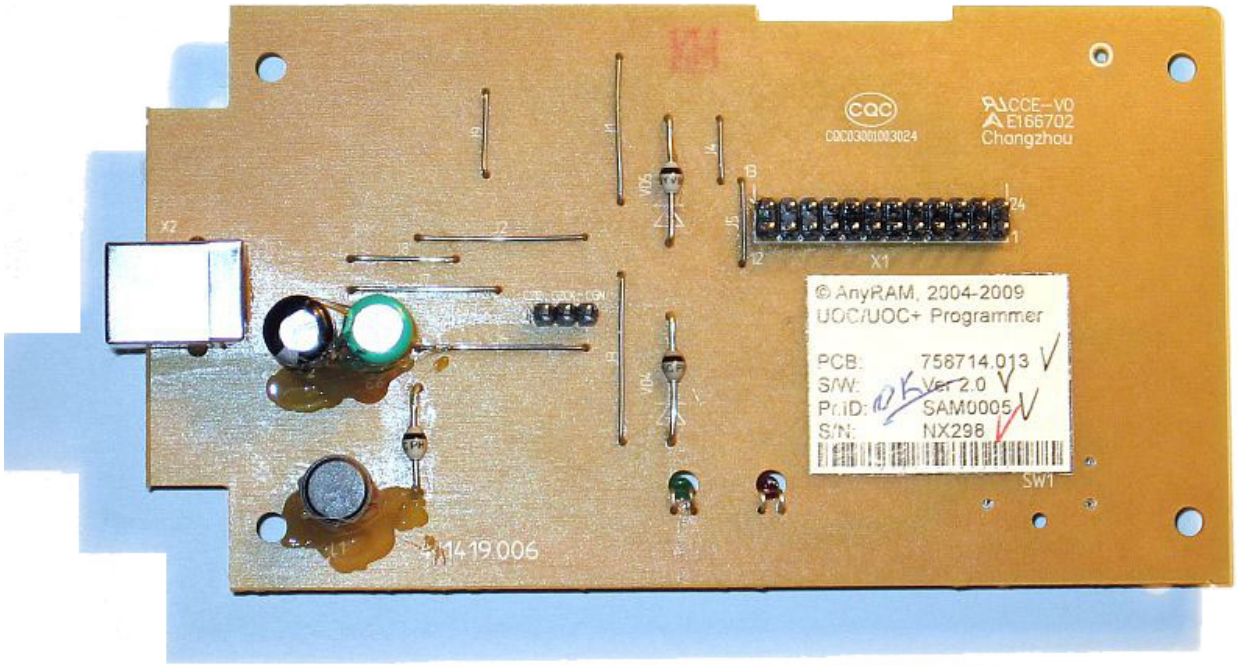
AnyRAM Production

Website: http://www.anyram.net
E-Mail: client@anyram.net

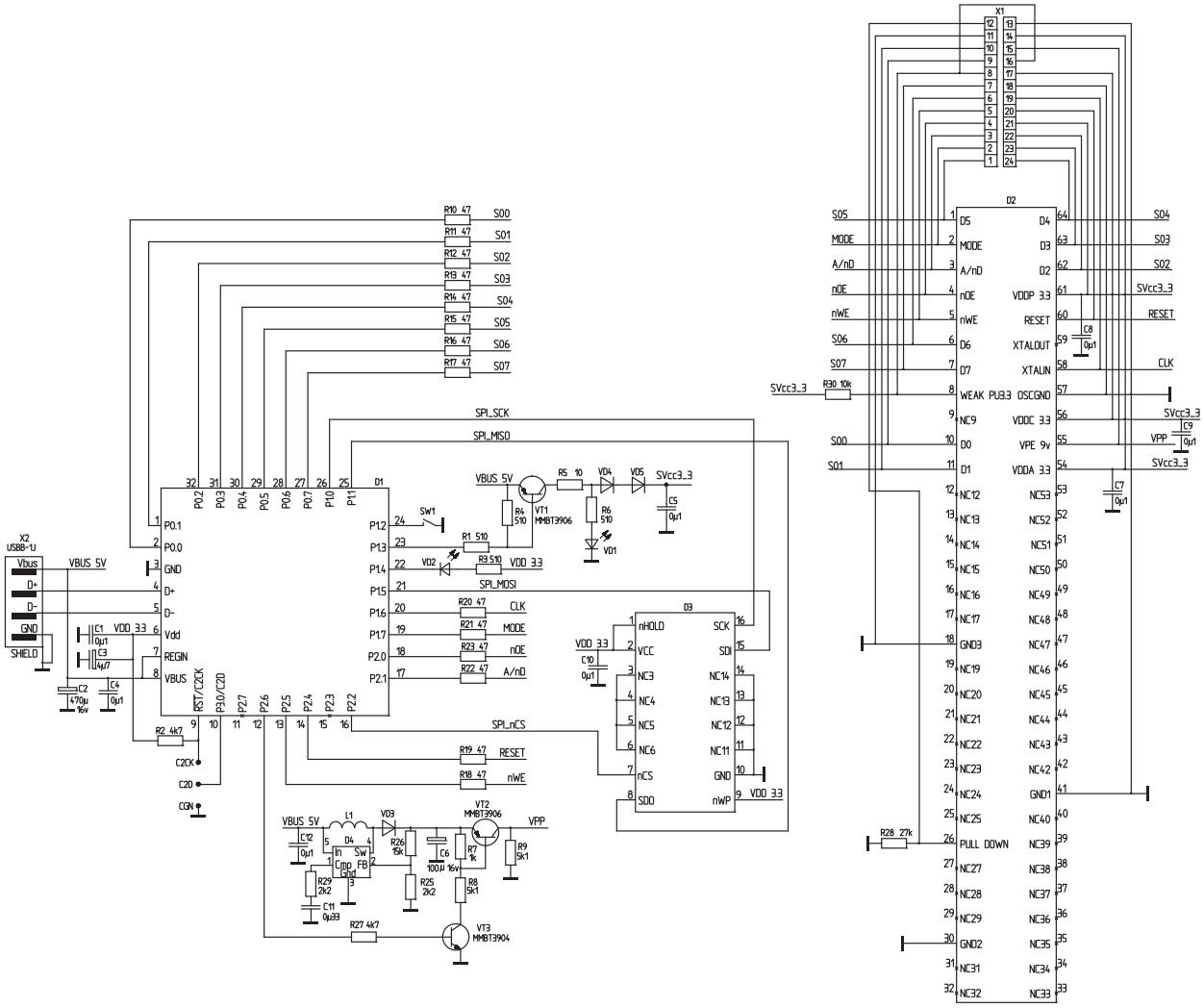
Краткое описание платы программатора UOC серии NX, TX

Программатор предназначен для программирования ИМС Philips/NXP Painter, Painter-II, UOC, UOC+, UOC-Leader, UOC-III, UOC-Top Fighter, UOC-Top Painter



© AnyRAM, 2004-2012

Принципиальная схема программатора



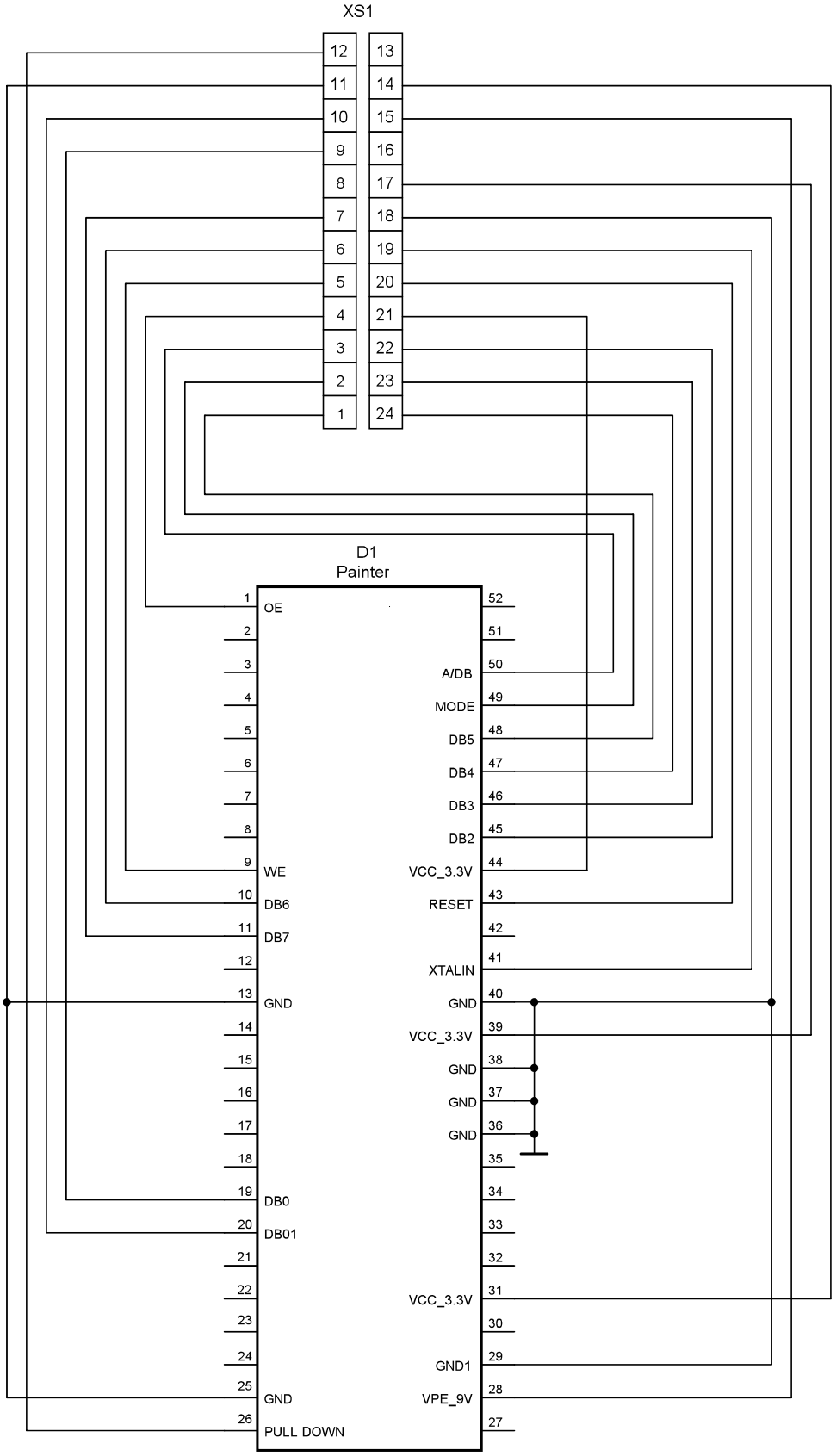
Сокет D2 на схеме используется для подключения UOC.

ИМС D3 в версиях программатора NX может не устанавливаться.

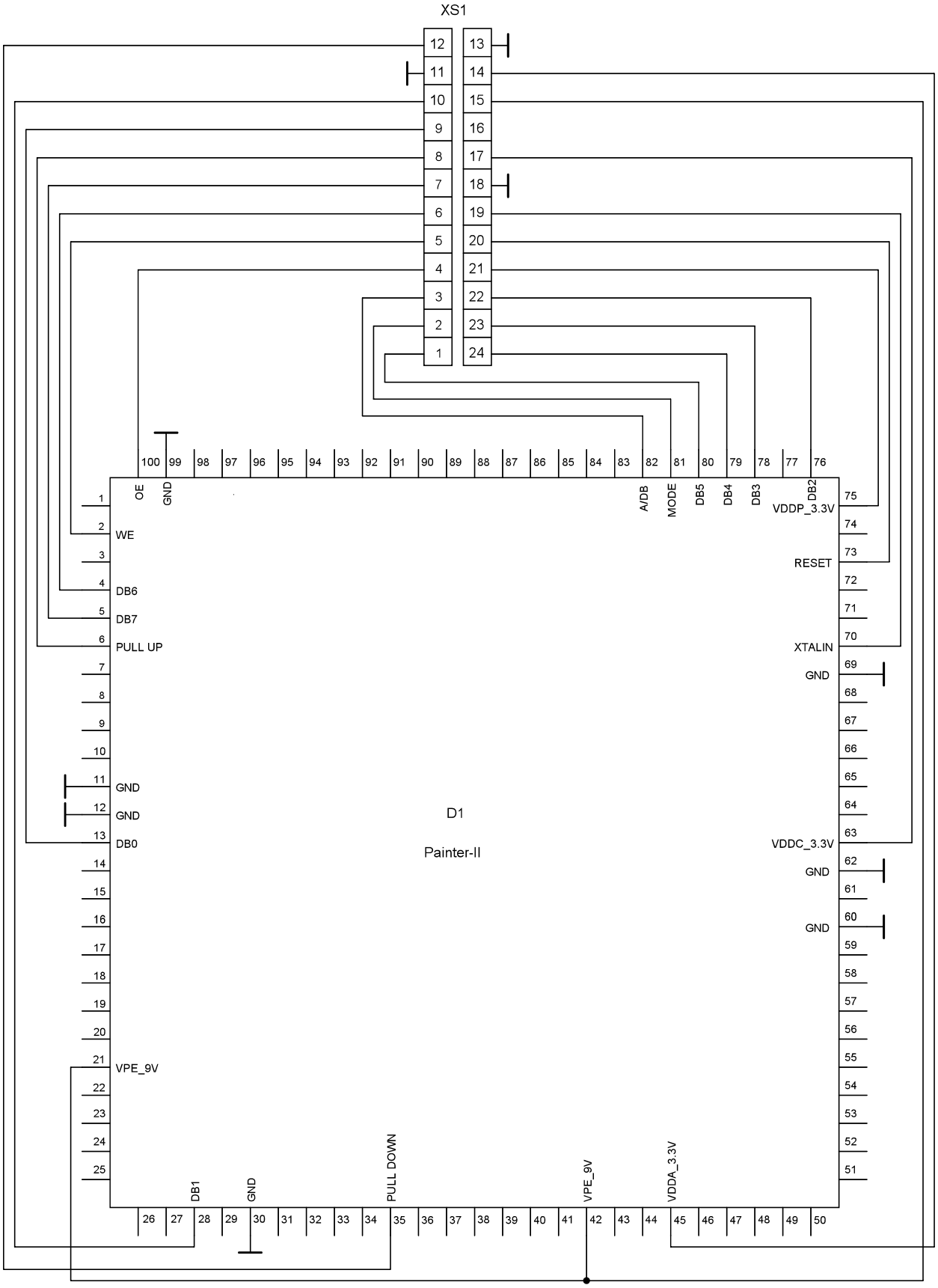
Кнопка SW1 в лабораторных версиях программатора не устанавливается.

VD1 – светодиод красного цвета, загорается при подаче напряжения на UOC.
VD2 – светодиод зелёного или синего цвета, индицирует режимы работы.

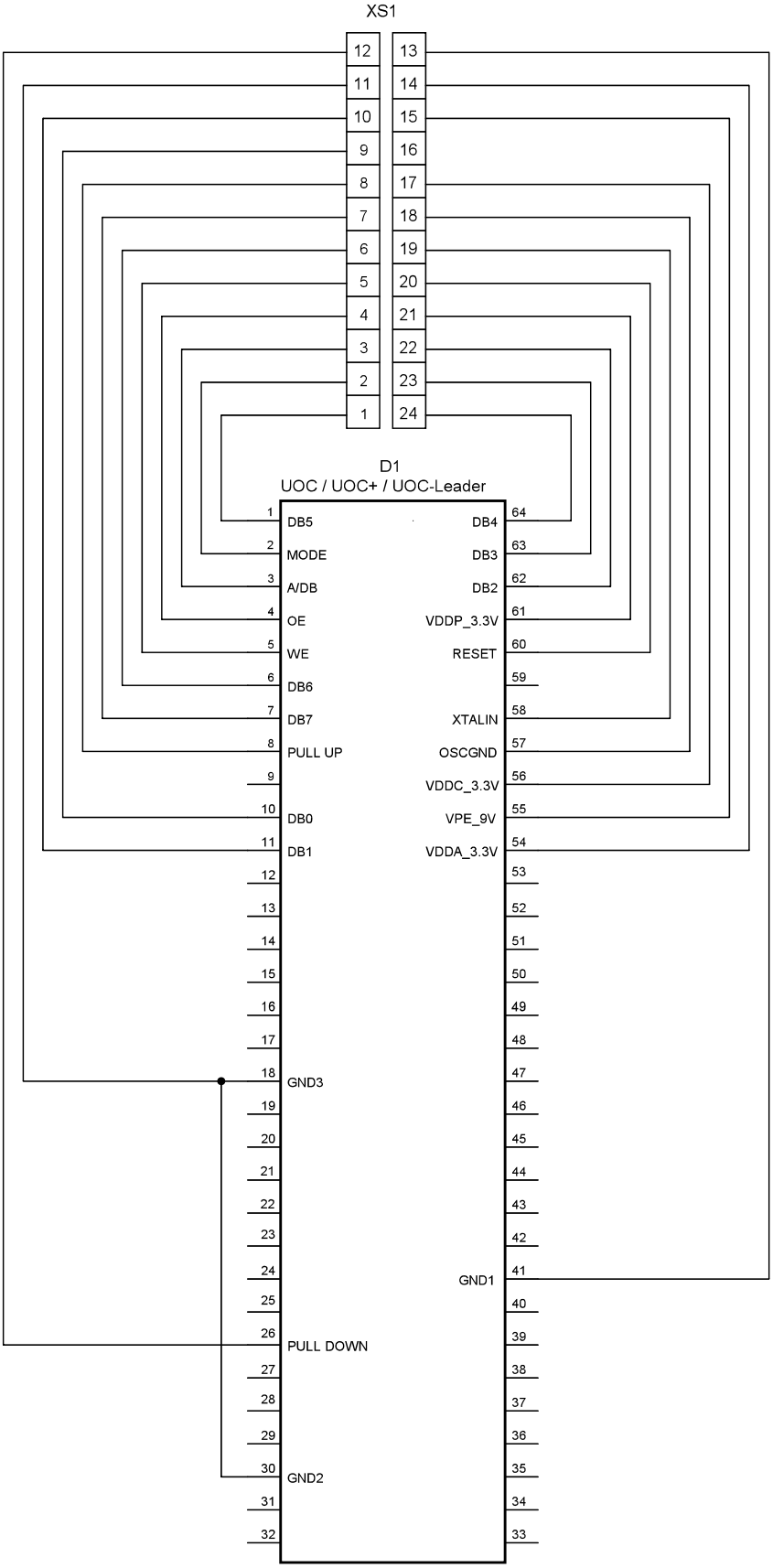
Подключение сокета для программирования **Painter** к разъёму X1



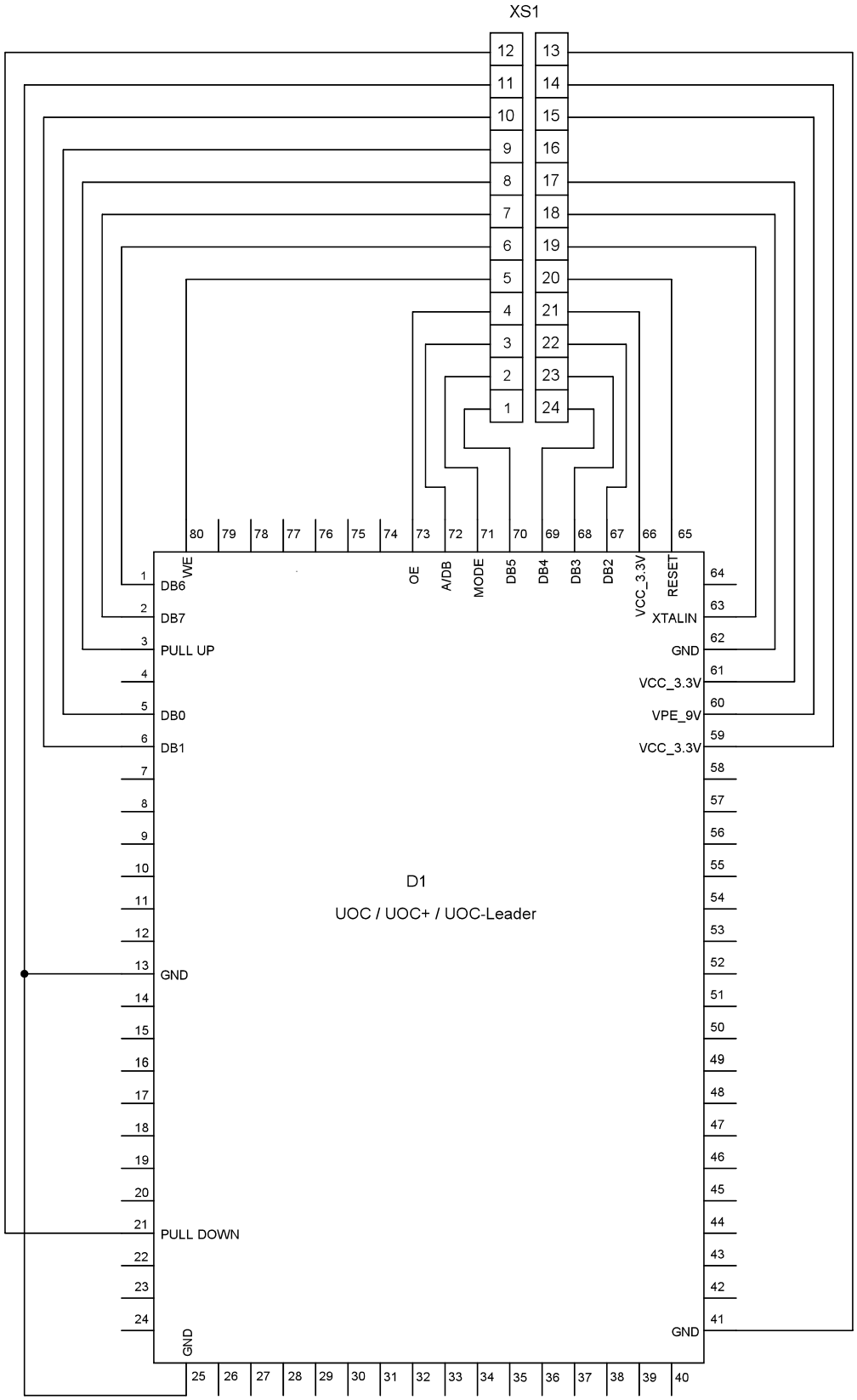
Подключение сокета для программирования **Painter-II** в корпусе QFP-100 к разъёму X1



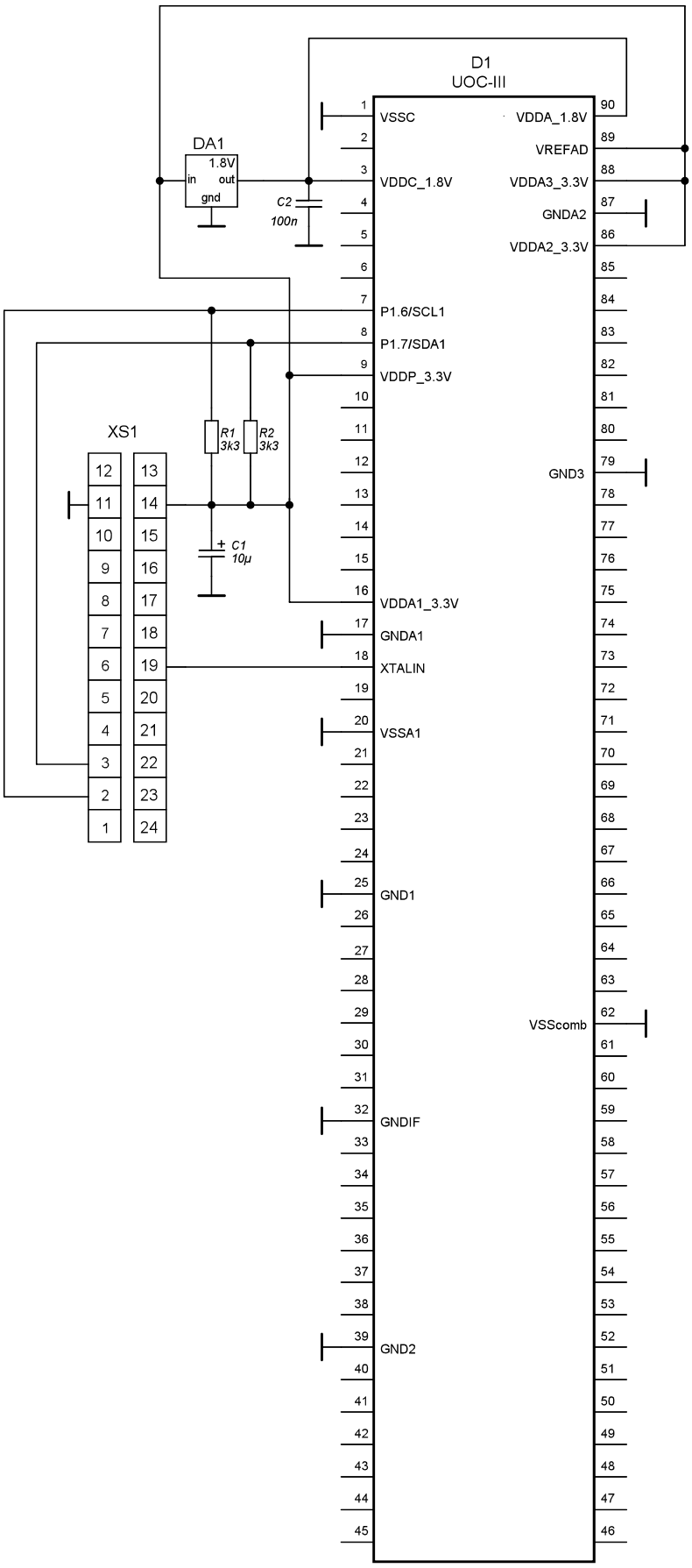
Подключение сокета для программирования ИМС **UOC / UOC+ / UOC-Leader** в корпусе SDIP-64 к разъёму X1



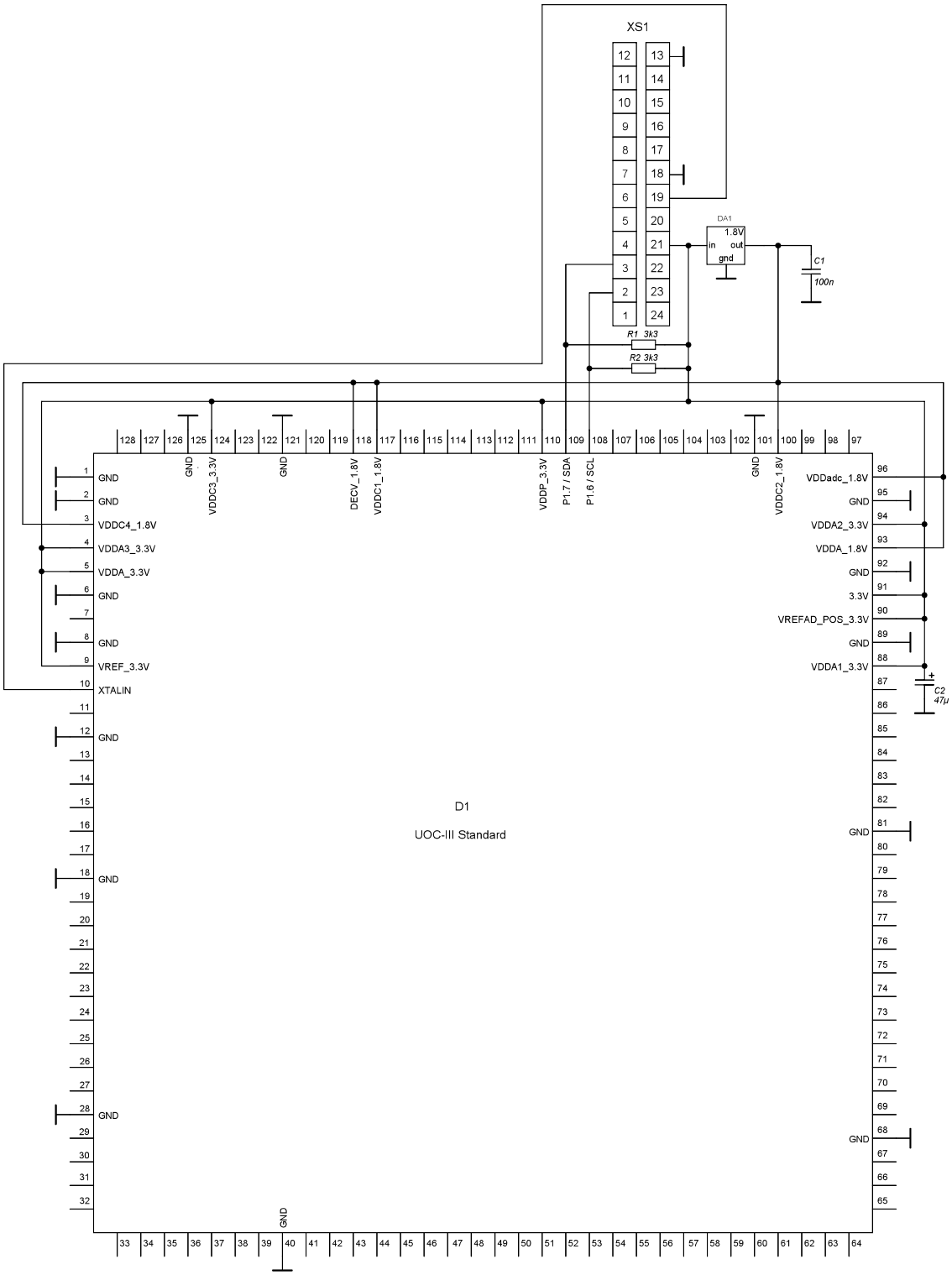
Подключение сокета для программирования **UOC / UOC+ / UOC-Leader** в корпусе QFP-80 к разъёму X1



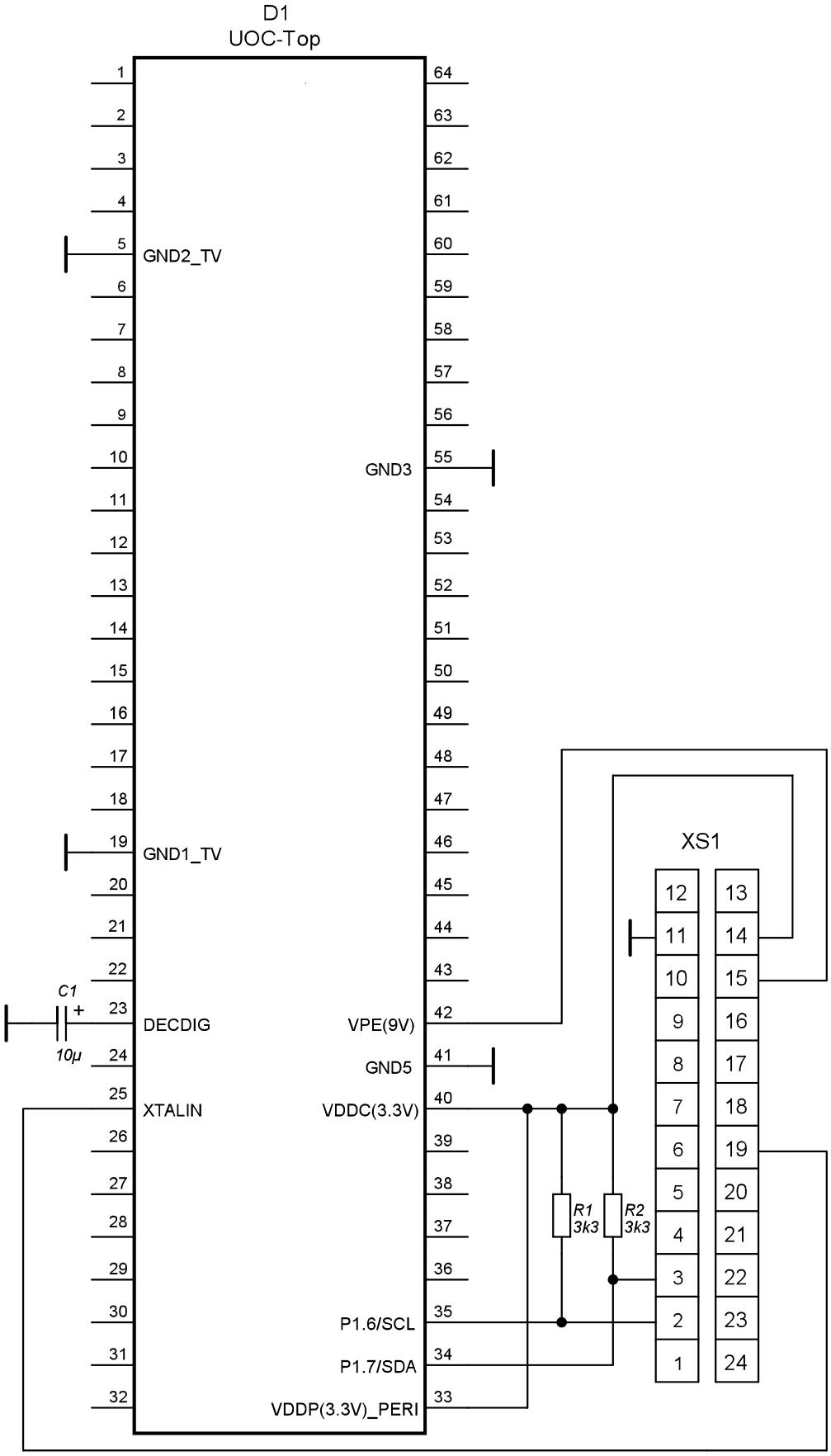
Подключение сокета для программирования **UOC-III** в корпусе QIP-90 к разъёму X1



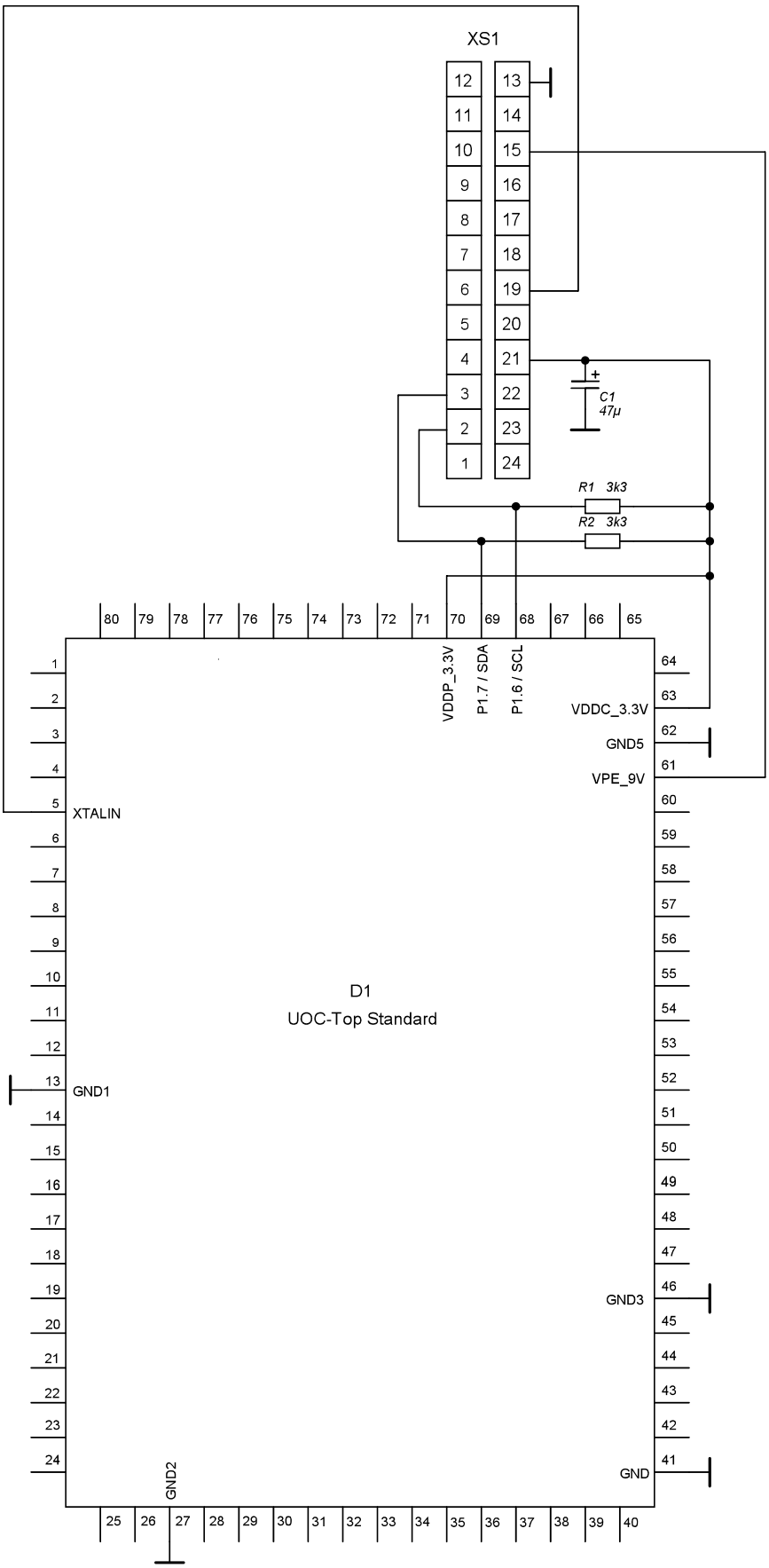
Подключение сокета для программирования **UOC-III** в корпусе QFP-128 к разъёму X1



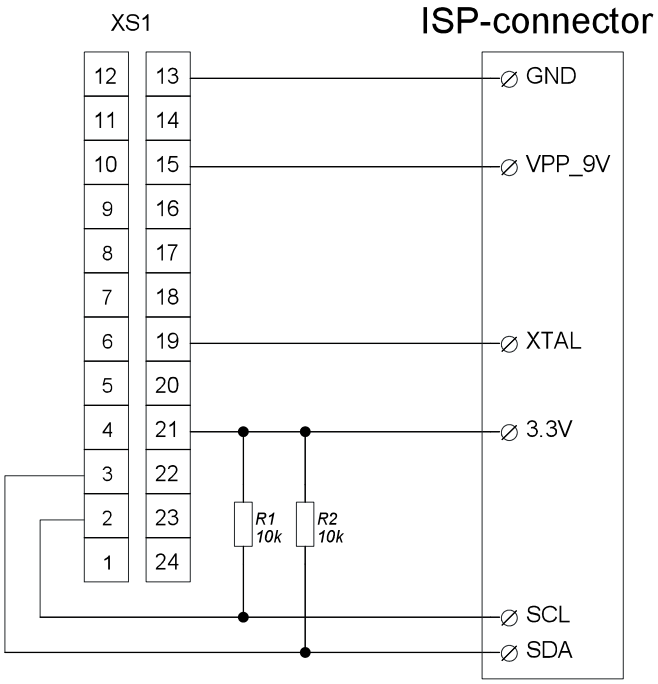
Подключение сокета для программирования **UOC-Top** в корпусе SDIP-64 к разъёму X1



Подключение сокета для программирования **UOC-Top** в корпусе QFP-80 к разъёму X1

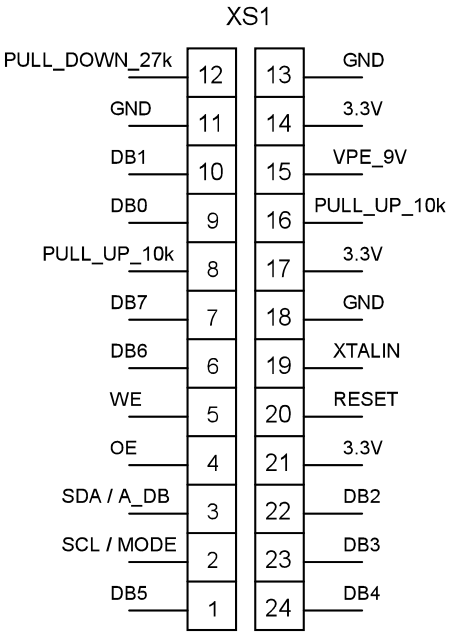


Подключение разъёма для **ISP**-программирования к разъёму X1



R1, R2 можно не устанавливать, т.к. подтягивающие резисторы, предусмотренные спецификацией шины I^2C, устанавливаются на шасси на котором установлена UOC.

Описание выводов разъёма X1 (вилки XS1)



Соответствие выводов программирования UOC в корпусе SDIP-64 и Painter в корпусе SDIP-52

При установленном ZIF-сокете для программирования UOC в корпусе SDIP-64, подключить Painter можно через переходник. Соответствие выводов UOC выводам Painter дано ниже.

Функциональное назначение вывода	**UOC (SDIP-64)**	**Painter (SDIP-52)**
DB5	1	48
MODE	2	49
A/DB	3	50
OE	4	1
WE	5	9
DB6	6	10
DB7	7	11
Weak pull-up, для каждого вывода должен быть установлен отдельный “подтягивающий” резистор 10k	8	2, 3, 4, 5, 6, 7, 8, 12, 14, 15, 16, 17, 18, 21, 30,51,52
DB0	10	19
DB1	11	20
GND	12, 18, 30, 41, 57	13, 25, 36, 37, 38, 40
Weak pull-down, “опускающий” резистор 27k	26	26
VPE, +9V	55	28
XTALIN	58	41
RESET	60	43
Vcc, +3.3V	54, 56, 61	31, 39, 44
DB2	62	45
DB3	63	46
DB4	64	47

Соответствие выводов программирования UOC/UOC+ в корпусе SDIP-64 и корпусе QFP-80

При установленном ZIF-сокете для программирования UOC в корпусе SDIP-64, подключить UOC в корпусе QFP-80 можно через переходник. Соответствие выводов корпуса SDIP-64 выводам корпуса QFP-80 дано ниже.

Функциональное назначение вывода	**UOC-PS (SDIP-64)**	**UOC-H (QFP-80)**
DB5	1	70
MODE	2	71
A/DB	3	72
OE	4	73
WE	5	80
DB6	6	1
DB7	7	2
Weak pull-up, “подтягивающий” резистор 10k	8	3
DB0	10	5
DB1	11	6
GND	12, 18, 30, 41, 57	13, 25, 41,62
Weak pull-down, “опускающий” резистор 27k	26	21
VPE, +9V	55	60
XTALIN	58	63
RESET	60	65
Vcc, +3.3V	54, 56, 61	59, 61, 66
DB2	62	67
DB3	63	68
DB4	64	69

Программирование микросхем с помощью **UOCLAB** версии **3.5**

Краткое описание функций

Программа ***uoclab*** была разработана специально для сервис-центров, занимающихся ремонтом телевизоров. Программатор позволяет считать прошивку из UOC, автоматически модифицировать её (пропатчить) и записать её в чистую микросхему.

Uoclab реализует стандартный набор функций фирменных программаторов Philips и NXP: проверка ИМС на чистоту (**BLANK**), запись (**WRITE**) и верификация (**VERIFY**) для микросхем Painter, Painter-II, UOC, UOC+, UOC-Leader, UOC-III, UOC-Top. Для UOC-III также поддерживается функция стирания микросхемы (**ERASE**).

Помимо стандартных для фирменных программаторов функций, программатор реализует ряд дополнительных функций:

1. чтение (**READ**);
2. идентификация типа микросхемы (**SIGN**);
3. анализ сигнатур;
4. поиск совместимых микросхем;
5. автоматическая модификация прошивок UOC и UOC-Leader (**Autopatch**) для того, чтобы прошивку из UOC одного типа можно было прошить в UOC другого типа без потери функциональности;
6. анализ файлов прошивок (**FILE INFO**).

Следует также помнить, что какими бы “умными” не были функции программатора, но они не могут предусмотреть всё многообразие программных особенностей семейств UOC. В особо сложных случаях код придётся исправлять вручную.

ВНИМАНИЕ! Программатор во время работы может потреблять значительный ток, поэтому используйте только качественный USB кабель! **Падение напряжения на кабеле не должно превышать 0.3В!**

ВНИМАНИЕ! Если Вы подключаете программатор к компьютеру первый раз, то сразу после подключения операционная система может попросить указать, где находятся драйвера для программатора. На вопрос системы следует указать каталог с драйверами на поставляемом вместе с программатором диске или указать автоматическую установку. После успешной установки драйверов можно запускать ***uoclab***.

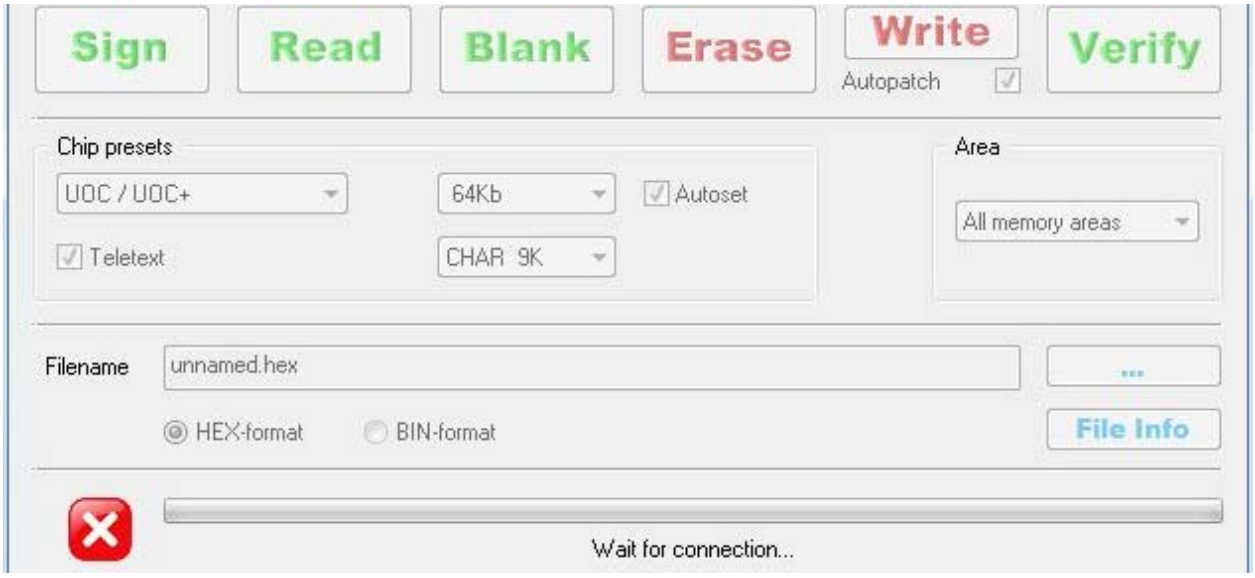
ВНИМАНИЕ! Если предполагается работа с файлами, то достаточно просто подключить программатор к компьютеру без программируемой микросхемы.

ВНИМАНИЕ! Микросхемы следует вставлять в сокет программатора до начала операции и вынимать из сокета программатора после уведомления о завершении операции, только при погасшем красном светодиоде (питание с Painter/UOC снято). В противном случае, есть опасность повреждения микросхемы.

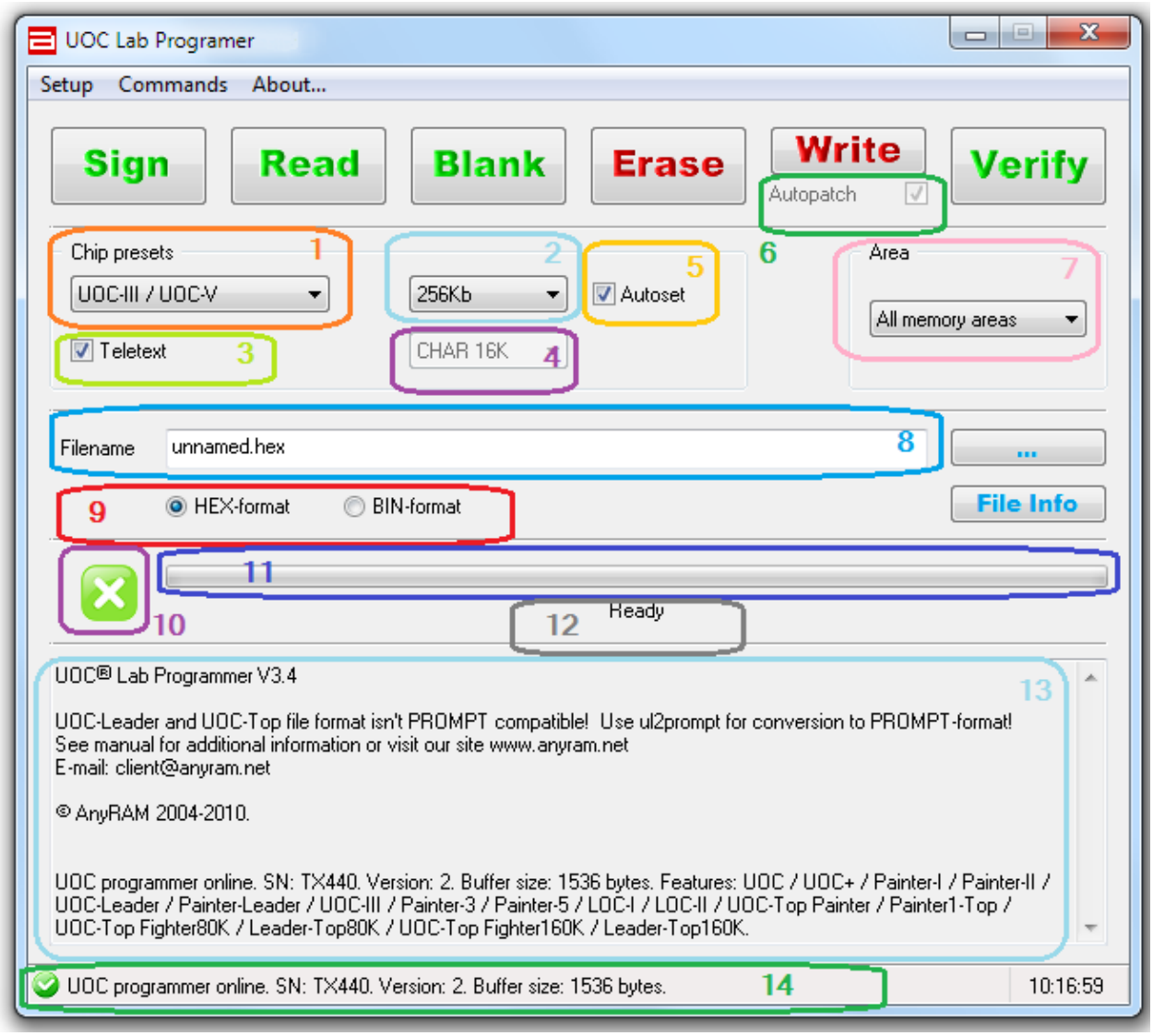
ВНИМАНИЕ! Все функции ***uoclab*** доступны только при <u>подключенном</u> программаторе!

Описание пользовательского интерфейса

При запуске ***uoclab*** появляется окно диалога. Если программатор не подключен, то ***uoclab*** ожидает подключения: ни одна функция не доступна.



Окно *uoclab* при отключенном программаторе (функции недоступны)



Окно *uoclab* при подключенном программаторе

В заголовке окна указывается текущая версия ***uoclab***(V3.5).

Назначение элементов в окне диалога.

1. Поле выбора типа микросхемы; возможны следующие варианты:
   UOC / UOC+,
   Painter-I,
   Painter-II,
   UOC-Leader,
   UOC-III / UOC-V (зависит от модели программатора),
   UOC-Top Painter (зависит от модели программатора),
   UOC-Top Fighter 80K (зависит от модели программатора),
   UOC-Top Fighter 160K (зависит от модели программатора).
2. Поле выбора объёма кодовой памяти (**CODE**); поле выбора неактивно, если в семействе нет микросхем с различным объёмом памяти.
3. Флаг присутствия телетекста в микросхеме. Включает/выключает обработку дополнительных областей памяти в микросхеме.
4. Поле выбора объёма памяти символов (**CHAR**); поле выбора неактивно, если в семействе нет микросхем с различным объёмом памяти.
5. Флаг активности функции автоматической установки параметров программатора по сигнатуре чипа. Определение типа микросхемы, объёма памяти **CODE** и **CHAR**, опции **Teletext**, производится по внутренней базе данных ***uoclab***. Если опознавание невозможно, то текущие настройки программатора не изменяются. Автоматическое определение установок производится перед началом любой операции с микросхемой.
6. Флаг активности функции автопатча при записи UOC. Если автоматическая модификация прошивки невозможна, то производится только изменение сигнатуры.
7. Поле выбора области программирования; поле выбора неактивно, если в микросхеме только одна область памяти.
8. Поле ввода имени файла; имя файла можно ввести вручную, а можно заполнить автоматически, используя кнопку “**…**”.
9. Выбор формата файла: Intel-HEX32 или бинарный формат. HEX-формат – предпочтительнее, т.к. позволяет сохранить только необходимые данные.
10. Значок, отражающий текущее состояние:

    - программатор не подключен;

    - программатор подключен, ожидание команды;

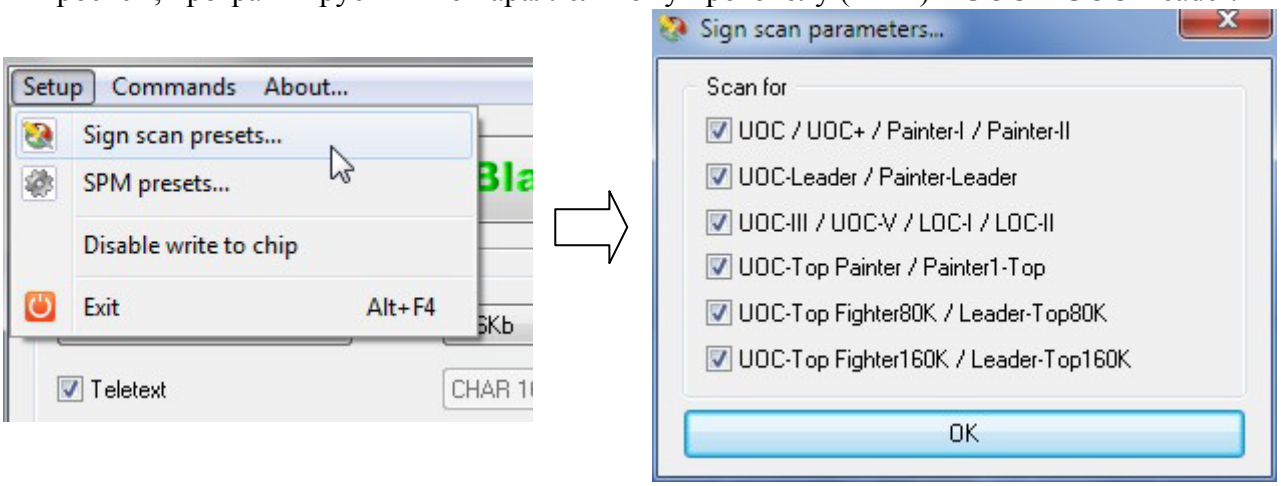
    - идёт выполнение команды, программатор занят; нажатие на эту иконку прерывает текущую операцию.
11. Полоса-индикатор, отражает ход выполнения операции.
12. Строка, отражающая текущее состояние ***uoclab***.
13. Область вывода информации и сообщений (лог).
14. Статус текущего подключения, информация о подключенном программаторе.

Кнопки “**SIGN”**, “**READ**”, “**BLANK**”, “**ERASE**”, “**WRITE**”, “**VERIFY**”, заставляют программатор выполнить соответствующую команду. По нажатию кнопки “**FILE INFO**”, ***uoclab*** обрабатывает файл прошивки, имя которого задано в поле **8** – “**Filename**”.

Автоопределение типа микросхем, опция “Autoset”

Программатор позволяет производить автоматическое определение типа подключенной микросхемы. Определение типа подключенной микросхемы производится поиском протокола обмена с последующим чтением сигнатуры, которая сравнивается с записями во встроенной в ***uoclab*** базе данных. Обработка считанных данных ведётся в 2 этапа: вначале производится поиск точного совпадения сигнатуры с записью в базе данных, если совпадение не найдено, то производится поиск совместимых по параметрам микросхем.

В некоторых случаях требуется отменить поиск определённых типов микросхем: например, при работе программатора в **ISP**-режиме (**I**n-**S**ystem-**P**rogramming mode – режим внутрисхемного программирования) рекомендуется отключить режим поиска микросхем, программируемых по параллельному протоколу (**PPM**) – UOC и UOC-Leader.

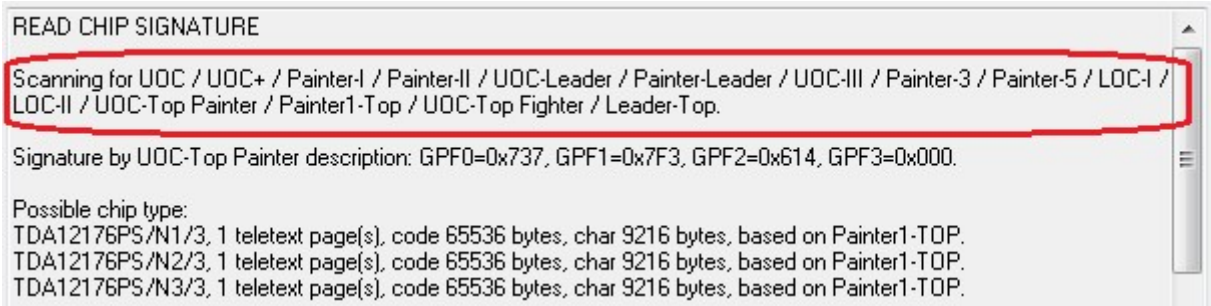


Меню установок определителя типа подключенной микросхемы

Параметры определителя типа подключенной микросхемы задаются в меню “**Setup→Sign scan presets**”. Выключение опознавания неиспользуемых типов микросхем ускоряет работу программатора, а также позволяет избежать посылки “лишних” сигналов при программировании в **ISP**-режиме. Также установки этого меню могут помочь в случае ошибочного опознавания сходных типов микросхем, например, UOC и UOC-Leader.

Например, если программатор подключается для программирования только в **ISP**-режиме, то можно отключить опознавание микросхем, программируемых в **PPM**, т.е. снять выбор с опций “**UOC/…**” и “**UOC-Leader/…**”.

Автоматическое определение типа подключенной микросхемы происходит при активной опции “**Autoset**” в начале любой операции с микросхемой: **SIGN”**, “**READ**”, “**BLANK**”, “**ERASE**”, “**WRITE**”, “**VERIFY**”.



Красным выделен лог начала работы автоопределителя типа микросхемы

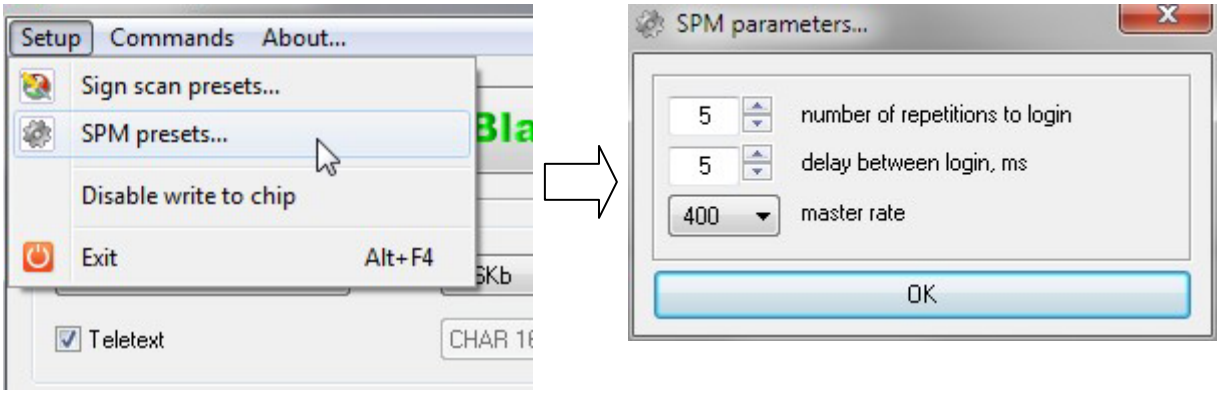
После определения типа микросхемы, установка объёма памяти **CODE** и **CHAR**, опции **Teletext**, производится по внутренней базе данных ***uoclab***. Если опознавание невозможно, то текущие настройки программатора не изменяются.

Интерфейс программирования

Для программирования микросхем программатор использует два типа интерфейсов: параллельный и последовательный. Параллельный интерфейс может быть использован для программирования микросхем Painter/UOC всех типов, однако, в лабораторном программаторе параллельный интерфейс используется только для тех микросхем, которые не поддерживают последовательного интерфейса программирования. Промышленная версия программатора использует параллельный интерфейс для программирования микросхем всех типов.

Параллельный режим программирования **PPM** - ***P**arallel **P**rogram **M**ode* - используется при программировании Painter, Painter-II, UOC, UOC+, UOC-Leader. Программирование в этом режиме в составе шасси невозможно.

Последовательный режим программирования **SPM** – ***S**erial **P**rogram **M**ode* используется при программировании LOC-I, LOC-II, UOC-III, UOC-Top Painter, UOC-Top Fighter. В этом режиме возможно программирование микросхем в составе шасси телевизора. Для программирования используется протокол I^2C со значительными отступлениями от стандарта. Параметры **SPM** задаются в меню "**Setup→SPM presets**".



Меню параметров SPM режима программирования

Вход в **SPM** происходит в <u>рабочем режиме микросхемы</u>, т.е. в момент входа в режим программирования микросхема уже отработала какое-то время в штатном режиме и, вполне возможно, уже ведёт обмен по шине I^2C. Для входа в режим программирования программатор должен подать последовательность **SPM** в момент паузы обмена данными. Количество попыток подключения задаётся в поле "**number of repetitions to login**", время между попытками – в поле "**delay between login**". На скриншоте указано 5 попыток входа со временем 5млс между попытками.

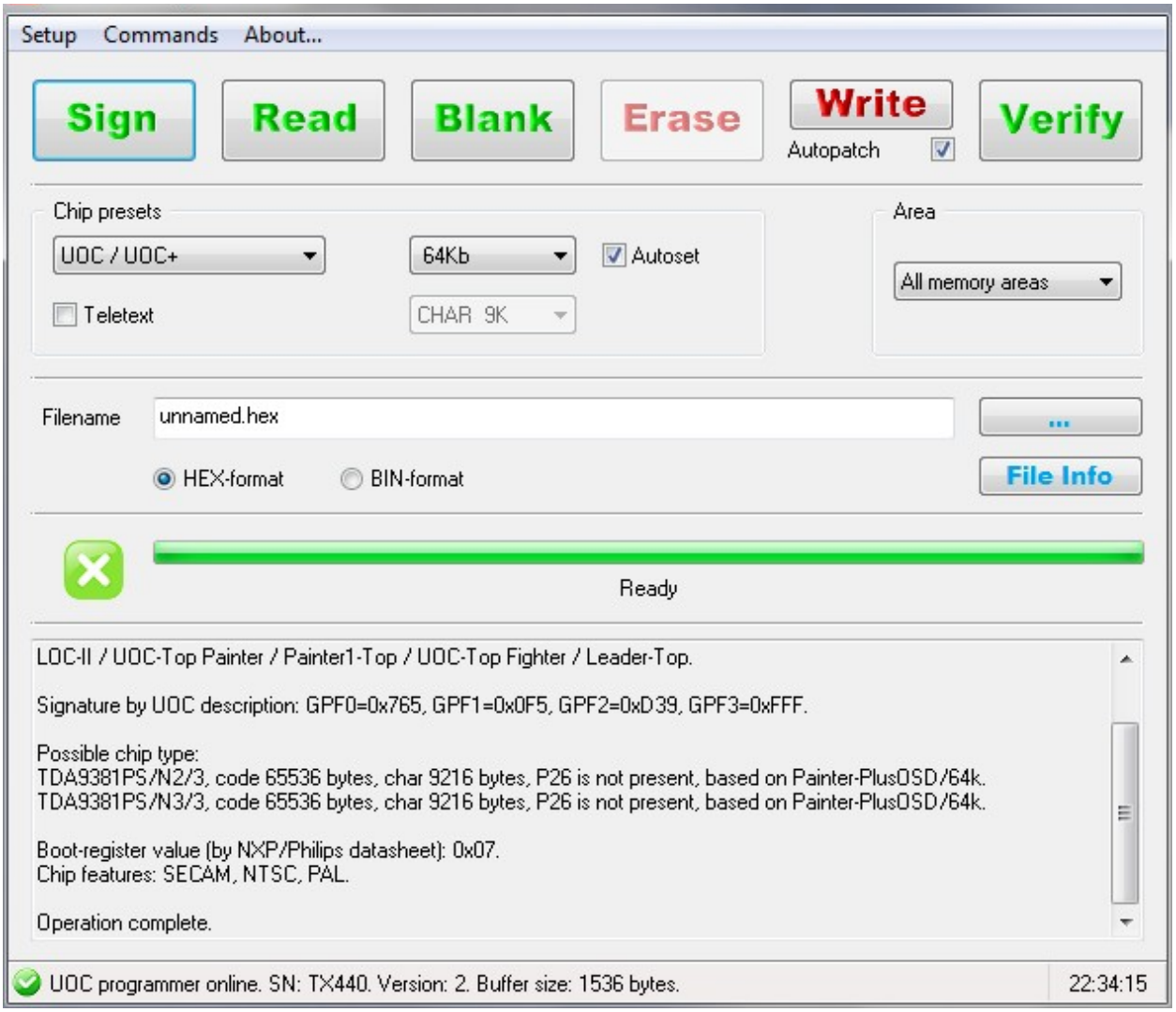
Поскольку импеданс шины I^2C на телевизионном шасси не является константой, может потребоваться изменение максимальной скорости обмена по шине I^2C в **SPM**-режиме. Изменение максимальной скорости производится в поле "**master rate**". Минимальная скорость обмена выбирается программатором автоматически.

Если программатор не поддерживает **SPM**-режим, то меню "**SPM presets**" будет неактивно.

Чтение сигнатуры (SIGN)

Сигнатура чипа записывается на заводе-изготовителе. Наиболее важной частью сигнатуры обычно являются части GPF0 и GPF1: это флаги, которые считываются через специальные регистры микроконтроллера и используются для инициализации UOC, т.е. сигнатура влияет на приём и отображение телевизионного сигнала. Для микросхем семейства Painter, сигнатура используется для установки режима работы телетекста, т.е. не оказывает такого решающего влияния, как в UOC.

Чтение сигнатуры чипа позволяет определить тип чипа даже при отсутствии маркировки. Однако автоматическое определение сигнатуры не всегда возможно. Основная причина – стирание номера микросхемы заводом изготовителем (тип микросхемы хранится в GPF2 и GPF3). В этом случае ***uoclab*** делает попытку опознавания чипа не по номеру, а по флагам GPF0 и GPF1.



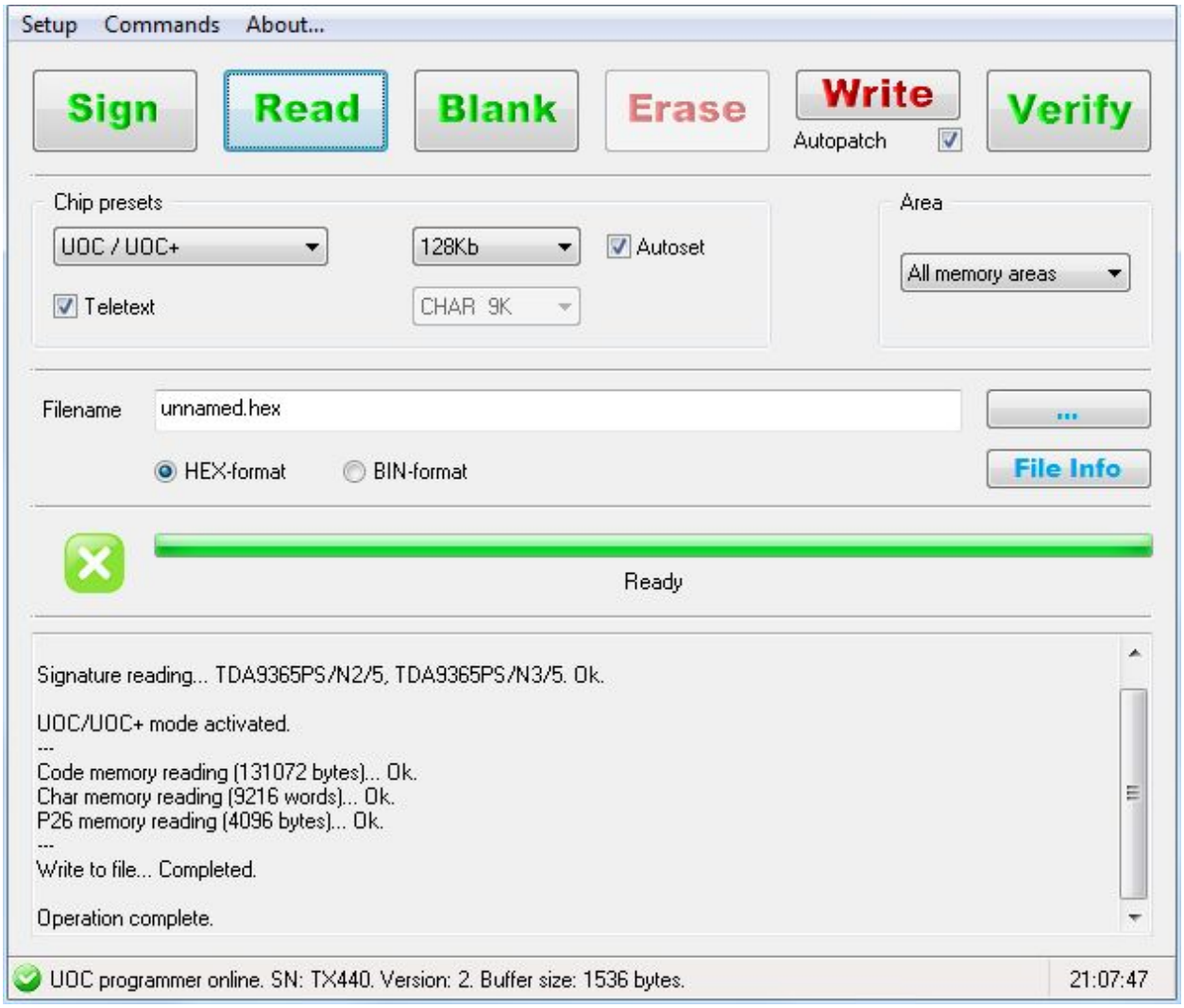
Скриншот после нажатия кнопки “*SIGN*”

В случае успешного чтения сигнатуры, будет выведена сигнатура чипа GPF0, GPF1, GPF2, GPF3, возможные типы микросхем, которые соответствуют считанной сигнатуре, а также дополнительная информация, характерная для выбранного типа микросхем (на скриншоте это значение boot-регистра и особенности, характерные для данного типа UOC).

Бывает так, что производитель телевизоров специально затирает сигнатуру и определение чипа по сигнатуре невозможно. В этом случае для последующих операций следует деактивировать опцию “**Autoset**” и установить тип и параметры параметры микросхемы вручную.

Чтение Painter/UOC в файл (READ)

Перед чтением следует правильно установить тип и параметры микросхемы, если опция “**Autoset**” неактивна, а также указать имя файла и формат хранения данных (**hex**-формат или **bin**-формат). Ввести имя файла можно вручную или с помощью кнопки “**…**”.



Скриншот после нажатия кнопки “*READ*”

Во время чтения микросхемы в окне статуса и лога отображается информация о текущей операции, полоса-индикатор (прогресс-бар) отражает ход выполнения операции. Тип упаковки и размер сохраняемого файла образа будет выбран в зависимости от выбранного типа микросхемы.

При чтении микросхемы учитывается поле выбора области программирования: для чтения полного образа микросхемы в окне должно быть установлено “**All memory areas**”. В противном случае в файл образа микросхемы попадёт только область указанная селектором.

Прервать процесс можно нажатием на иконку

во время операции.

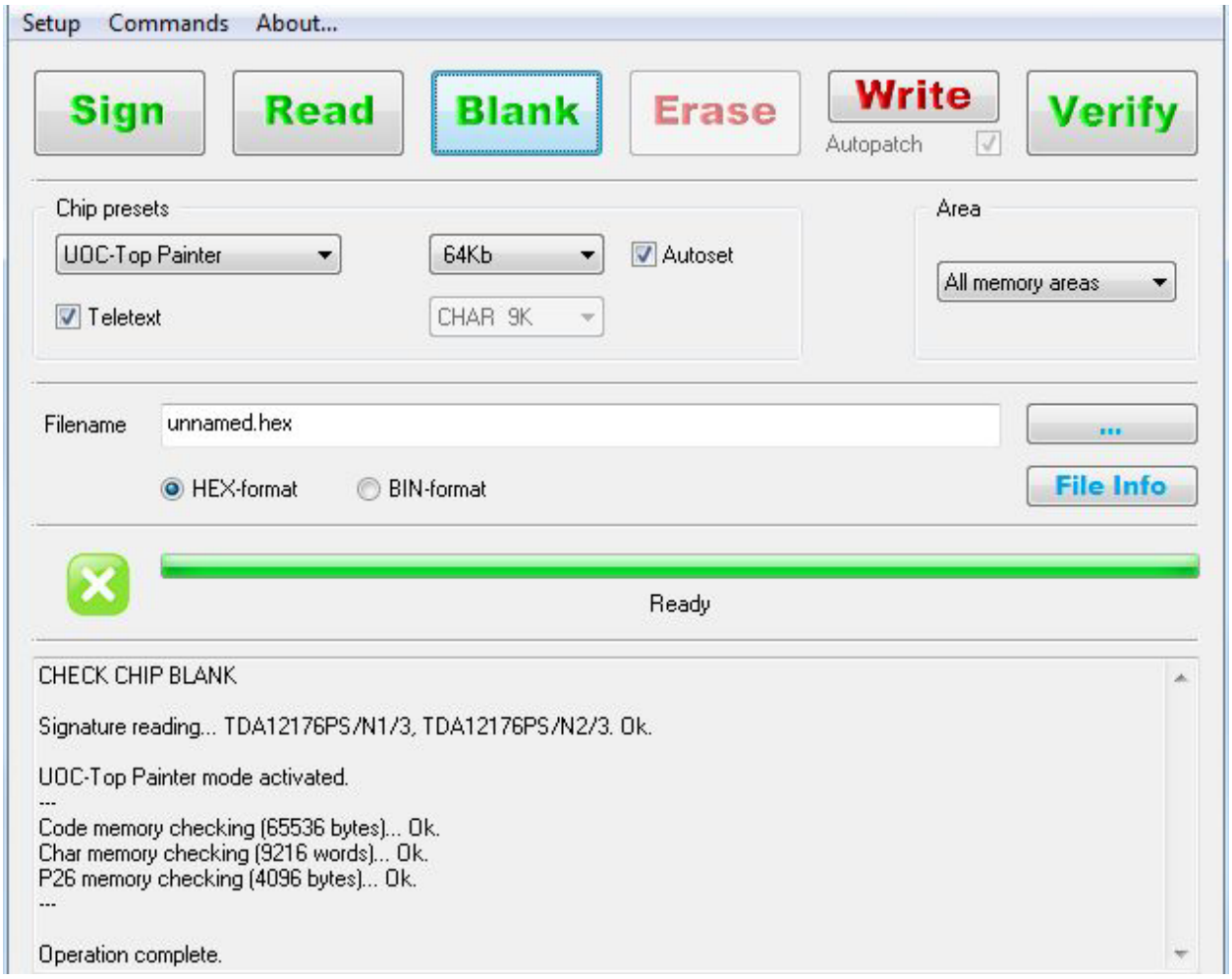
После чтения рекомендуется воспользоваться командой “**File Info**”, чтобы получить дополнительную информацию о считанной прошивке.

Сохранять файлы образов рекомендуется в **hex**-формате: этот формат файла оптимизирован для хранения образов с фрагментированием данных, а также является совместимым форматом для основного инструмента компоновки файлов Philips/NXP – PROMPT.

Проверка Painter/UOC на чистоту (BLANK)

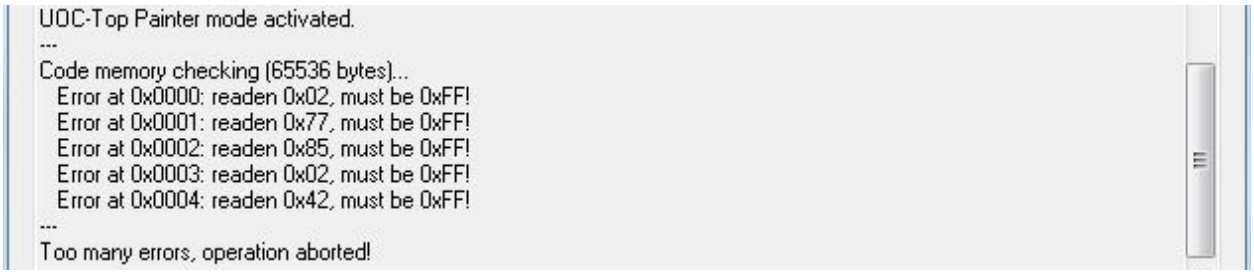
Проверять микросхему на чистоту рекомендуется с установленной опцией “**Autoset**”. Если опция выключена, то перед проверкой на чистоту следует правильно установить тип и параметры микросхемы, иначе операция может быть проведена с ошибками. Если требуется проверить только одну из областей памяти, то следует установить поле выбора области программирования в соответствующее значение.

Если проверка микросхемы на чистоту успешно завершена, то на экране появится следующее сообщение.



Скриншот успешного завершения проверки на чистоту “*BLANK*”

Если микросхема не чиста или установлен неверный тип микросхемы, то будет выведено сообщение об ошибках с указанием характера ошибки.

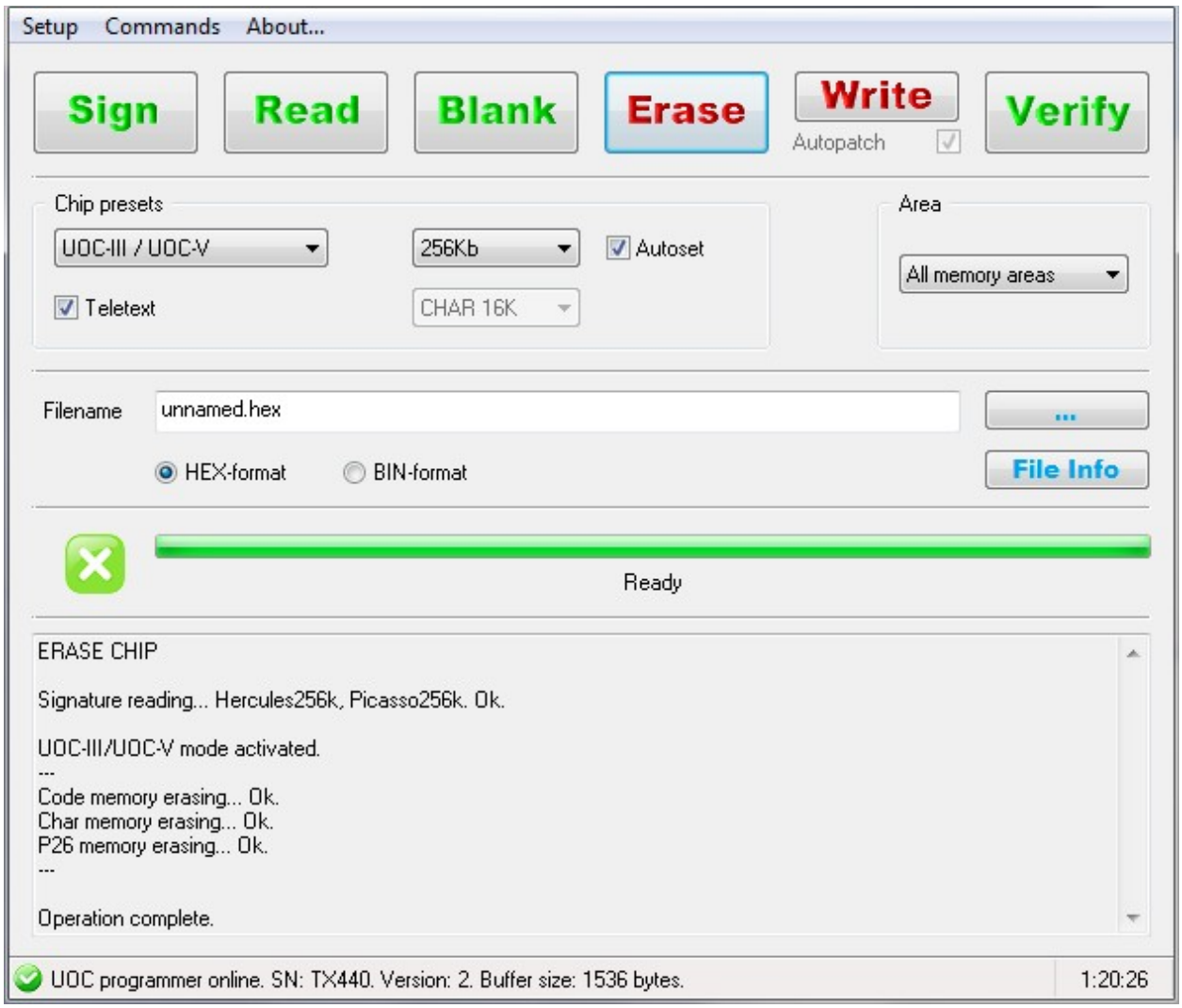


Скриншот сбоя во время проверки на чистоту “*BLANK*”

Стирание Painter/UOC (ERASE)

Стирать микросхему на чистоту рекомендуется с установленной опцией “**Autoset**”. Если опция “**Autoset**” выключена, то перед стиранием следует правильно установить тип и параметры микросхемы, иначе операция может быть проведена с ошибками. Если требуется проверить только одну из областей памяти, то следует установить поле выбора области программирования в соответствующее значение.

Когда стирание микросхемы будет завершено, на экране появится сообщение.



Скриншот завершения стирания микросхемы “*ERASE*”

После стирания, микросхему можно проверить на чистоту командой “**BLANK**”. Стирание возможно только для микросхем с **Flash**-памятью – UOC-III, UOC-V. Стирание микросхем с **OTP-**памятью возможно только на специальном оборудовании.

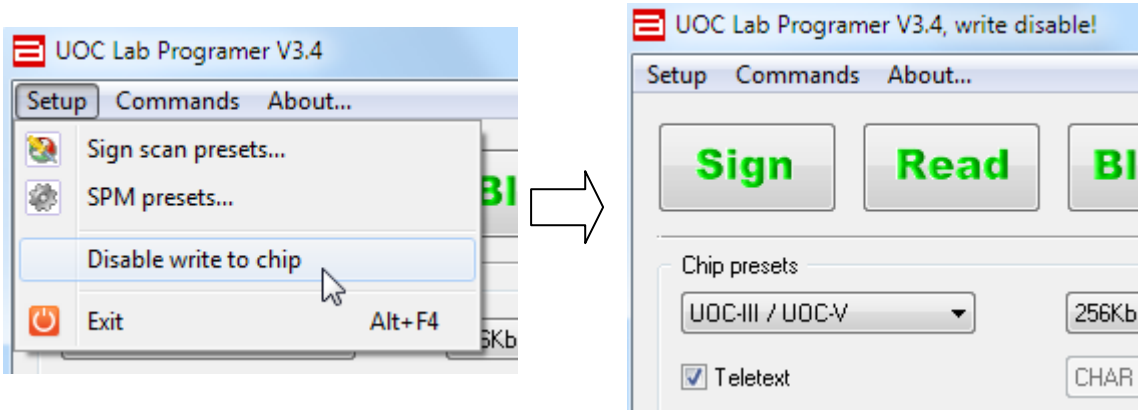
Стирание памяти производится страницами по **32К**. При стирании микросхемы также стирается и область сигнатуры, поэтому при последующей записи данных в такую микросхему следует обращать внимание на совпадение сигнатуры прошивки и аппаратные возможности микросхемы.

Программирование Painter/UOC (WRITE)

Перед записью следует правильно установить тип и параметры микросхемы, если опция “**Autoset**” неактивна, а также указать имя и формат файла (**hex**-формат или **bin**-формат). Ввести имя файла можно вручную или с помощью кнопки “**…**”.

Опция “**Autopatch**” позволяет автоматически изменить код прошивки если программа считана из микросхемы одного типа, а прошивается в микросхему другого типа (см. приложение). Если прошивка заранее подготовлена для прошиваемой микросхемы, то опцию “**Autopatch**” следует выключить. В текущей версии ***uoclab*** поддерживается автопатч UOC, UOC+, UOC-Leader. Следует отметить, что ***программирование UOC требует наличия хотя бы минимальных знаний об устройстве этих микросхем***.

Для тестовых целей была введена возможность имитации процесса записи данных в Painter/UOC: в этом режиме подготовка данных и образа ведётся также как и при реальной записи, но дамп данных вместо микросхемы будет записан в файл. Для запрещения записи в микросхему следует активировать режим “**Setup→Disable write to chip**”.



Скриншоты заголовка до и после активации режима “*Disable write to chip*”

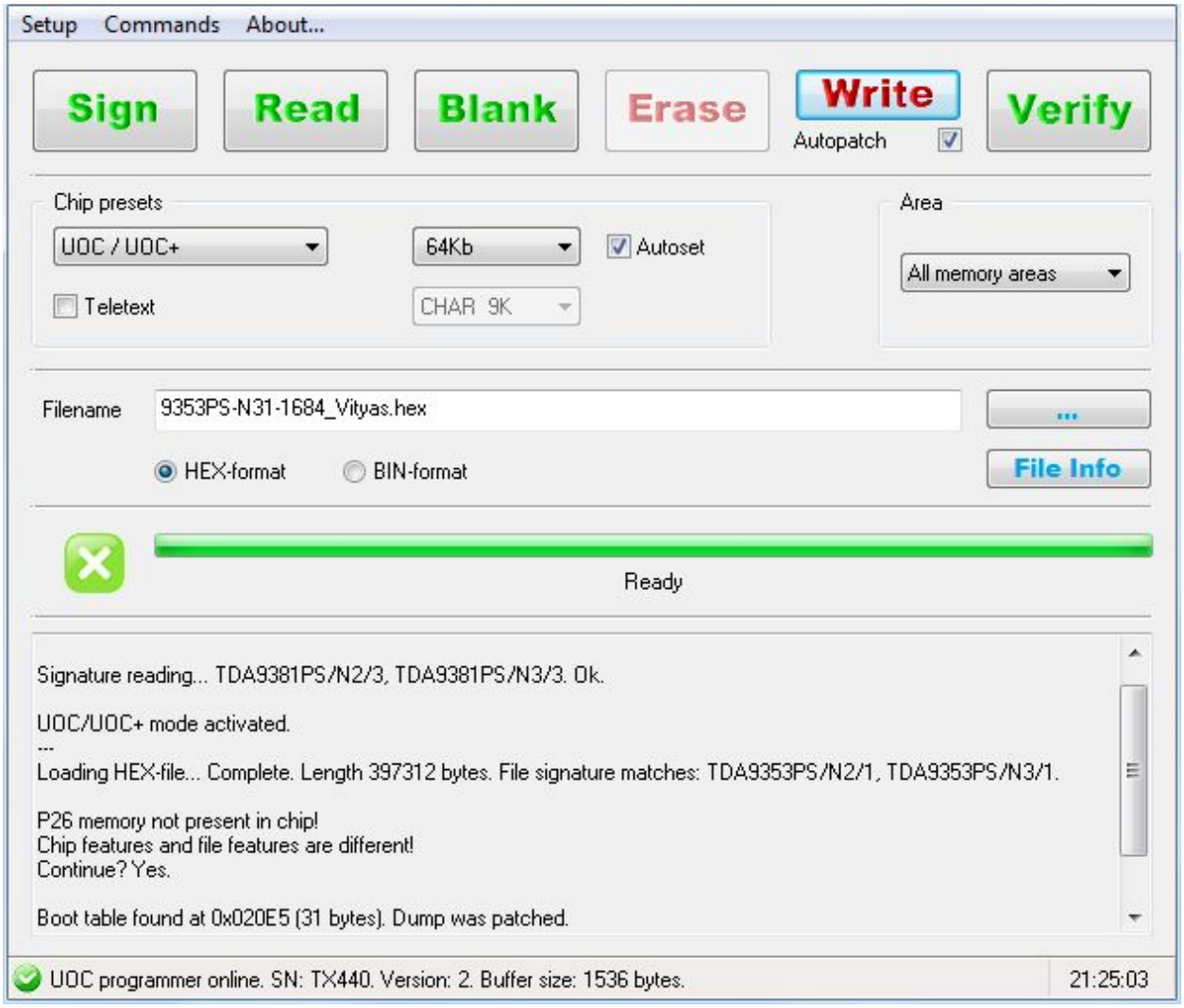
Во время записи микросхемы в окне статуса и лога отображается информация о текущей операции, полоса-индикатор (прогресс-бар) отражает ход выполнения операции. Перед программированием производится обязательная проверка соответствия файла образа и программируемой микросхемы. В случае несовпадения типов, пользователю будут заданы вопросы, уточняющие процесс программирования микросхемы.

При записи микросхемы учитывается поле выбора области программирования: для записи полного образа микросхемы в окне должно быть установлено “**All memory areas**”. В противном случае в микросхеме будет записана только область указанная селектором.

Прервать процесс можно нажатием на иконку

во время операции, однако, следует помнить, что прерывание процесса записи может привести к непоправимой порче микросхемы.

После успешного завершения записи автоматически начинается верификация микросхемы: побайтная верификация необходима для проверки корректности записанных данных, проверки с помощью MISR-регистров недостаточно!



Скриншот процесса записи микросхемы “*WRITE*”

На скриншоте показан процесс записи файла прошивки считанного из микросхемы TDA9353PS/N3/1 с телетекстом в микросхему TDA9381PS/N3/3 без телетекста. Видно, что пользователь намеренно разрешает такую запись: при включении телетекста, картинка на экране будет, как если бы телетекст отсутствовал в эфире, даже при наличии сигнала телетекста на входе – в TDA9381 отсутствует слайсер. Также видно, что активная функция “**Autopatch**” исправляет прошивку для того, чтобы в микросхеме TDA9381 включилась функция East/West-коррекции, как в TDA9353: об успешной подготовке прошивки для TDA9381 говорит надпись “**Boot table found at 0x020E5 (31 bytes). Dump was patched**”.

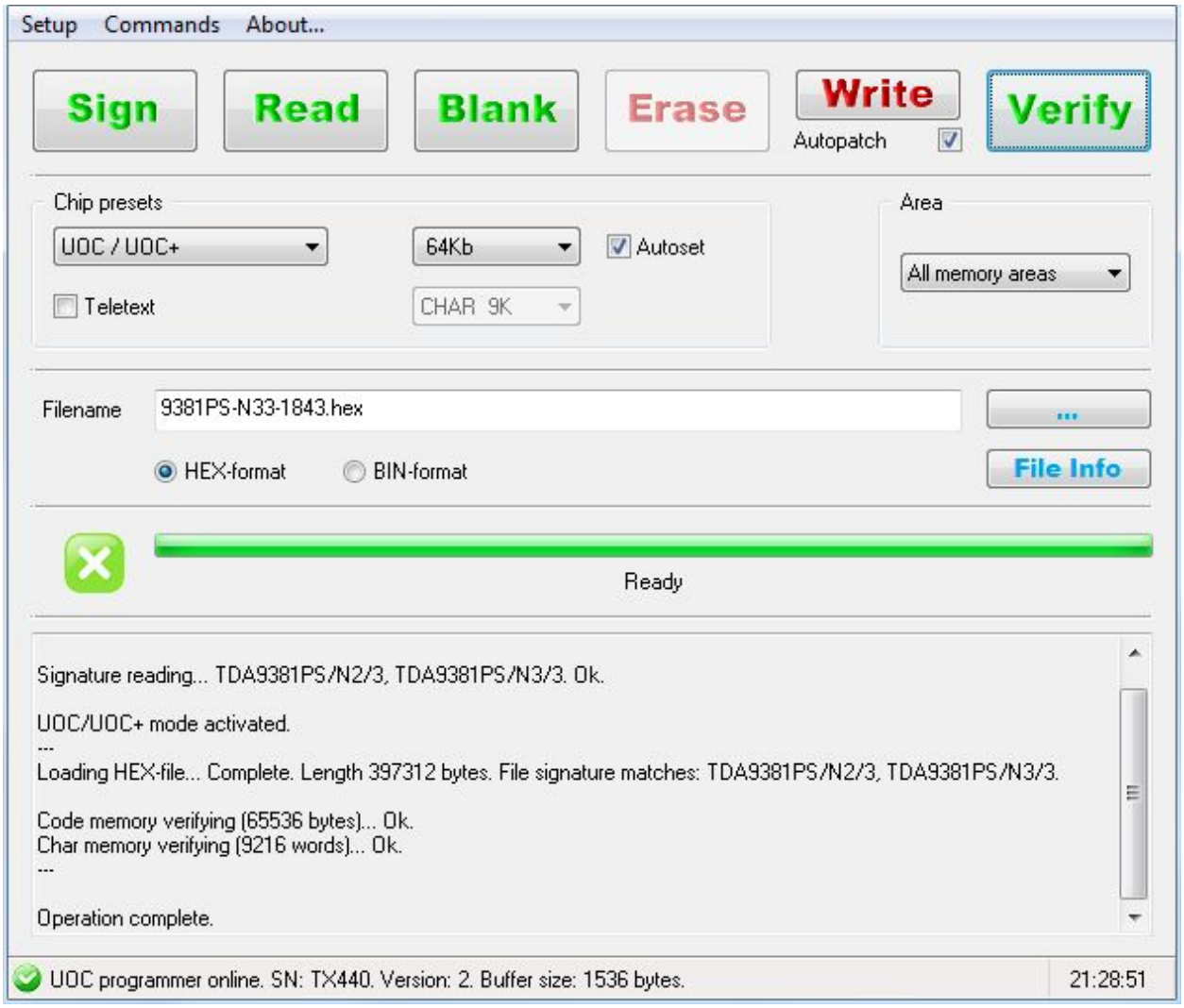
В текущей версии программы автоматический патч прошивок поддерживается для семейств UOC, UOC+, UOC-Leader. Алгоритм наложения патча описан в приложении.

Верификация Painter/UOC (VERIFY)

Верификация обычно проводится сразу после чтения микросхемы для подтверждения успешности записи. Однако, также верификация может быть использована для установления подлинности прошивки, исправности знакогенератора и т.п..

Перед верификацией следует правильно установить тип и параметры микросхемы, если опция “**Autoset**” неактивна, а также указать имя и формат файла (**hex**-формат или **bin**-формат). Ввести имя файла можно вручную или с помощью кнопки “**…**”.

При установленной опции “**Autopatch**” для проверки используется не сам файл прошивки, а модифицированный по методике автоматического патча образ (см. команду ***WRITE***). Это позволяет верифицировать микросхемы, прошивка которых была модифицирована в процессе записи, и образ прошивки не сохранялся.



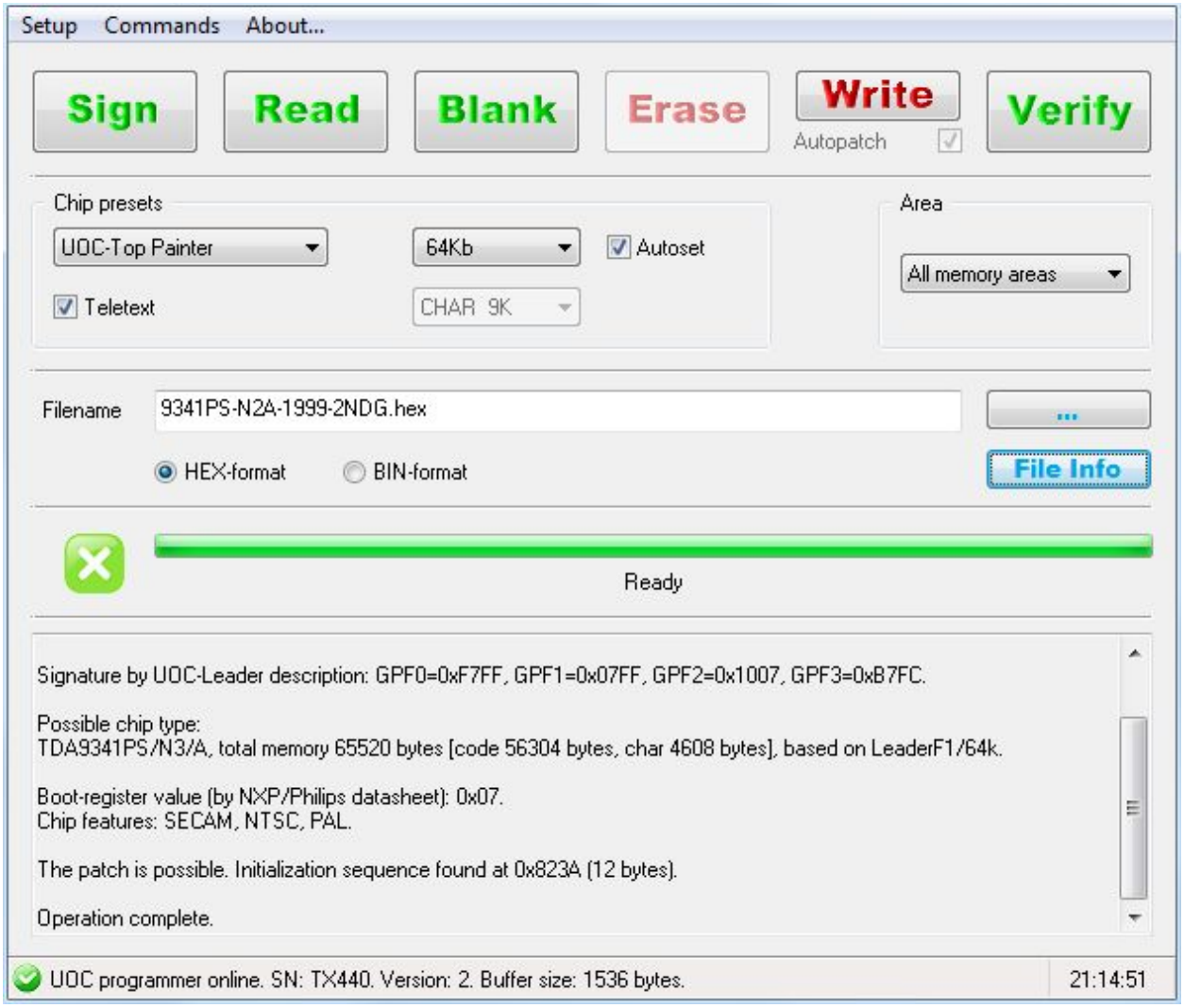
Скриншот верификации микросхемы “*Verify*”

Если в процессе верификации произойдёт сбой или будет установлен неверный тип микросхемы, то в лог будет выведено сообщение об ошибках с указанием их характера и причины.

Получение информации о файле прошивки (FILE INFO)

Для получения информации о прошивке, следует ввести имя файла, который требуется проанализировать, а также указать формат хранения данных в нём (**hex**-формат или **bin**-формат). Ввести имя файла можно вручную или с помощью кнопки "**…**".

После нажатия на кнопку "**FILE INFO**" на экран будет выведен результат анализа файла.



Скриншот процесса получения информации о файле "*File Info*"

Uoclab загружает файл прошивки и анализирует его на соответствие различным типам микросхем, описание которых хранится во внутренней базе данных программы. Используется двухпроходной анализ сигнатуры файла: на первом проходе делается попытка установить точный тип микросхемы по GPF0, GPF1, GPF2, GPF3; если первый проход завершился неудачей, то на втором проходе делается попытка установить совместимые типы микросхемы по GPF0, GPF1.

Во время анализа также вычисляются специфические особенности для данного семейства микросхем, например, значение boot-регистра, поддерживаемые цветовые стандарты, возможность автоматического патча и т.п..

Рекомендуемые ссылки

Некоторую полезную информацию по программированию Painter / Painter-II / UOC / UOC+ / UOC-Leader / UOC-III / LOC-I /LOC-II / UOC-V / UOC-Top Painter / UOC-Top Fighter можно найти по следующим ссылкам:

http://www.anyram.net/forum/viewtopic.php?id=18
http://www.anyram.net/forum/viewtopic.php?id=25
http://www.anyram.net/forum/viewtopic.php?id=40
http://www.anyram.net/forum/viewtopic.php?id=42
http://www.anyram.net/forum/viewtopic.php?id=72

Приложение 1

Краткое описание особенностей семейства Painter / UOC

Всё семейство **Painter/UOC** по особенностям контроллерной части можно разбить на 3 группы:

1) Painter-I / Painter-II / UOC / UOC-Top Painter;
2) UOC-Leader / UOC-Top Fighter(Leader);
3) UOC-III / UOC-V / LOC-I / LOC-II.

Контроллерные группы между собой программно несовместимы несмотря на то, что все они относятся к контроллерам типа **MCS-51**.

1. Эта группа появилась исторически первой. Основные признаки группы: первичные признаки - наличие 3х областей памяти (CODE/CHAR/P26); вторичные признаки - наличие слайсера телетекста, поддержка режима текстового 40x25. Основные внутригрупповые различия идут по объёмам CHAR-памяти (Painter-I/UOC vs Painter-II), а также методам программирования Painter/UOC vs UOC-Top Painter). Основные различия внутри семейства идут по объёмам памяти CODE и наличию телетекста и P26 соответственно. Разрядность выборки CODE/P26 - 8бит, CHAR - 12бит. Все микросхемы **OTP**-типа, т.е. однократно программируемые.

<u>Примеры</u>:
Painter - SAA5564/M3, CODE 96K, CHAR 9K, есть P26.
Painter-II - SAA5667/M1, CODE 192K, CHAR 12K, есть P26.
UOC Plus - TDA9585PS/N1/3, CODE 64K, CHAR 9K, нет P26.
UOC-Top Painter - TDA12176PS/N2/2, CODE 48K, CHAR 9K, есть P26.

2. Эта группа появилась в попытке создать более дешёвый контроллер на замену группе 1. Дешевизна достигается наличием только одной области памяти, разделяемой между программой и знакогенератором: программа находится на младших адресах блока памяти, память символов - на старших. Название такого типа устройств в терминологии Philips/NXP - *Split Memory Devices*. Основные признаки группы: первичные признаки - наличие только одной области памяти; вторичные признаки - отсутствие телетекста. Основные внутригрупповые различия идут по объёмам памяти (64K в UOC-Leader vs 80K/160K в UOC-Top) и методам программирования (UOC-Leader vs UOC-Fighter). Основные различия внутри семейства идут по пропорции деления блока памяти между кодовой и символьной частями - чем больше памяти под символы, тем меньше памяти под код. Разрядность выборки данных 16бит. Все микросхемы **OTP**-типа, т.е. однократно программируемые.

<u>Примеры</u>:
UOC-Leader - TDA9341PS/N3/A, SPLIT 64K.
UOC-Fighter 80K - TDA11136PS/N2/3, SPLIT 80K.
UOC-Fighter 160K - TDA11138H/N2/5, SPLIT 160K.

3. Эта группа - попытка создать продвинутый контроллер Premium-типа с низкой стоимостью. В целом, 3я группа очень похожа на 1ю: увеличенная производительность, но те же области памяти, тот же принцип работы в целом. Основные внутригрупповые различия касаются программных различий при работе с OSD. Основные различия внутри семейства идут по объёмам памяти CODE - 128K/256K и наличию телетекста и P26 соответственно. Разрядность выборки CODE/P26 - 8бит, CHAR - 16бит. Все микросхемы **FLASH**-типа, т.е. многократно перепрограммируемые.

Примеры:

UOC-3 - TDA12025PQ/N1F 01, CODE 128K, CHAR 16K, есть P26.

UOC-5 - TDA15603E/N1B 40, CODE 256K, CHAR 16K, нет P26.

Приложение 2

Методика изменения кода UOC для миграции прошивок из микросхемы одного типа в микросхему другого типа

Данная методика предназначена исключительно для ремонта телевизоров на базе UOC, т.к. многие заводы-производители не выполняют своих обязательств по поставке ремкомплектов, а ремонтировать телевизоры надо.

Эта статья преследует попытку разъяснить основы устройства UOC и основы программирования микросхем с помощью программатора ф.Энирам: врядли найдётся другой программатор, читающий Painter/UOC, а идея переноса кода из одного типа UOC в другой тип UOC базируется именно на возможности чтения этих микросхем.

Следует отметить, что микроконтроллерная часть UOC отличается крайней “живучестью” и код считывается в 90% случаев даже из неисправных ИМС.

UOC – двухкристальная микросхема, т.е. состоит как бы из двух микросхем в одном корпусе: микроконтроллера и видеопроцессора, которая называется Октава (**Octava**) (используется терминология NXP/Philips). Основной секрет микросхемы в том, что UOC всех типов, как это ни странно, одинаковы!

Многообразие типов делается программным способом, и секрет этого многообразия кроется в программной инициализации микросхемы после подачи питания. Как это делается программно рассмотрено в статье “Секреты UOC – инициализация” по адресу: http://www.anyram.net/forum/viewtopic.php?id=18.

Все типы UOC отличаются друг от друга тем, что на заводе в области **CHARROM** программируются ячейки **GPF0**, **GPF1**, **GPF2**, **GPF3**, которые однозначно определяют тип и функционал микросхемы и называются **сигнатурой**. Значения, которые записаны по этим адресам на заводе, можно найти в базе данных NXP/Philips (пакет **DDS** - **D**isplay **D**evelopment **S**tudio) или считать из микросхемы самим. В связи с тем, что память UOC однократно программируемая (**OTP** - **O**ne **T**ime **P**rogramming), следует условно считать, что содержимое по этим адресам поменять нельзя - можно только обнулить.

При подаче питания, программа, записанная в кодовой памяти UOC, стартует и определяет, что написано в ячейках **GPF0-GPF1 CHARROM**; программа читает значения по этим адресам через регистры TXT12 и TXT22. По адресам **GPF0**, **GPF1** записана конфигурация UOC: можно считать, что каждый бит разрешает какую-либо фунцию микросхемы.

В **GPF2** устанавливается тип UOC – это просто 12ти битный номер: содержимое этого адреса ни на что не влияет при работе UOC, но это значение используется сборщиком прошивок **PROMPT** из пакета **Display Development Studio**.

В **GPF3** фирменный сборщик прошивок **PROMPT** обычно записывает 0, но значение по этому адресу может вообще не программироваться: сборщик игнорирует значение по этому адресу.

В зависимости от того, что считано из **GPF0**, **GPF1**, далее по таблице выбирается значение “секретного” **boot**-регистра по адресу 0xFF. “Секретным” этот регистр считается потому, что его нет ни в одной спецификации на UOC.

Установка бита в 1 включает функцию UOC, установка в 0 – выключает. Назначение битов **boot**-регистра:

бит 7 – назначение неизвестно;
бит 6 – включает QSS-функции UOC;
бит 5 – включает функции UOC+ (FM-Radio и др.);
бит 4 – активирует функцию East/West (EW или иначе 90°/110°);
бит 3 – активирует демодулятор AM (AM для L-стандартов и др.);
бит 2 – активирует декодер SECAM;
бит 1 - активирует декодер NTSC;
бит 0 - активирует декодер PAL;

Например, для микросхемы TDA9381 значение **boot**-регистра равно 0x07, т.е. активны декодеры PAL, SECAM, NTSC, отсутствует AM-демодулятор, отсутствует коррекция геометрии по горизонтали (на выводе 20 вместо функции East/West должен быть подключен конденсатор AVL), режимы FM-радио недоступны, QSS - неактивен (это FM-микросхема).

Для микросхемы TDA9353 значение **boot**-регистра равно 0x17, т.е. активны декодеры PAL, SECAM, NTSC, отсутствует AM-демодулятор, есть коррекция геометрии по горизонтали (на выводе 20 присутствует East/West-парабола, конденсатор AVL в этом случае подключается к выводу 32), режимы FM-радио недоступны, QSS - неактивен (это FM-микросхема).

С точки зрения инициализации Октавы, разницы между микросхемами с телетекстом или без него – нет, т.к. это микроконтроллерная часть. И для TDA9381, и для TDA9351 значение для инициализации **boot**-регистра одинаково и равно 0x07. А вот чтобы превратить UOC (TDA9351) в UOC+ (TDA9551) надо в **boot**-регистр вместо 0x07 записать 0x27.

Теперь, когда основной механизм идентификации/инициализации UOC описан, становится понятным, откуда берётся такое многообразие типов UOC: на самом деле <u>во всех UOC одна и та же начинка</u>!

Однако разница для корпусов SDIP-64 всё-таки есть: в связи с недостаточностью выводов в корпусе SDIP-64, NXP/Philips разваривает по-разному кристаллы для FM и QSS вариантов.

Это значит, что из одной QSS-UOC можно получить любую другую QSS-микросхему, а вот сделать FM - не удастся, т.к. выводы из кристалла не подведены к выводам на корпусе микросхемы. То же относится к FM-UOC: из одной чистой FM-болванки можно получить любую другую FM-микросхему, а вот QSS-микросхему получить не удастся. При прошивке FM в QSS и наоборот возникнут проблемы со звуком.

UOC в корпусе QFP80 и других, могут быть запрограммированы как QSS-микросхемы, так и FM-микросхемы, независимо от типа, указанного на корпусе, т.к. все выводы, присутствующие на кристалле разведены на соответствующие выводы на корпусе.

Теперь можно описать критерии, по которым отличаются чистые болванки UOC:

1. QSS или FM (только для корпусов SDIP-64);
2. количество страниц телетекста: 0/1/10;
3. объём **CODEROM** (памяти программы): 64Кб или 128Кб, других болванок UOC не встречалось; надо также отметить, что критерий объёма почти никогда не "всплывает" при выборе болванки: большинство прошивок не превышают 64Кб, а болванки не бывают менее 64Кб, т.е. критерий 3 можно не учитывать.

Объём **CODEROM** в UOC обозначает последняя или предпоследняя цифра в маркировке, 1-32Кб, 2-48Кб, 3-64Кб, 4-96Кб, 5-128Кб. Обозначение TDA9351PS/N2/1I говорит, что в микросхеме 32Кб памяти (последний блок 1I).

Объём **CHARROM** во всех UOC одинаков: 9Кслов (разрядность слова – 12 бит). Структура памяти такова, что компактный формат хранения символов ведёт к образованию “дыр”, которые не используются знакогенератором. Эти “дыры” попадают на адреса **0x9FE**, **0x9FF**, **0xBFE**, **0xBFF**, которые разумно используются NXP/Philips для хранения сигнатуры чипа, **GPF0**, **GPF1**, **GPF2**, **GPF3** соответственно.

В UOC, в которых есть блок приёма телетекста присутствует память для хранения перекодировочной таблицы пакета 26 телетекста – **P26ROM** (национальная кодировка).

Исходя из написанного, можно вывести следующие правила программирования UOC:

1. нельзя FM-прошивку зашить в QSS-микросхему и наоборот! Это верно только для корпусов SDIP-64 (буквы **PS** после типа микросхемы в маркировке на корпусе);

2. при прошивке без потери функционала надо следить за тем, чтобы UOC имела ресурсов (память и телетекст) не меньше, чем в оригинале (правило – “кашу маслом не испортишь”).

Например, прошивка из TDA9381 (в которой нет телетекста), будет отлично работать в болванке TDA9353 (с одностраничным телетекстом), или в болванке TDA9361 (с 10ти страничным телетекстом): телетекст будет в наличии, но не будет использоваться прошивкой.

Разумеется, прошивка может быть зашита в одноимённую болванку вообще без модификации, но на практике такая возможность предоставляется крайне редко: очень много прошивок имеют размер 32Кб (последняя цифра 1 в маркировке типа UOC), а болванки не бывают менее 64Кб (последняя цифра 3 в маркировке типа UOC). В связи с тем, что сигнатура прошивки и микросхемы, в которую будет зашиваться прошивка будут различны, всё равно требуется модификация прошивки (по адресам **GPF2**, **GPF3** в области **CHARROM**), хотя бы для того, чтобы проходила верификация.

Пример: есть болванка TDA9352PS/N2/3I (QSS, 1стр. телетекст), чем можно её запрограммировать?

UOC TDA9352 можно без потери функционала запрограммировать прошивкой из TDA9382: TDA9382 это тоже самое, что TDA9352 только без телетекста. “Лишний” телетекст в 9352 роли не играет – правило “кашу маслом не испортишь”.

Хотя, если в TDA9352 зашить прошивку из TDA9365 (QSS с 10ти страничным телетекстом), то в тоже ничего страшного не произойдёт: с точки зрения телезрителя, после нажатия на кнопку “TxT”, телевизор покажет экран телетекста, который просто не будет приниматься, или телетекст будет работать в одностраничном режиме (зависит от прошивки). Поскольку трансляций с уверенным приёмом телетекста не так много, можно просто пожертвовать телетекстом, ведь остальной функционал остаётся неизменным: пользователи телевизоров на практике не замечают потери телетекста.

Также ничто не мешает запрограммировать TDA9382 (нет телетекста), прошивкой из TDA9365(10 стр.телетекст): видимость телетекста сохранится (экран, режимы и т.п.), а приниматься ничего не будет.

Дальнейшие действия будут пояснены на примере масочной прошивки телевизора “Витязь” TDA9351PS/N2/1I /1610. С помощью программатора прошивку следует считать в файл ***9351_1610.bin***. В наличии есть только болванки TDA9353PS/N3/3.

<u>Шаг 1: модификация сигнатуры</u>

Помним, что на заводе уже установлена сигнатура: записаны адреса **CHARROM GPF0**, **GPF1**, **GPF2**, **GPF3** и, если прошивку запрограммировать без изменений, то кроме того, что не пройдёт верификация (сигнатуры в микросхемах разные), не будет работать AVL, и микросхема будет выводить сжатое изображение, т.к. в сигнатуре TDA9353 включена East/West функция.

Программирование новой сигнатуры поверх старой приведёт к тому, что по адресам **GPF0**, **GPF1**, **GPF2**, **GPF3** в **CHARROM** будет полная “каша”, которая не будет соответствовать ни одному из типов UOC. При повторном программировании биты, которые уже установлены в 0 в сигнатуре, не поднимутся в 1, а те, которые находятся в 1, опустятся в 0.

Видимо, на это и был коммерческий расчёт у ф.Philips, когда его специалисты решили, что никто не сможет менять тип болванок, а при продаже, болванки TDA9353 стоят дороже, чем TDA9351, т.к. они имеют согласно спецификации больше функциональных возможностей, хотя физически между этими микросхемами нет разницы.

Для того, чтобы проходила верификация, в прошивке следует установить значение сигнатуры таким же, каким оно будет в болванке, в которую эта прошивка будет программироваться. Для этого вставляем в программатор чистую болванку TDA9353 и читаем её содержимое в **bin**-файл, например ***empt9353.bin***.

Далее открываем empt9353.bin в любом двоичном редакторе (например, **Ultraedit**). Программатор создаёт файл в формате фирменного сборщика прошивок - **PROMPT**. В этом формате с адреса 0x00000 хранится **CODEROM** (8-битные данные), а 12-битные данные для **CHARROM** разбиваются на 2 части: младшую и старшую. Младшие тетрады данных записываются начиная с адреса 0x40000, а старшие тетрады записывается начиная с адреса 0x50000.

Например, если по адресу 0x420 в **CHARROM** записано число 0x123, то в файле в формате **PROMPT** это будет выглядеть так:

по адресу 0x40000+0x420=0x40420 будет записано 0x23 (две младшие тетрады),
по адресу 0x50000+0x420=0x50420 будет записано 0xF1 (старшая тетрада).

Также по адресу 0x50420 можно вписать 0x01, но так уж повелось, что незначащие биты заполняют единицами.

Смотрим значения адресов **GPF0**(0x9FE), **GPF1**(0x9FF**)**, **GPF2**(0xBFE**)**, в считанном из пустой болванки файле ***empt9353.bin***:

[0x409FE] = 0x67, [0x509FE] = 0xF7: 12-битное значение [0x9FE]=0x767;
[0x409FF] = 0xF6, [0x509FF] = 0xF2: 12-битное значение [0x9FF]=0x2F6;
[0x40BFE] = 0x21, [0x50BFE] = 0xFC: 12-битное значение [0xBFE]=0xC21.

Всё необходимое о болванке узнали, файл ***empt9353.bin*** можно удалить.

Теперь открываем файл с прошивкой "Витязя" ***9351_1610.bin*** в двоичном редакторе (например, **Ultraedit**), и начинаем исправлять прошивку под TDA9353PS/N3/3.

Вписываем в прошивку значения для нашей болванки:

CHARROM GPF0:
по адресу 0x409FE вписываем 0x67 (две мл.тетрады [0x9FE]=0x767)
по адресу 0x509FE вписываем 0xF7 (ст.тетрада [0x9FE]=0x767),
в принципе, можно вписывать и 0x07, но так уж повелось,
что незначащие биты заполняют единицами; кстати, следует
заметить, что в этих ячейках уже и было записано 0x767,
т.е. значение не поменялось;

CHARROM GPF1:
по адресу 0x409FF вписываем 0xF6 (две мл.тетрады [9FF]=0x2F6)
по адресу 0x509FF вписываем 0xF2 (ст.тетрада [9FF]=0x2F6),
старое значение, которое было в ячейке [9FF] 0x0F5;

CHARROM GPF2:
по адресу 0x40BFE вписываем 0x21 (две мл.тетрады [BFE]=0xC21)
по адресу 0x50BFE вписываем 0xFC (ст.тетрада [BFE]=0xC21);

CHARROM GPF3:
не обращаем внимания на этот адрес.

Следует отметить, что старое значение из прошивки ***9351_1610.bin***, которое было по адресу **GPF2** равно 0xC0B, что соответствует TDA9351PS/N2/1, а болванки с размером 32Кб в природе не встречается, т.е. модификация хотя бы сигнатуры необходима в любом случае. Даже если бы удалось достать болванку TDA9351PS/N3/3 потому, что не бывает болванок с индексом меньше 3 – индексы 1 и 2 встречаются только в масочных прошивках.

Шаг 2: модификация boot-таблицы

Если этого не сделать, то болванка TDA9353 с прошивкой Витязя будет вести себя как настоящая TDA9353, хотя требуется полноценная замена TDA9351.

TDA9353, в которой изменена только сигнатура, будет выдавать пилу для диодного модулятора на выводе 20, хотя на плате к этому выводу подключен конденсатор для AVL (это схема подключения TDA9351). Конденсатор для AVL наша болванка будет "ждать" на выводе 32. При этом все размеры по вертикали также будут сжаты, т.к. в TDA9353 появляется регистр VX (expand) о котором прошивка для TDA9351 "ничего не знает".

Выше было указано, что при старте по таблице выбирается значение **boot**-регистра исходя из значений сигнатуры **GPF0**, **GPF1**. Заменив в таблице все значения на то, которое требуется для инициализации TDA9351, мы заставим работать болванку TDA9353 в требуемом для нас режиме, независимо от того, какая сигнатура у этой микросхемы записана на заводе.

Philips/NXP в своих библиотеках для UOC использует следующую таблицу **boot**-значений:

0B 0F 3B 3F 7F 4F FF 0A 3A 23 27 6F 33 37 32 7F 4F 7B 6B 22 03 07 17 13 43 02 12 7A 47 53 5F

Именно эту последовательность и надо искать в коде. Искомая строка находится по адресу 0x77A9. Теперь смотрим, какое значение **boot**-регистра должно быть у TDA9351 – ведь именно её должна имитировать болванка.

№ п/п	Значение **boot**-регистра	Микросхемы, соответствующие значению инициализации **boot**-регистра
0	0x0B	TDA9350/N1,TDA9360/N1,TDA9366/N1,TDA9380/N1
1	0x0F	TDA9351/N1,TDA9352/N1,TDA9361/N1,TDA9381/N1,TDA9382/N1
2	0x3B	TDA9362/N1,TDA9364/N1,TDA9383/N1,TDA9385/N1
3	0x3F	TDA9353/N1,TDA9355/N1,TDA9363,TDA9384/N1
4	0x7F	TDA9365/N1,TDA9386/N1,TDA9555H/N1
5	0x4F	TDA9367/N1
6	0xFF	TDA9391,TDA9395
7	0x0A	TDA9387/N1
8	0x3A	TDA9388/N1
9	0x23	TDA9550PS/N1,TDA9560H/N1,TDA9580H/N1
10	0x27	TDA9551H/N1,TDA9561PS/N1,TDA9581H/N1,TDA9551PS/N1
11	0x6F	TDA9552H/N1,TDA9567H/N1,TDA9567PS/N1,TDA9582H/N1, TDA9582PS/N1
12	0x33	TDA9562H/N1,TDA9583H/N1,TDA9583PS/N1
13	0x37	TDA9553H/N1,TDA9563PS/N1,TDA9584H/N1,TDA9584PS/N1
14	0x32	TDA9569H/N1,TDA9588H/N1,TDA9588PS/N1
15	0x7F	TDA9565PS/N1,TDA9586H/N1,TDA9586PS/N1,TDA9590H/N1, TDA9590PS/N1,TDA9591H/N1,TDA9592H/N1,TDA9592PS/N1, TDA9593H/N1,TDA9593PS/N1,TDA9594H/N1,TDA9594PS/N1
16	0x4F	TDA9554PS/N1
17	0x7B	TDA9564H/N1,TDA9585H/N1,TDA9585PS/N1
18	0x6B	TDA9566H/N1,TDA9566PS/N1
19	0x22	TDA9568H/N1,TDA9587H/N1,TDA9587PS/N1
20	0x03	TDA9350PS/N2,TDA9360PS/N2
21	0x07	TDA9351PS/N2,TDA9381PS/N2
22	0x17	TDA9353PS/N2,TDA9363PS/N2,TDA9384PS/N2

23	0x13	TDA9362PS/N2,TDA9383PS/N2
24	0x43	TDA9366PS/N2
25	0x02	TDA9387PS/N2
26	0x12	TDA9388PS/N2
27	0x7A	TDA9589H/N1,TDA9589PS/N1
28	0x47	TDA9352PS/N2,TDA9367PS/N2,TDA9382PS/N2
29	0x53	TDA9364PS/N2,TDA9385PS/N2
30	0x5F	TDA9365PS/N2,TDA9386PS/N2

Для TDA9351 значение boot-регистра равно 0x07. Теперь заменяем все значения boot-регистров (31 элемент) в таблице по адресу 0x77A9 на 0x07: после этой операции какой бы тип UOC ни определился по сигнатуре (а определится TDA9353), всё-равно UOC будет инициализирована тем значением, которое было в оригинальной микросхеме (TDA9351), т.к. мы “забили” все элементы таблицы этим значением.

На самом деле в таблице достаточно заменить одно значение (№ 22), которое соответствует TDA9353: вместо 0x17 следует написать 0x07. Изменения всей таблицы были сделаны для того, чтобы не вычислять какой именно элемент надо изменить.

Шаг 3: программирование

Прошивка готова, сохраняем исправленный код в файл ***9353_1610.bin***, а затем программируем прошивку в болванку TDA9353PS/N3/3.

Вот и всё, получен полный функционал TDA9351 в микросхеме TDA9353: теперь телевизор можно полноценно отремонтировать – методика неоднократно отработана.

Замечания

Последовательность (таблица) **boot**-регистров в разных прошивках может немного отличаться от приведенной. Например, в китайских телевизорах с маркировкой UOC, начинающейся с ***CH*** или ***NTDA*** попадается последовательность:

0B 0F 3B 3F 7F 4F 7F 0A 3A 23 27 6F 33 37 32 7F 4F 7B 6B 22 03 07 17 13 43 02 12 7A 47 53 5F

В телевизорах Thomson(TDA9554) попадается последовательность длиной 22 байта:

0B 0F 3B 3F 7F 4F 7F 0A 3A 23 27 6F 33 37 32 7F 4F 7B 6B 22

Несмотря на то, что последовательности разные, методика изменения прошивки остаётся прежней.

Бывает, что последовательность не находится, и это значит, что инициализация UOC делается не с помощью стандартных библиотек NXP/Philips, а простой записью требуемого значения boot-регистра: многие фирмы давно раскусили секрет и инициализируют UOC без использования фирменных библиотек. В этом случае, достаточно поменять только значения сигнатуры, чтобы проходила верификация, а прошивка сама сконфигурирует UOC под себя.

Достаточно редко попадаются UOC с нестандартным способом определения типа UOC, например, прошивка TDA9353PS/N2/2i /0869 вообще не предусматривает поддержки SECAM и использует специфическую процедуру инициализации с элементами защиты. **Boot**-последовательность в этой прошивке находится по адресу 0x884B и выглядит так:

0B 0F 1B 1F 5F

Для модификации этой прошивки надо заменить все байты последовательности на 0x17.

Наиболее универсальный способ - это дизассемблирование кода и выяснение особенностей инициализации и работы, присущие именно конкретной дизассемблированной прошивке.

Приведённая выше методика позволяет перемещать прошивки между разными типами UOC в 97% случаев.

Приложение 3

Методика изменения кода UOC-Leader для миграции прошивок из микросхемы одного типа в микросхему другого типа

Данная методика предназначена исключительно для ремонта телевизоров на базе UOC, т.к. многие заводы-производители не выполняют своих обязательств по поставке ремкомплектов, а ремонтировать телевизоры надо.

UOC – двухкристальная микросхема, т.е. состоит как бы из двух микросхем в одном корпусе: микроконтроллера и видеопроцессора, которая называется Октава (**Octava**) (используется терминология NXP/Philips). Основной секрет микросхемы в том, что UOC всех типов, как это ни странно, одинаковы!

Многообразие типов делается программным способом, и секрет этого многообразия кроется в программной инициализации микросхемы после подачи питания. Описание того, как это происходит, можно прочитать в **приложении 1**. Как это делается программно рассмотрено в статье "Секреты UOC – инициализация" по адресу: http://www.anyram.net/forum/viewtopic.php?id=18.

Со временем семейство UOC развилось, и появились новые ветви UOC: UOC3, UOC-Top, UOC-Leader, LOC.

UOC-Leader – это схемотехнически та же микросхема, что и UOC/UOC+ с совпадающей цоколёвкой: видеопроцессор такой же как и в обычной UOC, а вот микроконтроллер - более дешёвый. Удешевление достигнуто за счёт объединения памяти программы (**CODEROM**) и символьной памяти(**CHARROM**) – в этих ИМС это физически одно целое; физическая разрядность памяти 16бит. Также в этих ИМС не бывает телетекста и, соответственно, отсутствует память пакета 26 (**P26ROM**). В связи с таким построением, UOC-Leader также называют "**Split Memory Device**".

NXP/Philips остаётся верным своим традициям и обеспечивает разнообразие микросхем UOC-Leader прописыванием типа чипа в OTP память: все UOC-Leader, как и обычные UOC, физически одинаковы: ИМС отличаются друг от друга тем, что на заводе в памяти программируются ячейки **GPF0**(0x7FF8), **GPF1**(0x7FF9), **GPF2**(0x7FFA), которые однозначно определяют тип и функционал микросхемы и называются **сигнатурой**. Значения, которые записаны по этим адресам на заводе, можно найти в базе данных NXP/Philips (пакет **DDS** - **D**isplay **D**evelopment **S**tudio) или считать из микросхемы самим. В связи с тем, что память UOC однократно программируемая (**OTP** - **O**ne **T**ime **P**rogramming), следует условно считать, что содержимое по этим адресам поменять нельзя - можно только обнулить.

При подаче питания, программа, записанная в кодовой памяти UOC-Leader, стартует и определяет, что записано по ***GPF0*** и ***GPF1***: по этим адресам записана конфигурация UOC-Leader; можно считать, что каждый бит разрешает какую-либо фунцию микросхемы.

В зависимости от того, что записано в ***GPF0***, ***GPF1*** путём аппаратной перестановки битов строится значение регистра TXT32 в котором находится значение для инициализации "секретного" **boot**-регистра по адресу 0xFF. "Секретным" этот регистр считается потому, что его нет ни в одной спецификации на UOC-Leader или UOC.

Установка бита в 1 включает функцию UOC-Leader, установка в 0 – выключает. Назначение битов **boot**-регистра:

бит 7 – назначение неизвестно;
бит 6 – включает QSS-функции UOC;
бит 5 – включает функции UOC+ (FM-Radio и др.);
бит 4 – активирует функцию East/West (EW или иначе 90°/110°);
бит 3 – активирует демодулятор AM (AM для L-стандартов и др.);
бит 2 – активирует декодер SECAM;
бит 1 - активирует декодер NTSC;
бит 0 - активирует декодер PAL;

Например, для микросхемы TDA9341 значение **boot**-регистра равно 0x07, т.е. активны декодеры PAL, SECAM, NTSC, отсутствует AM-демодулятор, отсутствует коррекция геометрии по горизонтали (на выводе 20 вместо функции East/West должен быть подключен конденсатор AVL), режимы FM-радио недоступны, QSS - неактивен (это FM-микросхема).

Для микросхемы TDA9370 значение **boot**-регистра равно 0x17, т.е. активны декодеры PAL, SECAM, NTSC, отсутствует AM-демодулятор, есть коррекция геометрии по горизонтали (на выводе 20 присутствует East/West-парабола, конденсатор AVL в этом случае подключается к выводу 32), режимы FM-радио недоступны, QSS - неактивен (это FM-микросхема).

С точки зрения инициализации Октавы, разницы между микросхемами с телетекстом или без него – нет, т.к. это микроконтроллерная часть. И для TDA9381, и для TDA9351 значение для инициализации **boot**-регистра одинаково и равно 0x07. А вот чтобы превратить UOC (TDA9351) в UOC+ (TDA9551) надо в **boot**-регистр вместо 0x07 записать 0x27.

Теперь, когда основной механизм идентификации/инициализации UOC описан, становится понятным, откуда берётся такое многообразие типов UOC: на самом деле <u>во всех UOC одна и та же начинка</u>!

Методика патча UOC-Leader следующая:

1. Ищем последовательность **E4 F5 xx <u>85</u> xx <u>RR</u> 12 xx xx 7F 01 12**, где **xx** и **RR** могут быть любым числом!
2. В найденной последовательности меняем **85** (это код команды) на **75**, а вместо **RR** записываем **boot**-значение, которое нам надо - например, 0x07 (PAL, SECAM, NTSC).

Наиболее универсальный способ - это дизассемблирование кода и выяснение особенностей инициализации и работы, присущие именно конкретной дизассемблированной прошивке.

Приведённая выше методика позволяет перемещать прошивки между разными типами UOC в 80% случаев.