

Реализация алгоритмов ЦОС в ИСР ССС

Игорь ГУК
gii@scanti.ru

В данной статье показан пример разработки программного кода для ЦСП семейства TMS320C6000 в рамках интегрированной среды разработки (ИСР) Code Composer Studio (ССС). Материал базируется на предыдущей статье, опубликованной в журнале «Компоненты и технологии» № 7 за этот год, и для успешного освоения излагаемых вопросов рекомендуется с ней ознакомиться.

Реализация любой системы ЦОС на базе ЦСП включает:

- Разработку и отладку программного кода на языке высокого уровня или ассемблере в соответствии с выбранным алгоритмом. Результатом является набор текстовых файлов с программным кодом, описывающим реализуемый алгоритм.
- Оптимизацию программного кода с учетом архитектуры выбранного ЦСП. Результатом является один бинарный файл, содержащий цифровое двоичное представление разработанного программного кода.
- Тестирование разработанного программного кода.
- Разработку платы для проектируемой системы ЦОС.
- Тестирование программного кода на разработанной плате.

Для выполнения всех вышеперечисленных этапов необходимо иметь комплекс программных и аппаратных средств проектирования систем ЦОС (программное обеспечение и средства аппаратной поддержки).

Программное обеспечение (ПО) включает:

- текстовый редактор для написания программного кода;
- компилятор для трансляции файлов с кодом программы в бинарный файл;
- симулятор ЦСП для отладки кода в виртуальном режиме;
- загрузчик для переноса полученного бинарного кода программы из файла в память процессора.

Средства аппаратной поддержки (САП) представляют собой:

- персональный компьютер (ПК) для реализации возможностей ПО;
- отладочный модуль для тестирования бинарного файла с кодом программы;
- программатор для переноса разработанного кода в память ЦОС;
- эмулятор для тестирования разработанной платы.

Для реализации систем ЦОС фирма TI предлагает ряд технологических решений,

позволяющих существенно ускорить и облегчить процесс разработки:

- единый, интегрированный в каждый ЦСП фирмы TI отладочный интерфейс JTAG, позволяющий подключить несколько ЦСП к одному отладчику всего по 6 проводам и обеспечивающий высокую скорость обмена данными;
- технологию обмена данными в реальном времени — RTDX, базирующуюся на интерфейсе JTAG;
- библиотеку функций DSP/BIOS, использующую технологию RTDX и позволяющую

реализовать алгоритмы обмена данными в реальном времени;

- стандарт eXpresDSP, определяющий основные правила написания программного кода для реализации алгоритмов ЦОС на ЦСП, позволяющий упростить разработку программ и увеличить повторное использование кода.

Программная часть этих технологий воплотилась в интегрированной среде разработки ССС (Code Composer Studio), которая включает:

- текстовый редактор;

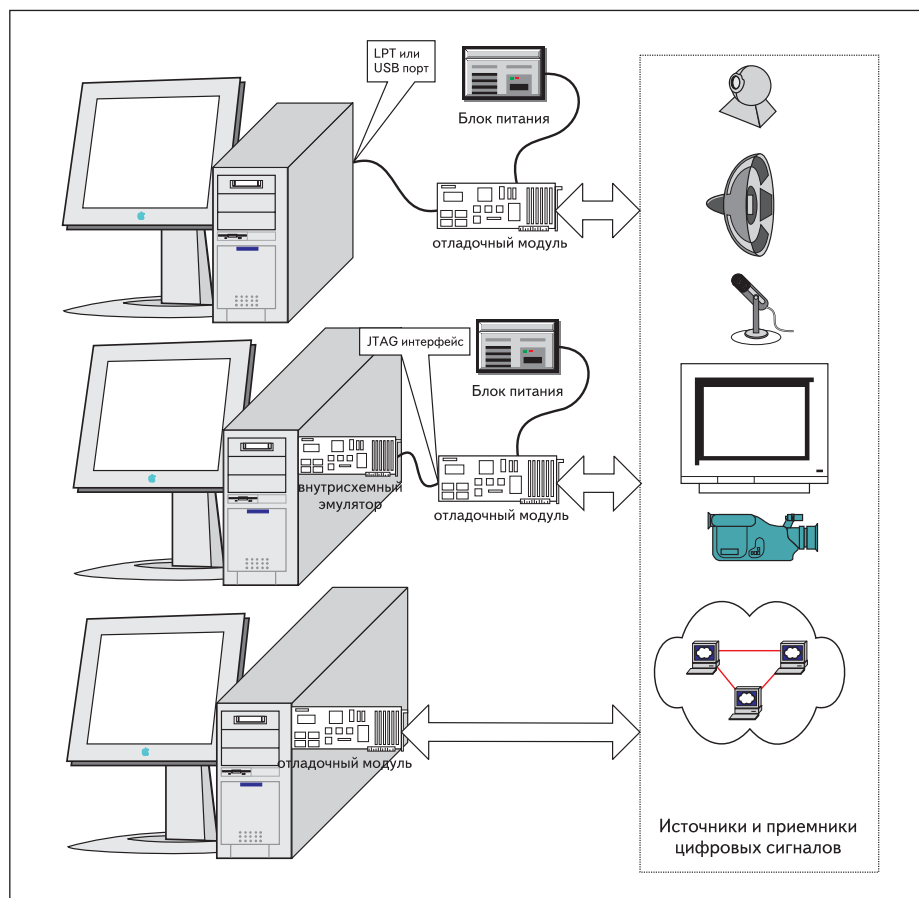


Рис. 1. Структуры программно-аппаратного комплекса разработки программного кода для ЦСП

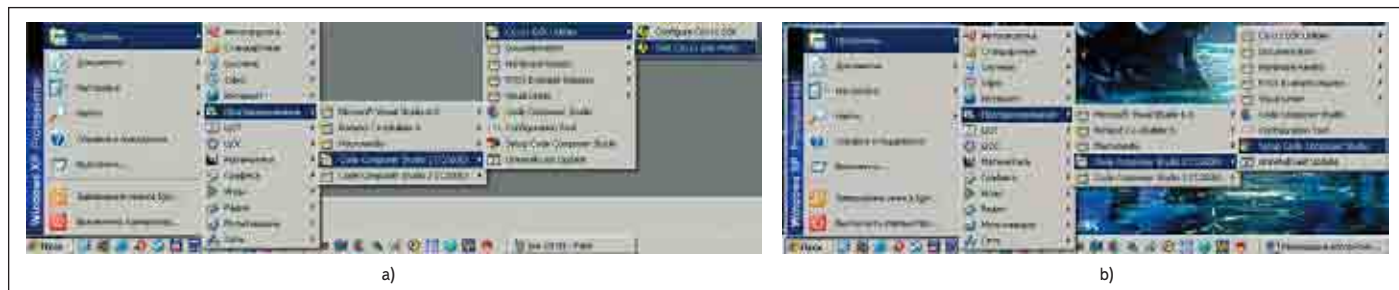


Рис. 2. Запуск утилит

- менеджер проектов;
- оптимизирующий компилятор;
- симулятор ЦСП;
- загрузчик программного кода в ЦСП;
- мультипроцессорный отладчик, оптимизированный для приложений ЦОС, который содержит уникальный набор средств анализа и отладки программ в реальном времени, базирующийся на RTDX и DSP/BIOS. Аппаратным элементом являются платы двух типов:

- внутрисхемный эмулятор;
- отладочный модуль.

Внутрисхемный эмулятор представляет модуль, устанавливаемый в слот PCI или подключаемый к персональному компьютеру при помощи порта USB или LPT, и кабель для подключения к отлаживаемой плате. Все ЦСП фирмы TI имеют единый интерфейс внутрисхемного эмулятора JTAG и работают с единым внутрисхемным эмулятором XDS-510 (или SDS-P-510). Через отладочный интерфейс доступны как внутренние регистры процессора, так и вся память и периферия. Эмулятор позволяет:

- производить загрузку кода программы и данных во внутреннюю и внешнюю память;
- устанавливать любое количество точек останова;
- производить контроль и модификацию содержимого памяти, регистров процессора и регистров периферийных устройств;
- проводить пошаговое выполнение программы;
- измерять время выполнения программы или ее частей;
- работать с несколькими ЦПОС.

Подключение внутрисхемного эмулятора абсолютно прозрачно для исполняемой программы и не оказывает на выполнение никакого влияния, при этом программа исполняется в реальном времени, без каких-либо задержек и ограничений по производительности. Такой подход радикально отличается от традиционного, при котором предполагается подключение на место процессора либо специальной микросхемы — прототипа, либо специального устройства — эмулятора ЦСП.

Отладочный модуль выполняется в виде отдельной платы, которая содержит:

- один из типов ЦПОС;
- внешнюю память;

- источник питания;
- JTAG контроллер;
- разъемы для подключения дочерних плат;
- разъемы подключения к ПК;
- дополнительные элементы, такие как АЦП/ЦАП, контроллеры сети, аудио-, видеокордеки, адаптеры PCI, шины и т. д.

Отладочный модуль может подключаться к ПК как через внутрисхемный эмулятор, так и напрямую через порты LPT или USB. Однако во втором случае обмен данными между ПК и отладочным модулем происходит значительно медленнее.

Ассортимент предлагаемых отладочных модулей очень широк: от плат начального уровня до специализированных устройств, позволяющих построить программно-аппаратные комплексы реализации алгоритмов обработки аудио и видеоданных, сетевые протоколы и т. д. На рис. 1 показаны возможные структуры программно-аппаратного комплекса.

Предлагаемые ПО и САП позволяют объединить весь процесс разработки систем ЦОС и выделить всего три этапа:

- *первый* — разработка и отладка программного кода с использованием ИСР CCS и одного из наиболее соответствующих поставленной задаче отладочных модулей;
- *второй* — разработка платы для проектируемой системы ЦОС;
- *третий* — отладка программного обеспечения на разработанной плате с использованием CCS и внутрисхемного эмулятора. Таким образом, для полноценной работы с ЦСП фирмы TI необходимы только три компонента:
- один из отладочных модулей, в наибольшей степени пригодный для решения поставленной задачи;
- один наиболее подходящий по характеристикам внутрисхемный эмулятор;
- интегральная среда разработки (ИСР) — Code Composer Studio (CCS).

Использование ПО и САП фирмы TI позволяет оценить разрабатываемую систему ЦОС и отработать ее решения на реальном ЦСП на самых ранних стадиях проектирования.

Более подробно ИСР CCS планируется рассмотреть в будущем. В данной статье приведены минимально необходимые знания по использованию ИСР. Возможности CCS показаны на примере создания программно-

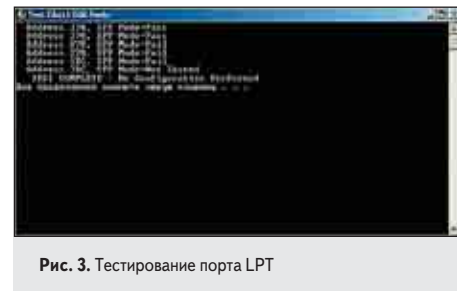


Рис. 3. Тестирование порта LPT

го кода фильтра, разработка которого описана в предыдущей статье.

После инсталляции CCS при первом запуске программы вначале необходимо произвести настройку ИСР для работы с конкретным подключенным к ПК отладочным модулем. Если отладочный модуль подключен через порт LPT, то вначале необходимо запустить утилиту определения номера порта и режима его работы (рис. 2a). Результат работы показан на рис. 3.

Затем запускается утилита *Setup Code Composer Studio*. Это можно сделать через главное меню (рис. 2b) или нажав на соответствующую иконку на рабочем столе. Появится диалоговое окно *Import Configuration* (рис. 4). Вначале необходимо нажать кнопку *Clear*, это приведет к сбросу всех настроек CCS. Затем указывается семейство ЦСП (рис. 4a), способ подключения отладочного модуля к ПК (рис. 4b), взаимное расположение младшего и старшего байтов (рис. 4c). Тип подключения определяется следующим образом (рис. 1): DSK — подключение через порт LPT или USB, EMU — через внутрисхемный эмулятор, EVM — использование отладочного модуля, включаемого непосредственно в один из слотов ПК. Потом в окне *Available Configurations* выбирается тип отладочного модуля. Пример выбора типа отладочного модуля при подключении через порт LPT с учетом номера порта и режима его работы (EPP или SPP) показан на рис. 4d. Далее необходимо нажать кнопку *Import* и завершить настройку нажатием кнопки *Save and Quit*. В результате появится окно запроса, где необходимо нажать кнопку подтверждения. Если настройка прошла успешно, будет запущена ИСР CCS (рис. 5). Процедура настройки ИСР показана на примере CCS версии 2. Для версии 3 основные этапы настройки сохраняются, но сам интерфейс (диалого-



Рис. 8. Подключение файлов к проекту

а в нем — пункт *Add Files to Project*. Появится диалоговое окно *Add Files to Project* (рис. 8), где указываются тип (файлы с расширением .cpp) и выбираются имена подключаемых файлов. Файлы подключаются к проекту после нажатия кнопки подтверждения. Заголовочный файл (с расширением .h) подключается автоматически при компиляции проекта. Находясь в диалоговом окне *Add Files to Project*, необходимо контролировать размещение подключаемых файлов на жестком диске (окно «Папка» на рис. 8), так как при открытии окна не происходит автоматический переход в папку проекта. Открытие происходит на том месте, где окно было закрыто в последний раз.

При создании кода для ЦСП необходимо распределить память процессора. В семействе TMS320C6000 для этого существует так называемый механизм секций: все функции и данные размещаются в поименованных секциях, а сами секции размещаются в различных областях памяти. Существует стандартный набор секций, который должен быть определен всегда. Однако можно назначить дополнительные секции и разместить в них код функций и данные разработанного алгоритма.

Распределение памяти указывается в файле с расширением «.cmd». Для начала необходимо его создать. В главном меню CCS выбирают раздел *File*, затем пункт *New* и подпункт *Sours File* (рис. 9).

В ИСП появится рабочая область текстового редактора, где и пишется код (рис. 10).

Образцы этих файлов можно посмотреть в поставляемых вместе с ИСП CCS учебных примерах. Листинг созданного файла для рассматриваемого имеет вид:

```
MEMORY
{
  IPRAM : origin = 0x0, len = 0x10000
  IDRAM : origin = 0x80000000, len = 0x10000
}

SECTIONS
{
  .vectors > IDRAM
  .text > IDRAM

  .bss > IDRAM
  .cinit > IDRAM
  .const > IDRAM
  .far > IDRAM
  .stack > IDRAM
  .cio > IDRAM
  .system > IDRAM
}
```

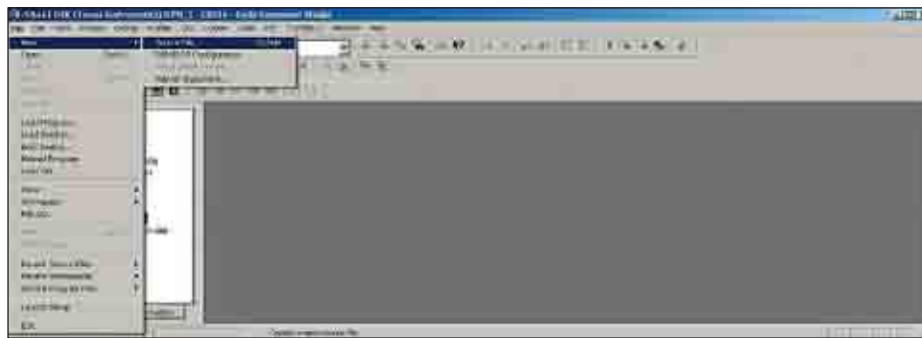


Рис. 9. Создание файлов с исходным кодом

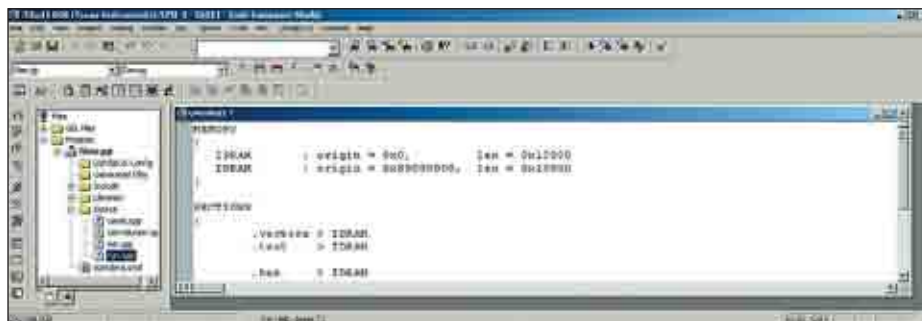


Рис. 10. Написание программного кода функции

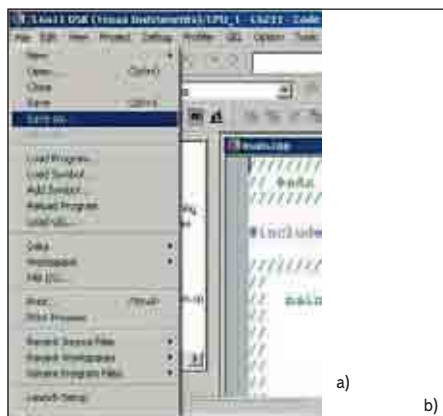


Рис. 11. Сохранение файлов с исходным кодом

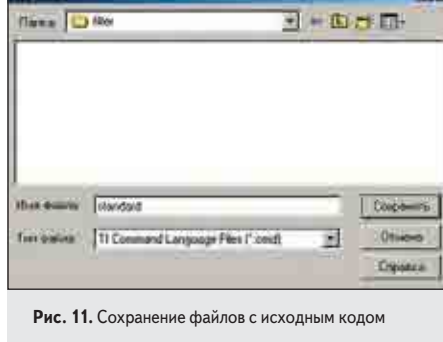


Рис. 11. Сохранение файлов с исходным кодом

В блоке MEMORY указаны имена областей памяти, их начальный адрес и длина. В блоке SECTIONS — имена секций и область памяти, где они расположены.

Набранный файл необходимо сохранить, указав тип «.cmd» и имя, например, «standard» (рис. 11a и 11b). Затем файл необходимо подключить к проекту, так как, в отличие от VS, автоматически это не происходит.

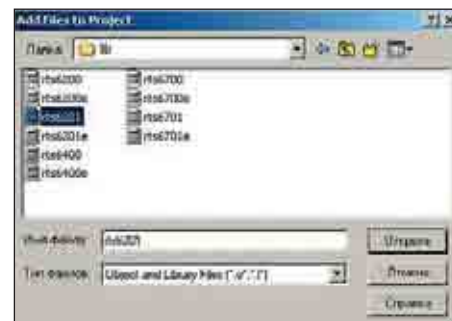


Рис. 12. Подключение библиотеки

Дальнейшая настройка проекта заключается в подключении внешней библиотеки, необходимой для работы с функциями языка C. Подключение происходит так же, как и в предыдущих случаях. Необходимо только правильно указать тип файла и путь к нему. В рассматриваемом примере подключается библиотека с именем «rts6201.lib», которая расположена в директории, где установлена ИСП CCS, в папке «c6000\cgtools\lib» (рис. 12).

ЦСП фирмы TI имеют как внутреннюю память, расположенную внутри чипа, так и возможность подключения внешней памяти. Для обеспечения работы с внешней памятью необходимо установить режим дальней адресации. Для этого в главном меню выбирают раздел *Project*, затем пункт *Build Options* (рис. 13a). Появится окно настройки параметров проекта (рис. 13b). В нем на закладке *Compiler* в окне *Category* выбирают позицию *Advanced*. Затем в окне *Memory Models* задают параметр *Far Calls & Data*.

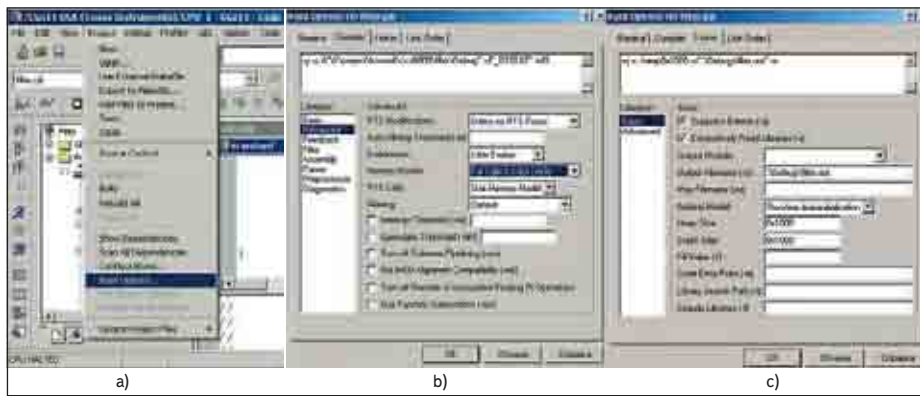


Рис. 13. Определение дополнительных параметров проекта




В рамках файловой модели не рекомендуется использовать динамическое распределение памяти. Однако CCS при создании выходного бинарного файла использует динамическую память («хип», или «Heap»), особенно при вызове функций языка C. Поэтому необходимо указать размер памяти, которая выделяется под динамические объекты. Это можно сделать в окне настроек параметров проекта на закладке *Linker* в категории *Basic*, где в окне *Heap Size* указывается размер динамической памяти (рис. 13с). В этой же категории в окне *Stack Size* необходимо указать размер области памяти, выделяемой под стек (рис. 13с). Размер стека и «хипа» можно определить экспериментально, при тестовых запусках проекта. В рассматриваемом примере используется одинаковый размер этих областей памяти, равный 0×1000 .

Следующий шаг настройки проекта — определение карты памяти. В главном меню необходимо выбрать раздел *Options*, далее — пункт *Memory Map* (рис. 14а). Появится окно настройки карты памяти (рис. 14б). В нем необходимо снять флажок с позиции *Enable Memory Mapping*. В этом случае CCS будет руководствоваться картой памяти, определенной в файле «*standard.cmd*».

И, наконец, заключительный этап настройки проекта — определение порядка загрузки бинарного файла в ЦСП. В примере используется автоматическая загрузка после каждой трансляции исходных текстовых файлов в исполняемый бинарный модуль (файл с расширением «.out»). Для установки этого режима необходимо выбрать раздел *Options* главного меню ИСР, далее — пункт *Customize*

(рис. 15а). В появившемся окне выбрать закладку *Program Load After Build*. На этой закладке установить флажок в позиции *Load Program After Build* (рис. 15б).

Проект собран и готов к компиляции. Ее можно осуществить в трех режимах (аналогичных режимам компиляции VS):

- Компиляция всех файлов проекта (значок  на панели быстрых кнопок CCS). При этом все файлы с расширением «.crr» транслируются в объектные модули (файлы с расширением «.obj»). Затем объектные модули компонуются в выходной файл с расширением «.out». Выходной файл содержит исполняемый бинарный модуль. Исполняемый модуль необходимо загрузить в память процессора. Это происходит автоматически в режиме *Load Program After Build*. Кроме того, это можно сделать через главное меню CCS: в пункте *File* выбрать подпункт *Load Program* (рис. 16а). В появившемся окне указать место расположения (папка *Debug* проекта), тип (.out) и имя загружаемого файла (рис. 16б).
- Компиляция только файлов проекта, которые были изменены после предыдущей трансляции (значок ). В этом режиме в объектные модули транслируются только измененные после последней компиляции исходные файлы. После этого происходит формирование нового выходного файла.
- Трансляция только одного активного (открытого в данный момент в текстовом редакторе CCS) файла (значок ). В этом режиме компоновка файлов не происходит,

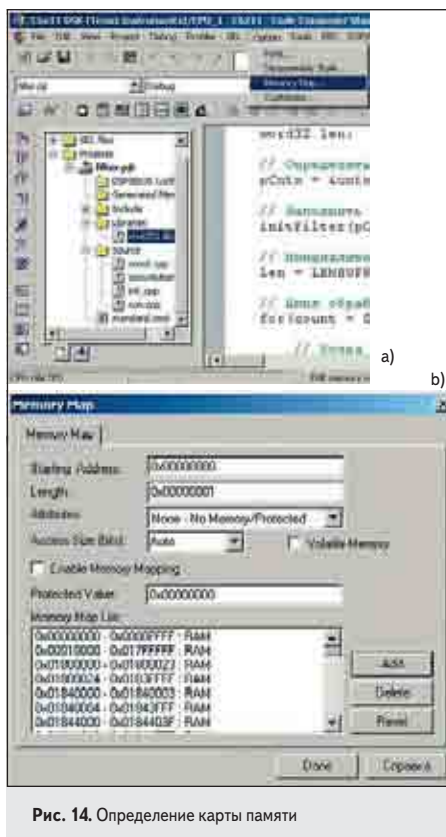


Рис. 14. Определение карты памяти

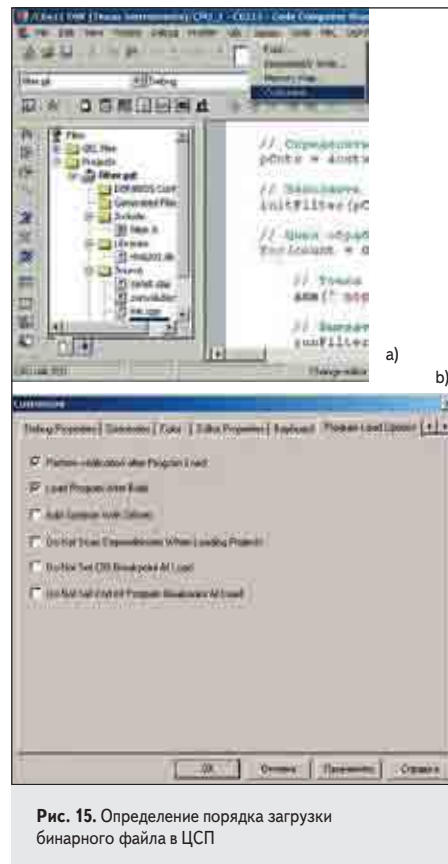


Рис. 15. Определение порядка загрузки бинарного файла в ЦСП

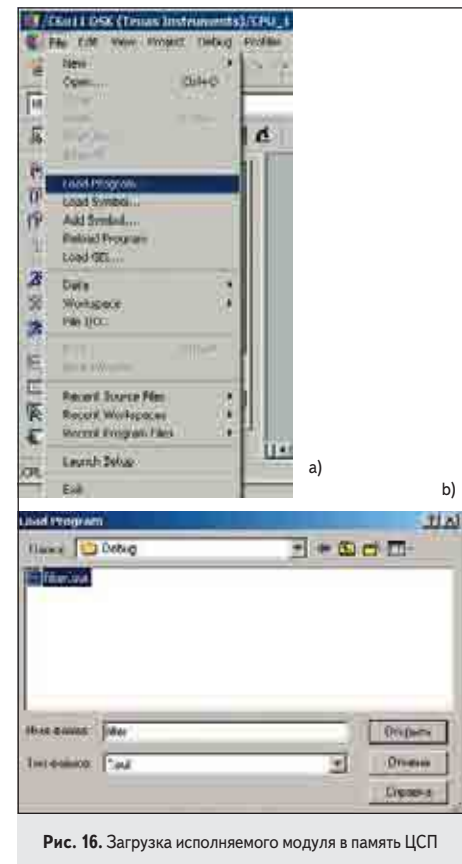


Рис. 16. Загрузка исполняемого модуля в память ЦСП

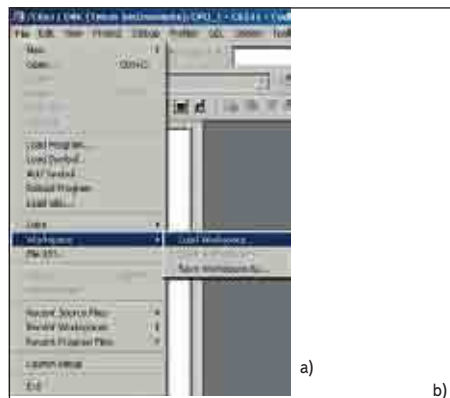


Рис. 22. Запуск проекта с сохраненными настройками интерфейса

роив интерфейс, необходимо сохранить эти настройки. Для этого в пункте *File* главного меню необходимо выбрать раздел *Workspase*, в нем пункт *Save Workspase As* и сохранить настройки рабочего пространства в нужной папке с произвольным именем (рис. 21а и 21b).

При повторном запуске ИСР производится загрузка не проекта, а рабочего пространства с сохраненными настройками интерфейса CCS (в пункте *File* войти в раздел *Workspase* и пункте *Load Workspase* выбрать нужное рабочее пространство, как это показано на рис. 22а и 22б).

Для тестирования программного кода необходимо скопировать в папку проекта файл, содержащий отсчеты входного сигнала. Его можно взять из примера в предыдущей статье. В отличие от VS, файл должен находиться в папке *Debug*. В эту же папку копи-

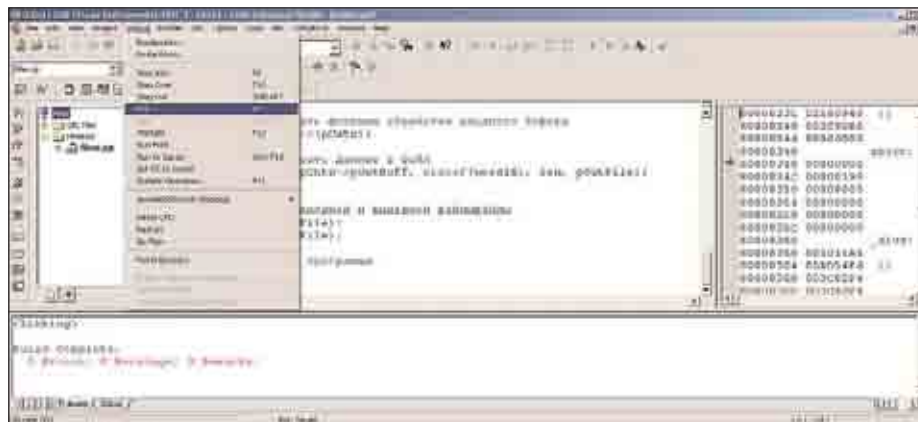


Рис. 23. Запуск программы на выполнение

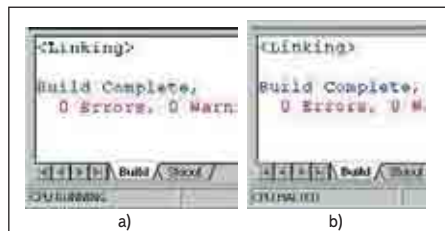



Рис. 24. Контроль работы ЦСП

ровать и выходной файл, полученный в предыдущей статье, но с другим именем, например, «outCPP.dat». Он будет эталоном при сравнении результатов работы программного кода в VS и CCS.

Запуск программы можно произвести несколькими способами:

- *первый* — через главное меню CCS: выбрать пункт *Debug*, затем — *Run* (рис. 23);
- *второй* — нажать кнопку  на кнопочной панели;
- *третий* — нажать функциональную клавишу *F5*.

Контроль работы ЦСП можно осуществлять по сообщению, выводимому в левом нижнем углу CCS (рис. 24а — процессор работает, рис. 24б — остановлен).

После остановки процессора в папке *Debug* появляется выходной файл «out.dat».

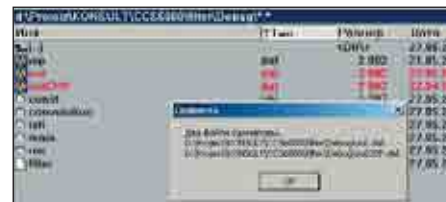


Рис. 25. Сравнение результата с эталоном

Сравнив его с эталонным файлом средствами любого менеджера окон, можно убедиться, что он идентичен файлу, полученному при тестировании программного кода в VS (рис. 25).

В следующей статье будет показано, как переписать часть программного кода на ассемблере.

Все файлы проекта можно скачать с сайта www.scanti.ru. Рекомендуется при первом запуске CCS загрузить проект, а не рабочее пространство, настроить интерфейс заново и сохранить настройки, так как параметры интерфейса учитывают конфигурацию ПК, на котором разрабатывается проект. При загрузке проекта будет выведено сообщение об ошибочном пути к библиотеке «rts6201.lib». Необходимо скорректировать путь в соответствии с размещением CCS на компьютере. ■