководством Ladyada по Adafruit. Доступна моя библиотека C,которая была разработана для деталей PIC 18F с HiTech C. Было бы тривиально заменить другой компилятор или чип.