Прошивка аппаратов методом клонирования.

Сначала была идея. Появилась примерно одновременно с появлением одноаппаратных прошивок. Она долго витала в облаках. Суть идеи проста, сменить серийный номер в прошивке и шить ей что угодно... Но идея быстро потухла, так как отличить бит от байта с моими знаниями трудно. Близился Новый Год и в праздничной кутерьме пришла та-же мысль, только задом наперёд. Будем думать, как править серийный номер в аппарате. Наверно каждый знает, что в PCL принтерах HP номер, да что номер, даже счётчик пробега можно без проблем править PGL командами. Для этого служат соответствующие команды, которые достаточно отправить в аппарат. Но увы, в замечательной технике Samsung это не работает. Это GDI принтеры, полный лайт. Собственно на данный момент просто поменять серийный номер не получилось. Однако... Были начаты опыты по ковырянию содержимого NVRAM. Выпаяна, серийный номер найден, заменен. Увы, системная распечатка не показала изменённого номера. Идея была позаброшена, Новый Год... Но неугомонный Alexkyahta проявил инициативу и продолжал колдовать. И наколдовал немного магии. Вот его сообщение на форуме, где он сообщает о первой удаче.

Для эксперимента был взят совершенно новый аппарат SCX-3200. Для начала выпаял nvram (U17 4256bwp), считал с него дамп, соответственно сохранил. Далее исправил в нем серийный номер в двух местах на тот серийник, к которому есть фикс. Запаял обратно микросхему. Включил аппарат и .. радостно засветился "зеленый глаз". Распечатал тестовую страницу - в ней в строке machine serial number : видим серийник , который вбили вместо "родного". Залил фикс, распечатал отчет. В отчете появилась F. Заклеил чип - аппарат радостно зеленый! Снова отчет - полет нормальный. Ну и конечно возвратил оригинальную прошивку. Положил на витрину. Теперь у меня в продаже два аппарата с одним серийником.

Суть такова. Серийный номер хранится в микросхеме NVRAM U17 (SCX-3200) с маркировкой 4256BPW. В ней, в числе прочего содержится и серийный номер. Он записан в двух местах (речь о SCX-3200, в ML-1660 серийник один). Микросхема эта несмотря на заумную маркировку, представляет простую микросхему 12C². В принципе, практически такая же как и те, что когда-то устанавливались в чипы SCX-4200. Соответственно прекрасно читается программой PonyProg2000 с настройками 12C Bus 16bit eeprom 24256 или любым другим программатором с поддержкой протокола 12C². Соответственно, достаточно выпаять микросхему, поправить серийные номера на те номера, на которые изготовлена ваша прошивка и зашить дамп в микросхему. После впаивания микросхему стоит убедится в том, что аппарат успешно изменил серийный номер распечатав системную распечатку. Если номер совпадает с требуемым, можно смело заливать прошивку с данным серийным номером. Кроме того в NVRAM содержится и счетчик отпечатанных страниц, именно по этому советую править свой дамп, а не использовать дамп из таблицы.

В принципе, этот способ подойдет для любого аппарата. Различий не так много. Например в ML-1660 NVRAM содержится в микросхеме 93C66. И тоже замечательно читается PonyProg2000. Конечно же с настройками 93C66.

Как осуществить замену серийного номера технически?

Есть несколько способов. Первый из них, это выпаять микросхему NVRAM и прошить на программаторе. Простой и надёжный способ.

На данный момент известны случаи, когда микросхема NVRAM (точнее 24C256 на SCX-3200) перепрашивалась без выпаивания. Можно припаяться непосредственно к выводам на микросхеме. Для этого также можно использовать специальные клипсы.

Однако на данный момент существует потенциально более совершенный метод, причем не связанный с выпаиванием. Метод предложен одним из читателей. Суть метода в том, что серийный номер изменяется командами терминала. Также есть возможность слить всё содержимое NVRAM.

Ниже полностью приведена описание процесса от Mesko.

СМЕНА СЕРИЙНОГО НОМЕРА МФУ ЧЕРЕЗ DEBUG

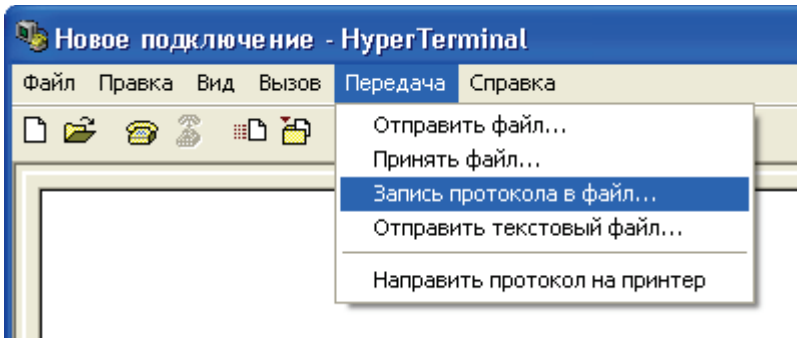
Сама идея смены серийного номера через Debug возникла в предположении, что при помощи достаточно мощного пакета команд сервисного режима удастся изменить флэш-память U17 24C256 без выпаивания данной микросхемы. Для чего это нужно: во-первых, это дает возможность сдать аппарат по гарантии, ведь номер можно легко вернуть на место, во-вторых, этот метод более быстр и безопасен чем выпаивание. Конечно, Debug также предполагает подпаивание к разъему, но при достаточно умелых действиях данное подключение осуществить проще, чем выпаивать флэш-память. Во время написания данной статьи у меня также возникла идея использовать токопроводящий клей для подключения Debug-а, я не смог протестировать его на практике, но, думаю, возможность использования его вместо олова существует.

Прочитать о сервисном режиме (Debug-е) можно здесь: http://www.alex-banzai.net/?action=forum&theme=441, также можно прочитать статью А. Лутова: МФУ Samsung SCX-4300, SCX-4200: Решение проблемы "Download From PC". Также необходимо знать, что на аппаратах серии SCX-4824(4828) необходимо распаять шину Rx, чтобы была возможность отсылать команды по Debug-у. Делается это небольшим наплавлением двух контактов(если хотите наплавлением перемычки). Местоположение перемычки можно увидеть на фото. Мне нет необходимости рассказывать о Debug-е, потому как о нем уже много чего написано, а потому я могу перейти к рассмотрению флэш-памяти, где хранится серийный номер.

Флэш-память использует протокол записи I2C, который поддерживается самим устройством (МФУ). Адрес eeprom = 0x000000AE и размер = 32768 байт (или 0x8000 в шестнадцатеричном виде). Но обращение к памяти происходит по адресам 0 – 7fff. Немного странно, потому как первая ячейка это 0, поэтому и получается сдвиг на 1 байт. Собственно это все что нам нужно знать для редактирования флэш-памяти. Мне неизвестно всегда ли флэш-память будет иметь адрес eeprom =

0x000000AE, но все аппараты то, что я правил имели именно этот адрес. Следует также помнить, что вся адресация команд Debug-a, а также смещения оперируют с шестнадцатеричными значениями.

Итак, мы уже разобрались с флэш-памятью и готовы приступить к поиску серийного номера внутри нее. Но как это сделать? Для начала мы подключаемся по интерфейсу Debug к нашему МФУ, лучше использовать для подключения программу HyperTerminal из средств Windows (сам я использовал Windows XP). Подключение происходит по способу описанному в статьях выше и после успешного соединения переходим в режим ожидания ввода команд: pROBE+>. Чтобы ознакомиться со списком команд, можно ввести Help, но мы будем пользоваться только двумя основными командами – **reada**(и аналогом **read**) и **writea**. Давайте сразу договоримся, прежде чем записывать изменения во флэш-память ОБЯЗАТЕЛЬНО делайте **Запись протокола в файл** в программе Hyper Terminal. Делается это для того чтобы можно было, если что пойдет не так отслеживать вносимые изменения и исправлять их. Старайтесь вести протокол всегда, когда работаете с Debug-ом, в трудную минуту протокол событий – это незаменимое средство для восстановления.



Итак, мы в режиме ожидания ввода команд pROBE+>. Чтобы вывести нашу флэш-память, нам следует знать ее адрес и размер. Адрес мы знаем – AE , размер тоже 8000. Теперь остается воспользоваться командой **reada.** Вводим:

eeprom reada 0 AE 0 1 8000

Чтобы понять, что мы ввели, давайте разберем команду:

eeprom reada – вывод данных (сама команда)

0 –канал (я всегда ставлю в 0, но существует еще значение 1)

AE – как нетрудно догадаться это адрес флэш-памяти

0 – смещение (т.е. с какого места нужно выводить информацию, здесь можно указать и 20 и 4A и 4B5 главное не перебрать за значение 8000)

1 – тип (ставьте всегда в 1)

8000 – размер (сколько нужно вывести байт от смещения, у нас смещение = 0, значит и выводим с этого места 0x8000 байт)

Но команда дает нам ошибку так как буфер переполнен, он, видите ли, не выводит больше 800. Ну ладно воспользуемся другой командой **read**, может она поможет. А **reada** поможет в будущем, чтобы вывести исправленный серийный номер.

Вводим заново:

eeprom read 4 8000

4 – устройство, это наша флэш-память с адресом AE. (Трудно сказать, почему она 4, потому как я ставил и 2 и 3 и получал такой же результат, при 1 выводится другая флэш-память).

Чтобы не попасть впросак, посмотрите в протоколе выходной заголовок флэш-памяти и убедитесь, что eeprom address равен AE:

i2c chan : 0

eeprom address = 0x000000AE

offset address = 0x00000000

morethan 256 = 1

eeprom size = 32768

page number = 64

8000 – размер (сколько нужно вывести байт от смещения, у нас смещение = 0, значит и выводим с этого места 0x8000 байт)

Следует отметить, что описание команд **read**, **reada**, **write** и **writea** вы можете вывести командой Help, но я думаю в этом нет необходимости, так как все аргументы мы расписали. Теперь я могу вас поздравить, вы стали обладателем дампа флэш-памяти. Вы можете спросить меня, можно ли использовать данный дамп для записи его внешним программатором. Конечно да, но предварительно придется написать программу по выдергиванию байтов из протокола, что пока не входит в наши планы. Научившись считывать флэш-память, мы теперь можем найти в ней наш серийный номер. Допустим номер нашего аппарата Z501BFAZ913900B (SCX-3200), но в таком виде вы его не увидите, потому как это ASCII символы, которые представлены во флэш-памяти слегка в преобразованном виде, а точнее каждый символ в памяти представлен своим шестнадцатеричным значением (согласно таблице ASCII). Я не стал приводить всю таблицу, а лишь символы, которые могут встречаться в номере. Сверху приведены сами символы, внизу их шестнадцатеричные значения.

0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	G	H
30	31	32	33	34	35	36	37	38	39	41	42	43	44	45	46	47	48
I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
49	4A	4B	4C	4D	4E	4F	50	51	52	53	54	55	56	57	58	59	5A

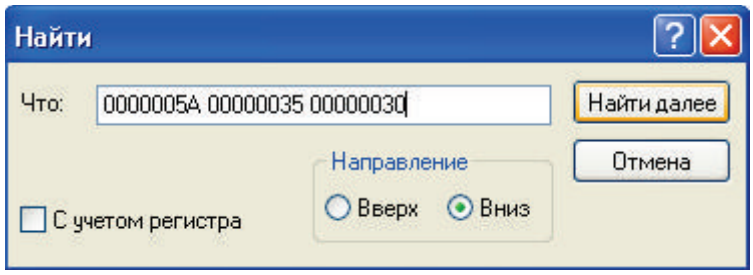
После перевода номера в шестнадцатеричные значения мы получили для своего номера Z501BFAZ913900B такую последовательность байтов **5A 35 30 31 42 46 41 5A 39 31 33 39 30 30 42.** Давайте теперь откроем наш протокол событий и посмотрим как устроены наши данные, после того как мы вывели командой **read** нашу флэш-память. Открываем наш протокол (я использовал **блокнот**) и смотрим наш дамп:

[00000020] 0000005A 00000035 00000030 00000031 00000042 00000046 00000041 0000005A

[00000028] 00000039 00000031 00000033 00000039 00000030 00000030 00000042 00000000

[00000030] 00000000 00000008 00000000 00000004 00000000 00000000 00000000 00000016

[00000038] 00000000 00000009 00000000 00000004 00000000 00000000 00000000 00000000



Я специально привел кусочек с нашим серийным номером, но нам пока неизвестно где он. Чтобы узнать, где он находится, нам следует воспользоваться текстовым поиском для нашего протокола. В строке поиска нам следует указать небольшую последовательность байтов известной нам. Это может быть и 0000005A 00000035 00000030, а может и 0000005A 00000035 00000030 00000031 00000042 00000046. Чем больше строка поиска, тем меньше вероятность нахождения похожих данных. Но в любом случае если Вы нашли несколько мест с номером вам необходимо сверить все байты, чтобы удостовериться, что это действительно номер, а не похожая последовательность байтов. Дальше если все байты сходятся необходимо переписать смещение начального байта, в данном случае смещение начального байта 0x00000020. Таким образом мне удалось выяснить, что серийный номер для SCX-3200 прячется по двум смещениям:

0x00000020

0x00000D10

для SCX-4824 это одно смещение:

0x00002618

Нужно ли находить каждый раз смещения? Я думаю нет, нам будет достаточно один раз вычислить смещения для разных серий МФУ.

Теперь уже более-менее понятно, что нужно сделать для смены серийного номера. Для начала, как ни странно, нужно знать сам номер, на который мы хотим сменить, и его шестнадцатеричное значение. Давайте возьмем получивший большое распространение номер Z501BFBZ801816T и запишем его в памяти eeprom. Чтобы его записать нужно перевести его в шестнадцатеричный вид **5A 35 30 31 42 46 42 5A 38 30 31 38 31 36 54.** Теперь когда у нас есть номер в шестнадцатеричном значении, мы можем его записать. Для записи мы будем использовать команду **writea**. Команда почти такая же как и reada, за исключением того что в конце дописывается 1 байт данных который мы хотим записать. Внизу я составил таблицу символов. Таблица разделена на две части. В первой части - 15 смещений (по длине номера) первой записи номера, во второй - 15 смещений второй записи. Следует отметить, что мы пробуем на SCX-3200 поэтому у нас 15*2=30 смещений. Для SCX-4824 это будет 15 смещений, так как номер во флэш-памяти записан в одном месте, а не в двух как в SCX-3200.

0x00000020	5A	5A	0x00000D10	5A	5A
0x00000021	35	35	0x00000D11	35	35
0x00000022	30	30	0x00000D12	30	30
0x00000023	31	31	0x00000D13	31	31
0x00000024	42	42	0x00000D14	42	42
0x00000025	46	46	0x00000D15	46	46
0x00000026	41	42	0x00000D16	41	42
0x00000027	5A	5A	0x00000D17	5A	5A
0x00000028	39	38	0x00000D18	39	38
0x00000029	31	30	0x00000D19	31	30
0x0000002A	33	31	0x00000D1A	33	31
0x0000002B	39	38	0x00000D1B	39	38
0x0000002C	30	31	0x00000D1C	30	31
0x0000002D	30	36	0x00000D1D	30	36
0x0000002E	42	54	0x00000D1E	42	54

Для наглядности оранжевым указано старое значение, а зеленым новое записываемое значение. Как видно из таблицы некоторые символы повторяются и нам можно поменять только 8 из них (в итоге 8*2=16). Но, к сожалению, мы не знаем, какой номер будет у нас в будущем, поэтому лучше перезаписать все 30 символов серийного номера, так надежнее.

Вот как это выглядит.

eeprom writea 0 AE 20 1 1 5A

eeprom writea 0 AE 21 1 1 35

eeprom writea 0 AE 22 1 1 30

eeprom writea 0 AE 23 1 1 31

eeprom writea 0 AE 24 1 1 42

eeprom writea 0 AE 25 1 1 46

eeprom writea 0 AE 26 1 1 42

eeprom writea 0 AE 27 1 1 5A

eeprom writea 0 AE 28 1 1 38

eeprom writea 0 AE 29 1 1 30

eeprom writea 0 AE 2A 1 1 31

eeprom writea 0 AE 2B 1 1 38

eeprom writea 0 AE 2C 1 1 31

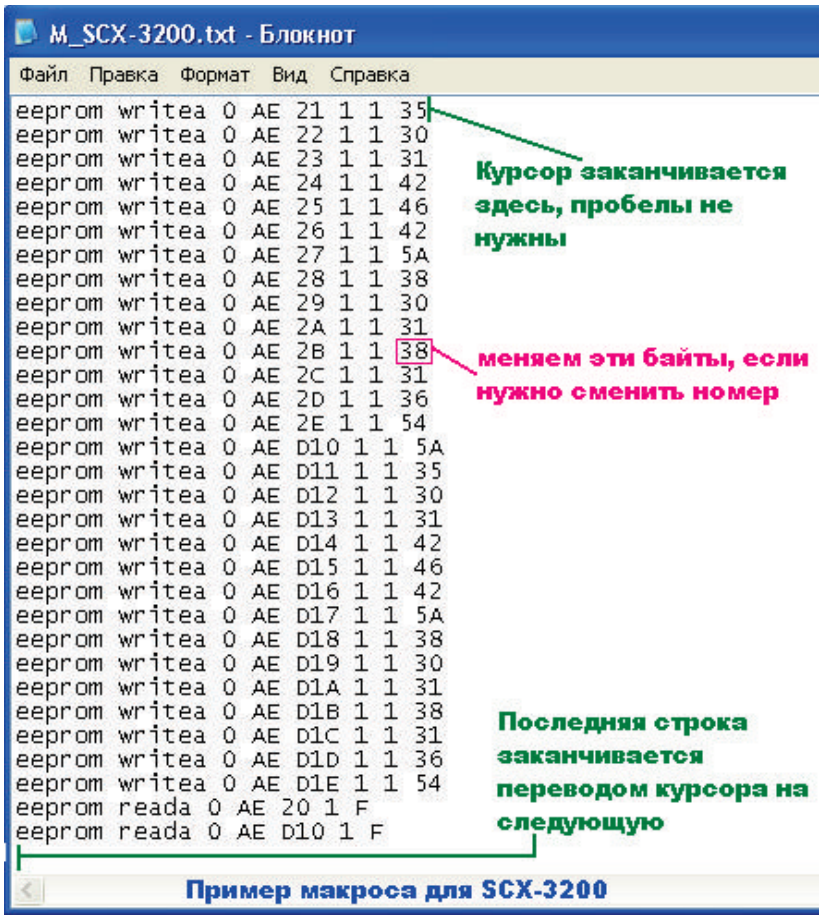
eeprom writea 0 AE 2D 1 1 36

eeprom writea 0 AE 2E 1 1 54

eeprom reada 0 AE 20 1 F (выводим наш измененный номер для сверки)

Соответственно следует также поступить и со второй колонкой и также ввести 15 команд. Вот и все мы изменили наш номер. Для проверки распечатываем **Configuration Report [Отчет конфигурации],** и смотрим **Machine Serial Number:** и если он стал Z501BFBZ801816T операция удалась.

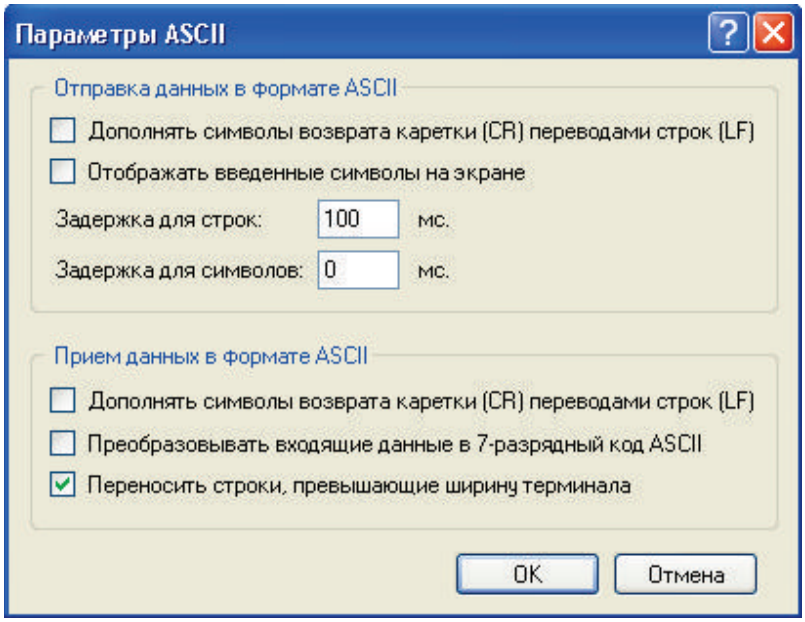
Может показаться, что мы справились с задачей, номер мы изменили, но мне как-то стало не по себе, когда я ввел 30 команд набирая их на клавиатуре. Но больше всего меня расстроило отсутствие макросов в HyperTerminal-е. В этот момент я стал подумывать над автоматизацией процесса, чтобы не вводить вручную каждый раз 30 команд. Мне пришлось перелопатить, несколько терминальных программ, я даже скачал новую версию HyperTerminal. Теперь она называется HyperTerminal Private Edition 7.0. Программа поддерживается все той же компанией Hilgraeve Inc, но теперь стала trial-ой с 30-дневным периодом, затирающей своего предшественника без возможности сохранения (Hyper-вирус наверно). Внешне (да и внутренне, похоже, тоже) программы почти не отличаются друг от друга, но увидев в новой версии возможность ввода макросов, я уже было обрадовался, но столкнувшись на практике с этой «хорошей» функцией (трудно подобрать соответствующее слово) я вдруг из оптимиста превратился в пессимиста. Мало того что в окошко ввода нельзя вставить из буфера команду, так еще и записать макрос у меня не получилось, при выходе из программы макрос сбрасывался. Время шло, я стал подумывать, что смысла продолжать эту идею нет и хотел свести все к идеи медленного альтернативного варианта. Но вдруг мне пришла в голову мысль, предпосылкой которой явилась увиденная мной в HyperTerminal-е функция **Отправить текстовый файл**. По сути это и есть тот же макрос, но только через текстовый файл. Здорово – это то что нам нужно. Значит, теперь мы можем свести все наши команды в текстовый файл, где каждая команда разделена переводом строки (т.е. тем же **Enter**-ом). Сделать такой файл очень просто. Создаем текстовый файл в блокноте и заносим туда наши команды, все команды разделяются переводом строки. Старайтесь не допускать пробелов после перевода строки(команды), это конечно не страшно и работать скорее всего будет, но все же. Последняя строка(команда) заканчивается простым переводом строки на новую также без пробелов.



Сказано – сделано. В приложении я дал 2 примера макросов - M_SCX-3200.txt и M_SCX-4824.txt, они работоспособны, но требуют сменить байты номера, если номер вам не подходит. После создания макроса нам необходимо отправить его функцией **Отправить текстовый файл.** И, о чудо, действие пошло. К сожалению, когда я впервые отправил макрос на выполнение радость моя была не долгой, макрос остановился не выполнив даже 2 команду. А в место следующей команды

вывелся «огрызок». Откуда ехали туда и приехали.

(Нецензурными словами здесь ругаться не принято, а жаль, много было высказано в этот момент в адрес разработчиков HyperTerminal-a). И начался долгий период изысканий и анализа в чем может быть проблема. А проблема оказалась достаточно банальна, мы слишком быстро посылали команды. Т.е. нам нужна задержка в выполнении команды. Ну где ее взять если макросов нет. На нашу радость, эту возможность программисты из Hilgraeve Inc предусмотрели. Открываем в программе HyperTerminal **Файл** – **Свойства** – вкладка **Параметры** – **параметры ASCII строк.** И в окошке **Задержка для строк** изменяем значение с 0 на 100**.** Здесь можно ввести любое другое значение, может, у кого шустрый компьютер, тогда придется поставить значение побольше. И наконец, после повторного выполнения макроса свершилось то к чему МЫ шли так долго. Ура!



В заключение могу отметить, что выводить каждый раз всю флэш-память нет необходимости, а можно просто добавить предварительный просмотр той области памяти, которую изменяет макрос. Это увеличит скорость и если случится сбой, мы сможем восстановить эту область из протокола.

Пример для SCX-4824:

eeprom writea 0 AE 2618 1 1 36 – макрос до изменения

eeprom writea 0 AE 2619 1 1 35

……..

eeprom writea 0 AE 2625 1 1 31

eeprom writea 0 AE 2626 1 1 4B

eeprom reada 0 AE 2618 1 F – эту команду можно поставить первой и тогда мы увидим затираемые значения.

eeprom reada 0 AE 2618 1 F – макрос после изменения

eeprom writea 0 AE 2618 1 1 36

eeprom writea 0 AE 2619 1 1 35

……..

eeprom writea 0 AE 2625 1 1 31

eeprom writea 0 AE 2626 1 1 4B

eeprom reada 0 AE 2618 1 F

В общем, решать вам, пользоваться ли выводом полной флэш-памяти или ограничится лишь той областью, которую мы изменяем.

Итоги

Итак, можно подвести итоги: Данный метод не легок, но при достаточно внимательном отношении будет проще в реализации, чем выпаивание флэш-памяти. По своей доступности он больше годится для сервисных инженеров, чем для домашних пользователей, но естественно он не является панацеей от всех бед, это лишь один из способов облегчить жизнь сервисного инженера. Это другой взгляд на проблему. Лучше он или хуже выпаивания покажет время, но я думаю можно использовать их вместе. Выпаивание больше применимо к принтерам, а смена через Debug к МФУ.

Mesko

Практические рекомендации

1. Как должен работать этот метод?

Если мы знаем номер на замену, мы можем заранее подготовить комплект команд для смены серийного номера под определенный аппарат.

Пример: возьмем SCX-4824

Считываем флэш-память и вычисляем смещение серийного номера. Берем нужный номер и переводим в шестнадцатеричное значение согласно таблице ASCII, составляем 15 команд вводим их в текстовый файл(макрос) и отправляем его через Debug, воспользовавшись командой **Отправить текстовый файл**, предварительно увеличив задержку.

2. Для чего нужно выводить флэш-память?

Во-первых, для резервного копирования на случай случайного изменения при попытке правки номера; во-вторых, чтобы посмотреть сам номер в памяти и вычислить смещение. Теперь самое главное, номер хранится в памяти по определенным адресам. Не могу сказать, верно ли это для разных версий прошивок аппаратов одной серии, но похоже это действительно так.

3. Метод работает долго, нужно вычислять смещение каждый раз заново? Можно ли ориентироваться на дампы прочитанные программатором?

Отчасти так, но если заранее позаботиться о составлении макроса, и привязать его к определенной прошивке пользоваться им будет проще, чем выпаивать каждый раз микросхему. Т.е. запустили Hyper Terminal, выполнили макрос и все, далее только сверить номер и убедиться что все хорошо.

Если правильно считывать дампы не искажая адреса, то на них можно ориентироваться, потому как смещения одинаковы, но лучше пользоваться просмотром памяти - надежнее.

4. Я что-то сделал не так, номер не изменился, а макрос что-то записал не туда?

Действия такого рода можно разделить на 2 типа:

1. Такое действительно возможно, если изменяться смещения серийного номера во флэш-памяти либо поменяется адрес устройства. Я использовал адрес **AE**, но возможно вместо **AE** у кого-то будет другой адрес. Скорее всего, это маловероятно, но исключать такой поворот событий нельзя.
2. Если пользователь неправильно ввел макрос или использовал макрос от другого аппарата (человеческий фактор).

Лекарством для всех этих случаев будет просмотр предварительно снятой флэш-памяти и нахождение новых адресов с последующей правкой испорченных байтов индивидуальным макросом. Помните я просил считать флэш-память используя **Запись протокола в файл**, как раз на этот случай.

5. Макрос не выполняется до конца?

Вы забыли поставить задержку или следует увеличить ее значение.

6. Какая команда лучше reada или read, write или writea? И почему нельзя записать номер одной командой?

По функционалу лучше использовать команды reada и writea, но команда reada не выводит за один раз более 0x800 байтов и поэтому для вывода флэш-памяти мы используем обычный **read**. Небольшая проблема в том, что команда **read** использует вместо адреса устройство и вопрос может возникнуть в том, а что если у кого-то флэш-память **AE** не соответствует устройству 4, хотя это маловероятно, но возможно.

Потому что команда writea(да и write) записывают по одному байту. Если бы эта команда смогла записывать более 1 байта информации, без макроса можно было спокойно обойтись,

скопировав команду на выполнение из буфера (команда **Передать главному компьютеру).** Но, к сожалению это не так…

7. Можно ли воспользоваться этим методом для сброса счетчика страниц?

Общий счетчик сбросить можно, только нужно отследить смещения. Счетчик картриджа сбрасывать нет смысла, так как он дублируется показаниями чипа.

8. Существует ли возможность прошивать EEPROM через USB, минуя Debug?

Это единственный вопрос, который остался для меня нерешенным. В командах EEPROM-а, нет упоминания о загрузке какого-либо образа либо куска данных извне. Но кто знает, возможно, нам повезет и появится возможность заливать дамп наподобие прошивки.

Скачать прошивки, дампы и прочее можно здесь: http://www.testcopy.ru/laser-printer-mfp/20-format-a4/125-2011-02-04-15-40-56.html

Скачать последнюю версию данной статьи можно здесь:
http://www.testcopy.ru/images/stories/smena_seriinika/klonir.pdf

Максимов Денис

Банзаракцаев Алексей ака Alexkyahta

Mesko