FLASH память

Доброе утро, день, вечер.

При программировании на ПК обычно мы не беспокоимся о кол-ве памяти, но на ESP-32 нельзя забывать о памяти. С его выделением, освобождением, чтением есть множество особенностей. Начнем изучение с FLASH памяти, которая обычно уже распаяно в ESP-32. Самая распространненый вариант - 4МБ, с ним и будем работать.
Флеш память не энергозависимая, а значит мы можем решить проблему сохранения данных после перезагрузки.

Первое что нужно сделать - это правильно разбить память. По умолчанию esp-idf разбивает flash память самостоятельно, но мы можем это изменить. Выбрать один из профилей разбивки или создать свой. Прежде чем создавать - нужно узнать как строится эта таблица, какие есть типы и подтипы разделов, какие данные в них хранятся и какого размера они могут быть.

Какие типы разделов допустимы?

- APP - сюда можно записать данные прошивки.
- DATA - любые прикладные данные
- В документации можно увидеть ANY - это функция для поиска раздел, лучше использовать конкретный тип, чтобы избежать неожиданных результатов
  Помимо типа раздела, нужно еще указать подтип и для APP и DATA они отличаются

Начнем с APP:

- factory - это раздел на котором хранится прошивка, если есть OTA разделы, то он будет запасным, служить как резервная копия. Если OTA обновление завершится с ошибкой, то загрузка производится в factory раздела. Этого раздела может не быть при наличии OTA.
- ota_от 0 до15. 0x10 до 0x1F. Слоты для OTA прошивок. Сюда помещается код прошивки полученной на лету. Over the Air. Если factory раздела нет, то код с кабеля так же прилетит сюда.
- test - зарезервированный подтип для заводских испытаний. Резервный загрузчик раздела, если не найден factory или ota. Можно настроить на чтение strapping pin, которые мы разбирали в предыдущем видео.
  Для разделов APP должно быть выравнивание по 64КБ 0x10000 и размер не менее скомпилированного кода.

Подтипы DATA:

- ota 0x00 - раздел данных OTA, в котором хранится информация о выбранном слоте OTA в данный момент. Обязателен при наличии OTA разделов типа APP. Он фиксирован 0x2000 8кб байт.

- phy 0x01 - хранит данные для инициализации устройства phy. По умолчанию данные хранятся в самом приложении. Такой раздел позволяет не прошивать при каждом изменении phy констант.
- nvs 0x02 - энергонезависимое хранилище NVS с API. Используется для данных калибровки PHY каждого устройства, хранения данных WIFI, а самое главное для хранения своих данных. То есть записывать туда значение своих переменных. Ограничений у nvs много, например максимальный размер под одним ключом 1984 байта. Если нужно хранить большие данные, то этот тип не подойдет, т.к. придется разбивать на несколько ключей. Фрагментация так же имеет место.
- nvs_keys 0x04 - используется для хранения ключей шифрования NVS, если оно включено. Размер всегда 0x1000 4 кб байт.
- coredump - хранит данные при возникновении ошибок. Снимок состояния системы на момент сбоя, включая значения регистров процессора, стек, состояния задач и другую полезную информацию для отладки.
- efuse_em - хранит данные для работы с eFuse и аварийным режимом. Может хранить данные для восстановления устройства в аварийном режиме, например резеврный загрузчик или минимальную прошивку. просто efuse - это область данных доступная только для чтения, с параметрами, которые для этого предназначены.
- esphttpd - используется для хранения данных, связанных с веб-сервером, html страницы, css файлы, js файлы и др. ресурсы.
- fat - позволяет организовать систему fat, которая может хранить огромные файлы по меркам esp-32, но несет за собой много лишнего кода и логики
- spiffs - более легковесная файловая система, специально для микроконтроллеров и флеш памяти с высокой производительностью. Максимальный размер файла - 4мб.
- littlefs - легковесная файловая система для микроконтроллеров, но с поддержкой директорий, в отличии от spiffs, а так же отказоустойчивая. Но spiffs тянет за собой меньший объем кода и имеет меньшие накладные расходы, что бывает критично. В разных сценариях выигрывают разные фс по производительности.

Перед созданием своей таблицы вспомним еще про некоторые ограничения:

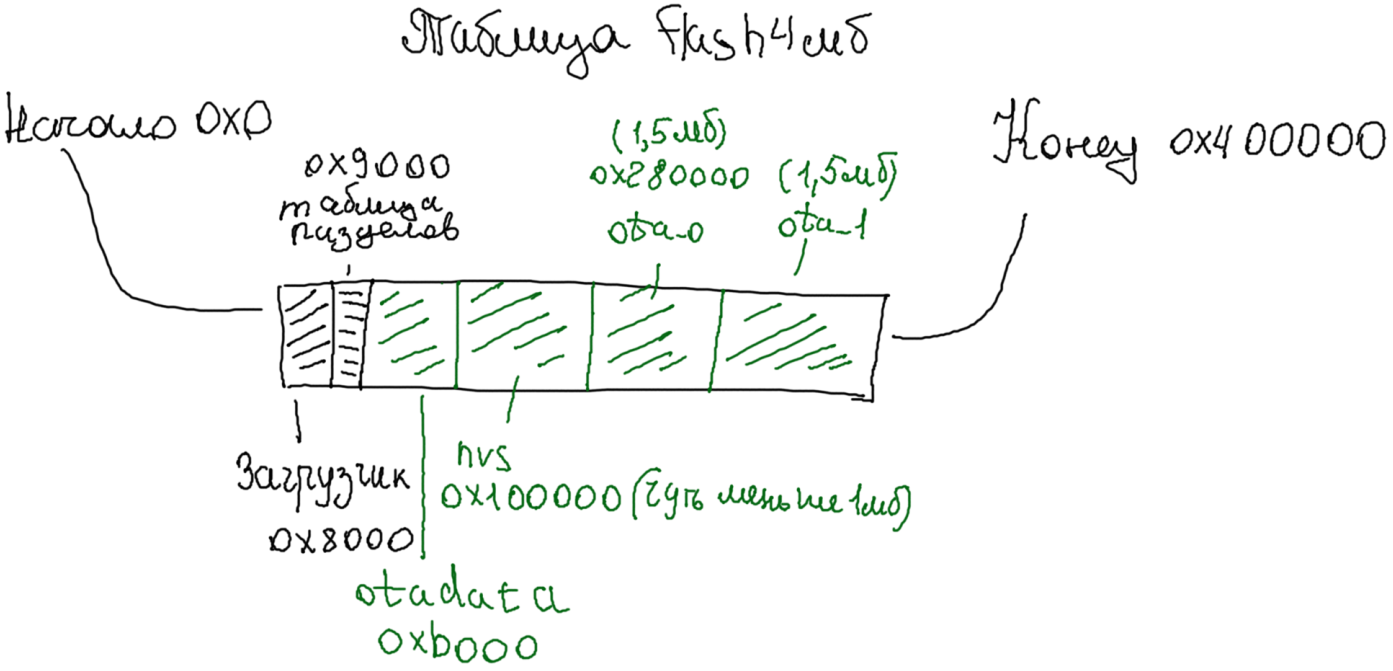
- начальная область памяти занята под загрузчик, если он используется и следующие разделы лежат с адреса лежащего в константе CONFIG_PARTITION_TABLE_OFFSET
- Минимальный размер сектора в flash памяти - 4 КБ.
- Максимум может быть 95 записей. Таблица разделов записана сразу после загрузчика, то есть начало таблицы будет в CONFIG_PARTITION_TABLE_OFFSET + 0x1000

Перейдем к созданию своей таблицы:

```
# Name,   Type, SubType, Offset,   Size, Flags
otadata,  data, ota,     0x009000, 0x002000,
nvs,      data, nvs,     0x00b000, 0x0f5000,
```

```
app0,     app,  ota_0,   0x100000, 0x180000,
app1,     app,  ota_1,   0x280000, 0x180000,
```

Посмотрим на схему



Начало выделено под загрузчик и саму таблицу разделов. Далее идет наша разметка, она обозначена зеленым цветом. Otadata раздел обязательный при использовании ota, который хранит выбранный ota. nvs чуть меньше 1 мб в котором мы можем хранить свои данные во время работы программы и два раздела ota по 1,5 мб под саму прошивку.

А теперь посмотрим на практике:

- Переходим в каталог с нашим проектом `cd ~/esp/led`
- Запускаем alias и первого видео `idf`
- Вызываем `idf.py menuconfig` для настройки таблицы разделов
- Выбираем `Partition Table->Partition Table` и видим 4 предустановленных варианта разметки и один кастомный
- Выбираем кастомный раздел `Custom partition table CSV`
- Создаем файл `partitions.csv` и пишем туда:

```
  # Name,   Type, SubType, Offset,   Size, Flags
otadata,  data, ota,     0x009000, 0x002000,
nvs,      data, nvs,     0x00b000, 0x0f5000,
app0,     app,  ota_0,   0x100000, 0x180000,
app1,     app,  ota_1,   0x280000, 0x180000,
```

- Сохраняем файл и выходим
- Прошиваем таблицу разделов `idf.py partition-table-flash`

Готово. Мы разобрались с возможными вариантами разметки flash памяти и теперь может это делать под свои нужды. У Вас может возникнуть много вопросов по работе OTA,

хранилищу NVS или работе с файловыми системами. Все это мы разберем в следующих видео.

Спасибо за просмотр