Преподу: Основной кусок бака. Пока описал только про IDA – почему выбрал ее. В разработке. Читать и ругать только то, что до pause. Поправил, дописал 6 страниц. Сейчас повествования начинается с выбора среды дизассмеблирования -> выбор симулятора и связка -> анализ полученного кода (это еще даже до середины не дописано)

Исходными данными проекта явилась прошивка некоего устаревшего устройства (на базе TMS32054). В связи с этим был разработан и предложен следующий цикл разработки программного обеспечения:

1) Дизассемблирование прошивки.
2) Анализ и тестирование полученного кода.
3) Написание кодека на языке Си.
4) Первичное тестирование.
5) Модернизация кодека.
6) Повторное тестирование кодека.
7) Анализ полученных результатов.

Этап первый. Дизассемблирование.

Дизассемблирование – это ключ к исходному коду любой программы, будь это компьютерный вирус или любой другой бинарный файл. Транслирование исходного файла на язык ассемблер позволяет увидеть и изучить алгоритмы работы программы. Актуальность технологии бесспорна, несмотря на то, что технология является узкопрофильной. Именно поэтому сред дизассемблирования немного. К наиболее известным и активно используемым средам можно отнести: Sourcer, Hiew и IDA Pro. Каждый из них доказал свое право на существования, однако использовать будем только один - IDA Pro. Этот выбор обуславливается положительными и отрицательными сторонами всех трех дизассемблеров:

1) *Sourcer* – один из простейших дизассемблеров. Позволяет конвертировать исходный файл в ассемблерный код, однако, примерно в 30% случаях требуется вмешательство программиста для исправления ошибок дизассемблирования, что усложняет отладку большой программы. Так же отсутствуют базы данных и интерактивность интерфейса – это неимоверно сказывается на удобстве, скорости и качестве обратной разработки.
2) *Hiew* - редактор двоичных файлов, ориентированный на работу с кодом. Имеет встроенный дизассемблер для x86, x86-64 и ARM, ассемблер для x86, x86-64 [1]. Кроме того, данный редактор распространяется бесплатно. Однако в связи с тем, что Hiew разработан в начале 90-ых и с момента создания интерфейс не обновлялся. Программа не удобна при работе, полное отсутствие интерактивности и командный режим только усложняет работу. Кол-во поддерживаемых форматов входных файлов и процессоров сильно ограничено.

NEW!!

3) *IDA Pro* (англ. Interactive DisAssembler) — интерактивный дизассемблер, отличается исключительной гибкостью, наличием встроенного командного языка. Поддерживает множество форматов исполняемых файлов для большого числа процессоров и операционных систем [2]. Среда рассчитана на подключение отдельных модулей, расширяющих функциональность среды. Последние версии (IDA SDK) дают возможность пользователям самостоятельно разрабатывать модули. Разрабатываемые модули могут быть отнесены к одному из трех классов:
    - процессорные модули – позволяют разбирать программы для новых процессоров;
    - загрузчики – необходимы для поддержки новых форматов входных файлов;
    - плагины – модули, расширяющие функциональность IDA (добавление новых команд, улучшение анализа посредством установления обработчиков событий [3])

На данный момент IDA является лидером среди дизассемблеров на рынке. Позволяет в автоматическом и полуавтоматическом режиме проводить дизассемблирование. Продвинутый уровень интерактивности позволяет изменять названия ф-ий и переменных, сохраняет искомую позицию, записывает изменения в базу данных (с возможным экспортом), позволяет выполнять переходы по адресам двойным кликом мыши. Кроме того, следует отметить, что существуют как платные, так и бесплатные версии данной среды, что позволяет ее использовать всем желающим. Создателем среды является Ильфак Гильфанов, разработчик с многолетним опытом и специалист по компьютерной безопасности [3].

Как видно, выбор Interactive DisAssembler оправдан. Среда обладает рядом функциональных преимуществ перед конкурентами, а кроме того поддерживает дизассемблирование программ, написанных для используемого нами TMS32054. Для работы интерактивным дизассемблером IDA необходимо выполнить следующие действия:

а. Запускаем среду IDA. **File -> New ->Binary File** (Рис. 1). После выбираем файл прошивки (aces.bin);

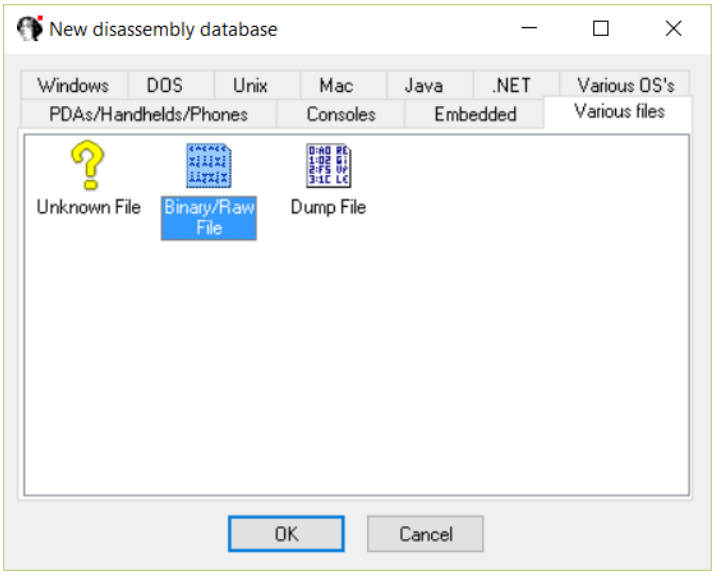

Рис. 1 – Выбор типа входного файла.

б. Затем откроется окно выбора типа процессора. Выбираем **TMS32054** (Рис. 2).

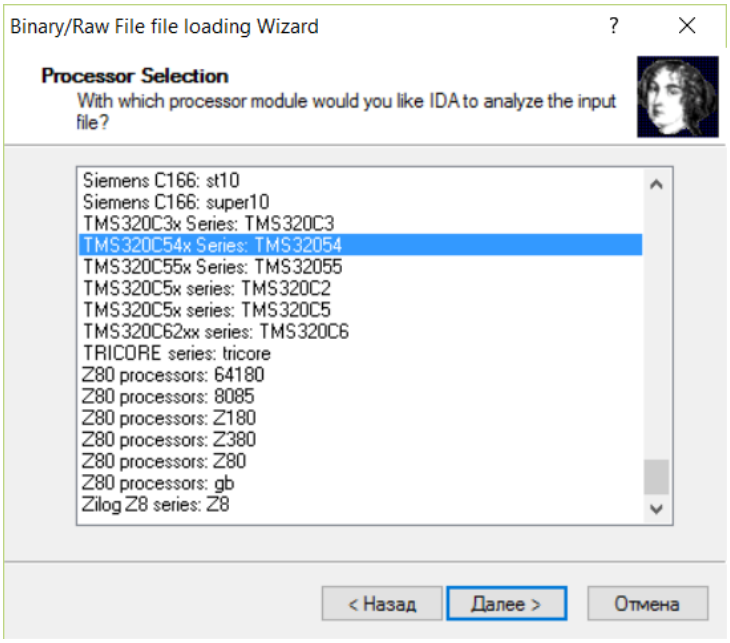

Рис. 2 – Выбираем нужный процессор.

в. После выбора процессора IDA предложит указать участок памяти, куда будет загружена прошивка (Рис. 3). Так как задача стоит получить искомый код прошивки - изменений не вносим, оставляя все по умолчанию.

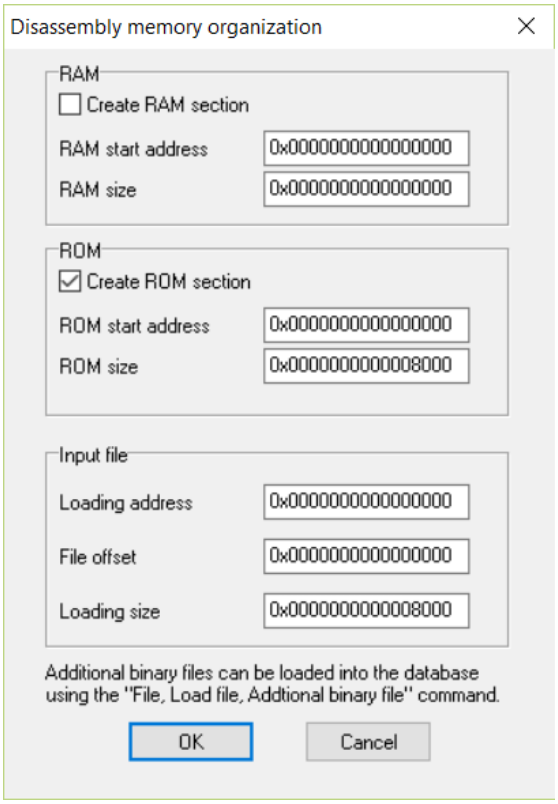

Рис. 3 – Выбираем нужный процессор.

г. Сразу после завершения настроек отроется окно IDA View-A (Рис. 4)

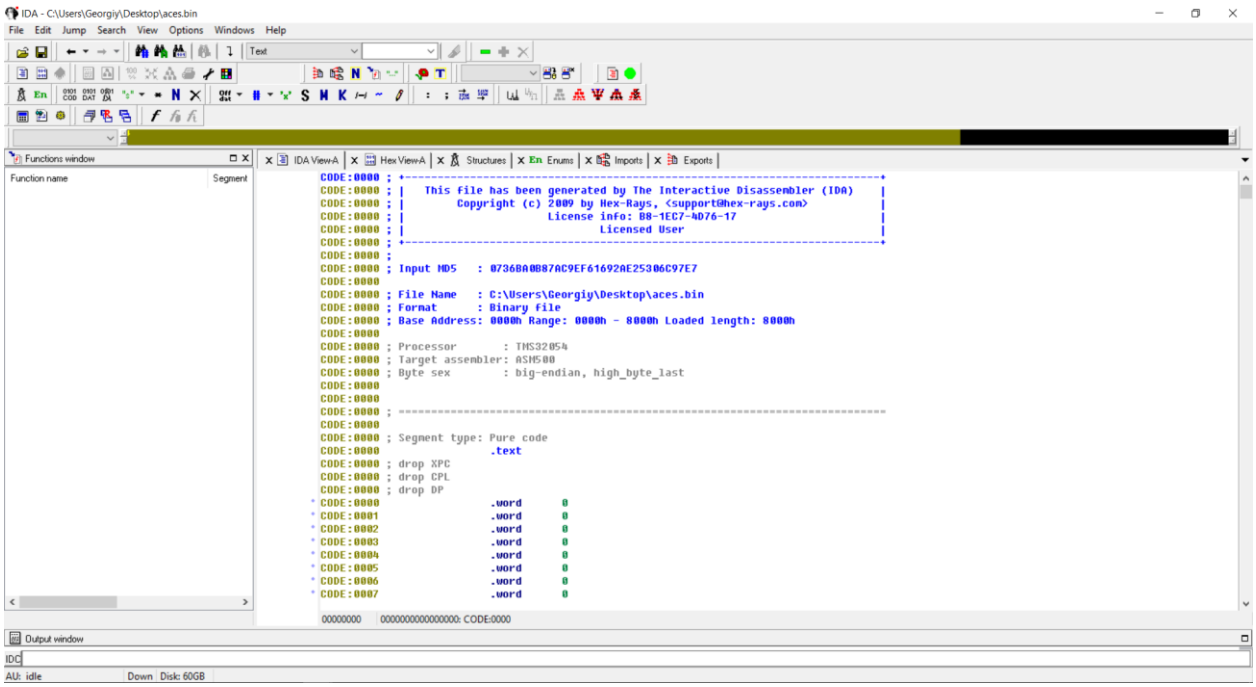

Рис. 4 – Основное окно разработки IDA View.

д. В открывшемся окне предоставлено содержимое бинарного файла. Устанавливаем курсор на нулевой адрес и жмем кнопку "C". Небольшие участки кода преобразуется в ассемблерные команды (Рис. 5). Перемещаясь по адресному пространству, пользователь выполняет преобразование бинарного кода в ассемблерные команды. IDA самостоятельно подписывает начало и конец функций, присваивая названия по адресу, например sub_2010. Если пользователь, проанализировав содержимое функции, осознал суть происходящего и желает изменить название функции или добавить комментарий – это выполняется двумя кликами ПКМ. Проводить статический анализ кода, в котором все функции подписаны и прокомментированы – намного удобнее, нежели в обычном ассемблерном файле.

```
CODE:200D                .word  140Ah
CODE:200E                .word 0F073h ; sЁ
CODE:200F                .word  200Eh
CODE:2010
CODE:2010 ; =============== S U B R O U T I N E =======================================
CODE:2010
CODE:2010 ; Attributes: noreturn
CODE:2010
CODE:2010 sub_2010:                              ; CODE XREF: CODE:5E53↓p
CODE:2010                                        ; CODE:5E56↓p
CODE:2010                adds    *(byte_140A), A
CODE:2012                ld      #1400h, 0, B
CODE:2014                and     #0FFFFh, B
CODE:2016                sub     A, B
CODE:2017                rcd     bgeq
CODE:2018                stl     A, *(byte_140A)
CODE:201A
```

Рис. 5 – Дизассемблированный участок.

Однако, для того, чтобы найти участок ассемблерного кода, который содержит кодек, статичного анализа недостаточно. Для удобного анализа кода потребуется симулятор микропроцессора. Симулятор позволит анализировать изменения регистров, аккумуляторов и памяти, что упростит первостепенную задачу – задачу обнаружения кодека. В связи с тем, что микропроцессор TMS32054 является разработкой компании Texas Instrument [4], в качестве симулятора будет использоваться Code Composer Studio [5] – симулятор, разработанный той же фирмой для тестирования своих собственных процессоров. Данная среда зарекомендовала себя на рынке достойным образом, поэтому выбор стал очевидным. Существуют как платные версии, так и бесплатные с ограниченным набором микропроцессоров. Необходимый нам для работы микропроцессор TMS32054 попадает как в платную версию программы, так и в бесплатную.

Среда состоит из двух модулей: ***CCStudio(CCS)*** и ***CCStudio Setup(CCSS)***. Перед использованием симулятора необходимо установить тип симулируемого процессора в CCSS (рис. 6)

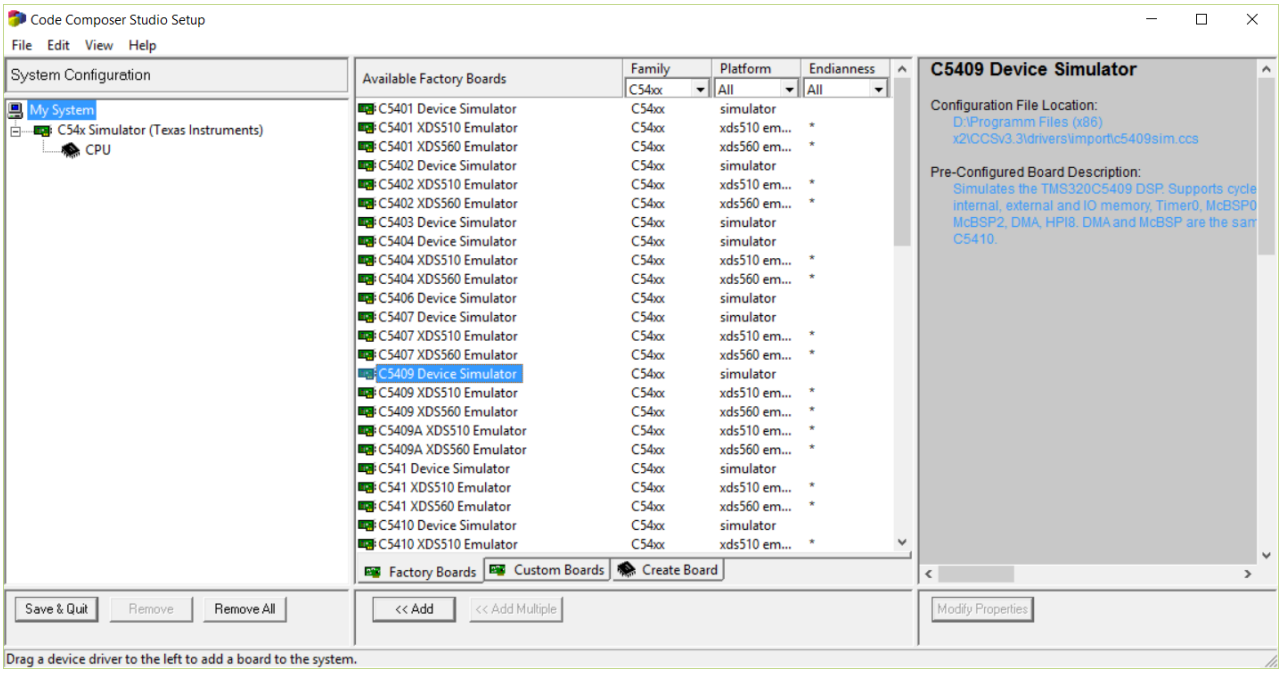

Рис. 6 – Настройка CCS.

После того, как нужный процессор найден, добавляем его к используемым (**<<Add**) и запускаем ***CCS*** (**Save&Quit**). Среда перезапуститься и откроется основное окно CCS. Отобразить нужные окна можно установив соответствующие галочки во вкладке ***View***. Потребуется следующие три окна: окно памяти, окно разборки и окно основных регистров (рис. 7).

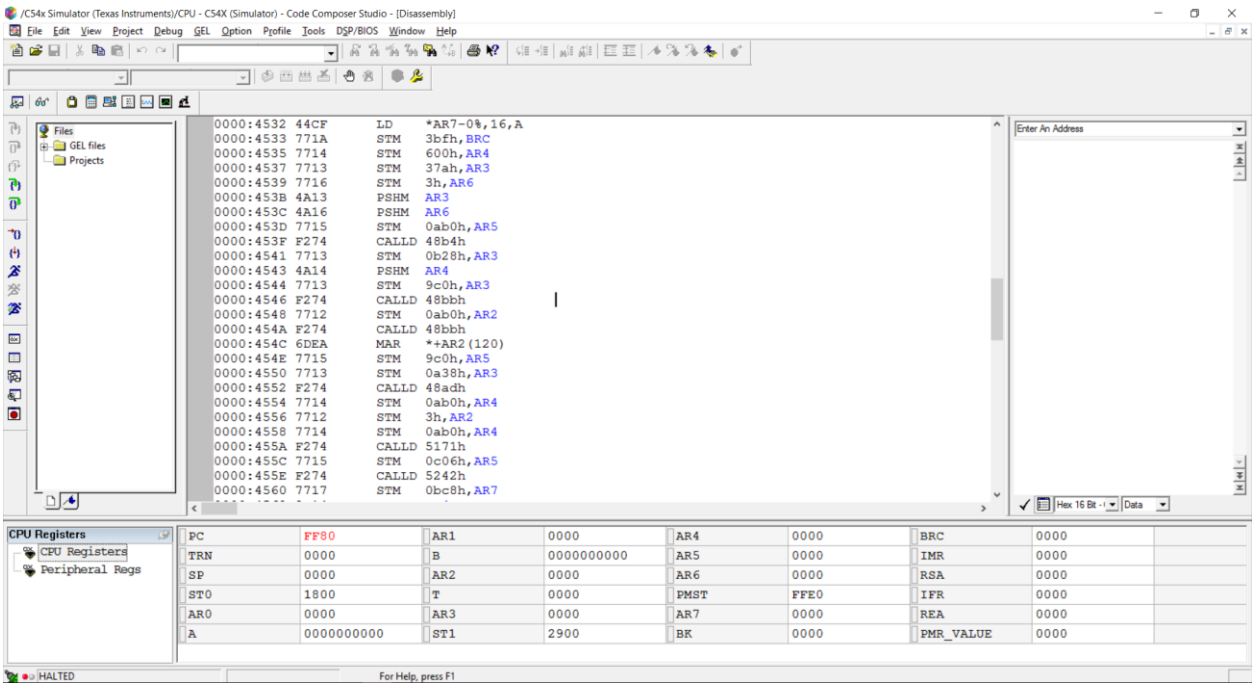

Рис. 7 – Окно Code Composer Studio.

После настройки окон необходимо загрузить саму прошивку в симулятор и данные. Так как в данном микропроцессоре используется Пристанская архитектура (как и в большинстве современных компьютеров), и данные и исполняемые коды хранятся вместе, в одной памяти. Дизассемблированная прошивка поступает на вход CCS в виде dat-файла, содержащем необходимые заголовке, ассемблерные команды и данные в 16-ом коде. IDA способна сгенерировать файл содержащий все команды в 16-разрядном коде, однако ни заголовки, ни формат данных не подходит для CCS. Для корректной работы на вход CCS необходимо подать файл с 16-битным словами формата 1 (Рис. 8), с соответствующими настройками в заголовке файла [6]:

- 1651 – корректная сигнатура для правильного распознавания и считывания;
- 1 – формат подаваемых чисел (1 – это 16-разрядный формат). Существуют так же целые (2), типа long (3), float (4);
- 0 - начальный адрес;
- 0 - текущая страница;
- FFFF - конечный адрес.

```
1651 1 0 0 ffff
0x0000
0x0000
0x0000
0x0000
0x0000
0x0000
0x0000
0x0000
0x0000
0x0000
0x0000
0x1100
0x0000
```

Рис. 8 – Формат dat-файла для CCS.

IDA способна сохранить результат дизассемблирования в 16-битном формате типа 2 (рис. 9).

```
2185  7313 0062 F010 0003 8061 ED01 FA43 2191 E589 4592 4761 CB89 8793 F000 0002 8061
2195  8816 F000 0001 8814 7664 01FF 11E4 6524 8163 F77F 8910 7715 0011 7212 0060 7213
21A5  0062 721A 0061 F420 F272 21B4 7711 0040 6F85 0D59 1964 8914 6DB1 3093 28E4 6000
21B5  F00F 0001 8292 4910 6E8E 21A4 0163 8910 FC00 F000 0014 8060 F274 2318 F062 0008
21C5  FC00 F110 0018 FA4E 21D9 F020 6666 F300 0009 FA4F 21D9 F020 4CCC F300 000F 890E
21D5  F020 2666 F067 028F FC00 F6B8 F310 0001 891A F310 0008 890E F300 000F 44F8 000B
21E5  F48C F47F F57E F300 6320 8913 F030 0003 F562 890E 7714 0008 F004 FFFF F272 2201
21F5  1593 F764 F000 0001 962D FA20 2201 F180 8392 1193 F010 0004 F764 F7B8 FC00 F6B8
2205  F6B9 F310 0009 F761 F300 652F 8913 771A 0007 F272 2215 1093 3C83 F130 000F 8192
2215  F47C FE00 F7B8 F7B9 7311 0065 8064 8163 7312 0062 7313 0061 8060 F274 2318 F062
2225  0004 8066 3066 2061 F468 F00F 0001 8267 1064 F010 0001 881A 8064 7712 0068 F272
2235  2247 F420 F7B6 0067 F130 03FF 890E F576 0162 8913 2163 6F69 0D65 4563 4369 8368
2245  A598 B394 8391 F6B6 1065 8812 8813 6F66 0C5E 880E F420 4764 2892 F00F 0001 4409
2255  E732 ED00 4593 F520 4764 C798 FC00 7312 0066 7313 0068 7314 0069 E808 F074 2000
2265  806A F7B6 7212 0068 E808 F274 2183 7213 006A F420 8067 6F68 0D00 8914 6F69 0D00
```

Рис. 9 – Формат сохранения результата от IDA.

Конвертация формата 2 (IDA) в формат 1 (CCS) выполняется при помощи небольшой программы, написанной на Си. Результатом работы программы станет dat-файл с необходимыми заголовками и нужным форматом данных. Он-то и будет подаваться на вход CCS.

Этап второй. Анализ полученного кода.

Перед вторым этапом разработки стоит целый ряд сложнейших задач:

- Найти функции кодека среди прошивки;
- Подтвердить работоспособность кодека, полученного после дизассемблирования;
- Провести анализ структуры кодека и его составляющих.

Поиск функции кодека.

Для упрощения задачи поиска функции кодека в прошивке, потребовалось еще несколько вариантов прошивки этого же процессора. Каждая из прошивок отличается внутренним состоянием области памяти. Удобно было взять:

1) Состояние памяти, до и после кодирования;
2) Состояние памяти, сразу после принятия данных и после декодирования.

Рассмотрим поиск функций на примере кодера. Анализируя прошивку до кодирования представилось возможным найти адреса, в которые записываются входные данные. Функции, которые обращаются к этим адресам были выделены в группу 1. Таких функций оказалось не много, так как прошивка занимает примерно четверть всей памяти, в следствии чего свободной памяти на кристалле много и функции редко обращаются к одним и тем же адресам без логической связи друг с другом. Затем была проанализирована прошивка, полученную после кодирования. Функции, которые записывают данные в адреса, в которых располагаются закодированные данные, были выделены в группу 2. Сопоставив функции групп 1 и 2, обнаружились функции, которые одновременно обращаются и к первым и ко-вторым адресам – это и есть основные функции кодера (выделили их в группу 3). После того, как примерное расположение функций стало очевидным. Однако этого недостаточно, необходимо полностью определить границы кодера, то есть какие близлежащие функции являются кодером, а какие нет. На данном этапе отлично проявил себя симулятор CCS. Выполняя и анализируя код функций группы три, и прилежащих функции можно удалось с точностью до команды определить, какие из функций принадлежат кодеру, а какие нет. CCS не предоставляет гибкого интерфейса, поэтому работу CCS удобно было комбинировать с IDA, оставляя пометки и комментарии в среде дизассемблера. Поиск декодера выполнялся подобным образом.

PAUSE

Подтвердить работоспособность кодека.

После того, как границы кодера и декодера были определены, потребовалось подтвердить работоспособность кода кодека после дизассемблирования. Тестировать отдельные участки кода оказалось намного проще, нежели всю прошивку целиком. Именно поэтому этап проверки кода идет только после обнаружения границ кодека.

Вероятность возникновения ошибки при дизассемблировании небольшая, т.к. IDA полностью поддерживает TMS32054. данный модуль является официальным, а значит относительно надежным) и способна в автономном режиме дизассемблировать код, однако: «Здоровое недоверие — хорошая основа для совместной работы» [7].

Во избежание повторений, проверка будет проведена на примере кодера. В симулятор (CCS) последовательно загружаются два dat-файла: первый - полученный после дизассемблирования с помощью IDA, и второй - файл с битами, которые необходимо закодировать. Выполняется кодирование, средствами симулятора. Результат кодирования должен совпадать с эталонной моделью не только по содержанию (бит в бит), но и по адресам расположения данных. Эталонная модель необходимо взять из результата работы самого устройства. Т.к. задача стоит подтвердить правильность дизассемблирования – трех, четырех тестов хватит. Нужды подтверждать работоспособность алгоритмов нет, т.к. плата является рабочей (хотя и устаревшей). Входные биты должны быть , в адреса подаются входные данные,

Литература:

1) Краткое описание Hiew. URL: https://ru.wikipedia.org/wiki/Hiew
2) Статья о возможностях IDA. URL: http://www.idasoft.ru/idapro/
3) Интервью с Ильфаком Гильфановым. URL: http://fcenter.ru/online/softarticles/interview/6704
4) Сайт фирмы, разработавшей процессор. URL: http://www.ti.com/
5) Раздел сайта TI, посвящённый CCS. URL: http://www.ti.com/tool/ccstudio
6) Описание заголовка dat-файла. URL: https://e2e.ti.com/support/development_tools/code_composer_studio/f/81/t/277303
7) Сталин в беседе с Б. Куном; С. 190