Тестирование программного кода для ЦСП TMS320C6000

Игорь ГУК
gii@scanti.ru

В статье рассмотрен принцип тестирования программного кода в ИСР CCS. Приведены примеры использования возможностей RTDX, визуализации процесса вычислений и контроля времени выполнения программного кода. Материал базируется на предыдущих статьях, и для его успешного освоения рекомендуется ознакомиться с ними.

Необходимо запустить CCS и создать новый проект. Напомним, что для этого в главном меню ИСР нужно выбрать раздел *Project*, в нем — пункт *New*, затем в появившемся диалоговом окне выбираются семейство ЦСП и тип проекта, а также указываются место размещения проекта на жестком диске и его имя.

В папку проекта необходимо скопировать файлы с исходным С-кодом рассмотренного в прошлой статье фильтра, за исключением файла с функцией *main*(). Скопированные файлы, кроме заголовочного файла, подключаются к проекту (в главном меню раздел *Project*, пункт *Add Files to Project*).

Затем необходимо создать новую функцию *main*(). Для этого создается новый файл (раздел *File* главного меню, пункт *New* и подпункт *Sours File*), в рабочей области текстового редактора набирается программный код функции:

```
#include «filter.h»
int main(void) {
        // Обьявление переменных
        word32 count;     // Переменная цикла

        // Определить указатель на контекстную структуру
        pCntx = &cntx;

        // Заполнить контекстную структуру
        initFilter(pCntx);

        // Цикл обработки входного буфера
        for(count = 0 ; count < 100; count++) {
                // Точка подключения зонда
                asm(« nop «);

                // Вызвать функцию обработки входного буфера
                runFilter(pCntx);

                // Точка подключения зонда
                asm(« nop «);
        }

        // Выход из программы
        return 0;
}
```

Код функции достаточно прост. В пояснении нуждается только оператор *asm*(«*nop*»), который является расширением языка С в CCS. Он служит для гарантированного наличия пустых операторов (*nop*) до и после вызова функции обработки входного буфера *runFilter*(). В данном случае пустые операторы — удобное место для подключения контрольных точек.

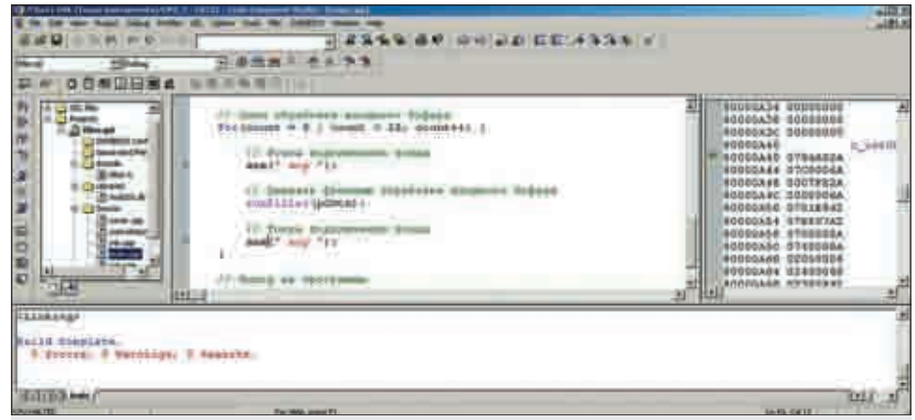

Рис. 1. Установка точек зондирования

Набранный файл с кодом функции *main*() необходимо сократить с именем, например *mainCCS.cpp*, и подключить к проекту. Провести трансляцию проекта и загрузку бинарного кода в ЦСП.

Для тестирования программы необходимо подключить входной и выходной файлы. При этом используются возможности RTDX CCS. В отличие от условий, описанных в предыдущих статьях, оба файла должны существовать. Файлы подключаются при помощи точек зондирования, которые необходимо вначале определить. Для этого необходимо установить курсор на строку кода, содержащую оператор *asm*(«*nop*»), и нажать клавишу на панели быстрых кнопок. Действие повторить для второго оператора *asm*(«*nop*»). Окно текстового редактора CCS примет вид, показанный на рис. 1.

При подключении тестирующих файлов через точки зондирования их формат должен быть изменен. Раньше это были двоичные файлы, являющиеся простой последовательностью отсчетов сигнала в двоичном представлении. В этот раз входной и выходной файлы имеют текстовый формат, где первой строкой идет заголовок, а в последующих строках представлены данные в 32-разрядном целочисленном виде. Пример таких тестовых файлов приведен в таблице. Входной файл содержит единичный импульс с амплитудой 1024, а выходной — реакцию на него (импульсную характеристику) рассматриваемого в статье примера фильтра.

Таблица. Тестовые файлы в текстовом формате

Входной файл «», содержащий единичный импульс	Выходной файл «», содержащий реакцию фильтра на единичный импульс	Комментарии
1651 1 0 0 0	1651 1 80000ca4 0 1b	Заголовок, где указаны параметры кодирования данных.
0x00000400 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000	0xFFC0001E 0x007AFFB0 0x01880188 0xFFB0007A 0x001EFFC0 0x00000000 0x00000000 0x00000000	Входные и выходные отсчеты. Представлены в целочисленном 16-разрядном формате, по два значения в каждой строке. Причем старший разряд идет первым.

На сайте **w ww .scanti.r u** можно найти консольные программы для конвертирования двоичного и текстового представления тестовых файлов для 16-разрядных целочисленных отсчетов. Программы очень просты. При запуске без параметров выводятся правила их использования. Там же находятся готовые тестовые файлы в текстовом формате, предназначенные для проекта, рассматриваемого в данной статье.

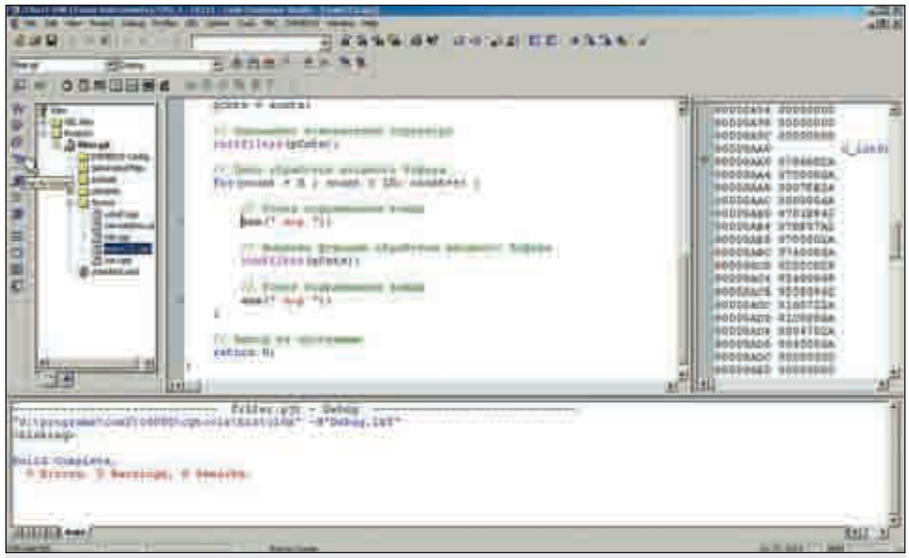

Рис. 2. Запуск программы на выполнение до курсора

Прежде чем подключить файлы, нужно выполнить программу до первого оператора *asm*(«*nop*»). Для этого устанавливают курсор на данном операторе и запускают выполнение программы до курсора нажатием клавиши на панели быстрых кнопок (рис. 2). Это необходимо сделать, чтобы произошла инициализация всех программных элементов тестируемого алгоритма.

После выполнения программы напротив курсора появится желтая стрелка. Окно текстового редактора примет вид, показанный на рис. 3.

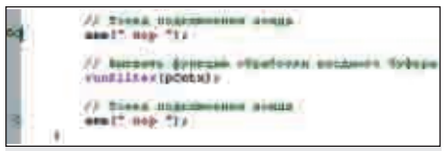

Рис. 3. Выполнение программы до курсора

В главном меню CCS выбирают пункт *File*, в нем подпункт *File* I/O (рис. 4а). Появится окно *File* I/O (рис. 4б), где нужно перейти на закладку *File Input* и нажать кнопку *Add File*. В диалоговом окне *File Input* (рис. 4в) выбрать имя входного файла и подтвердить ввод.

В окне *File* I/O установить флажок в позиции *Wrap Around* для циклического считывания данных из входного файла (рис. 5а). Здесь также необходимо указать адрес приемного буфера (в окне *Address* указать имя входного буфера через указатель на контекстную структуру) и количество считываемых из файла отсчетов (в окне *Length* указать длину буфера также через указатель на контекстную структуру). Отметим, что считывание данных из подключенного файла происходит по 32 разряда, поэтому в рассматриваемом примере длина буфера делится на два, так как в качестве входного и выходного сигналов используются 16-разрядные отсчеты. Кроме этого, необходимо, чтобы длина входного буфера была четной. Для выполнения данного условия нужно скорректировать макрос в заголовочном файле:

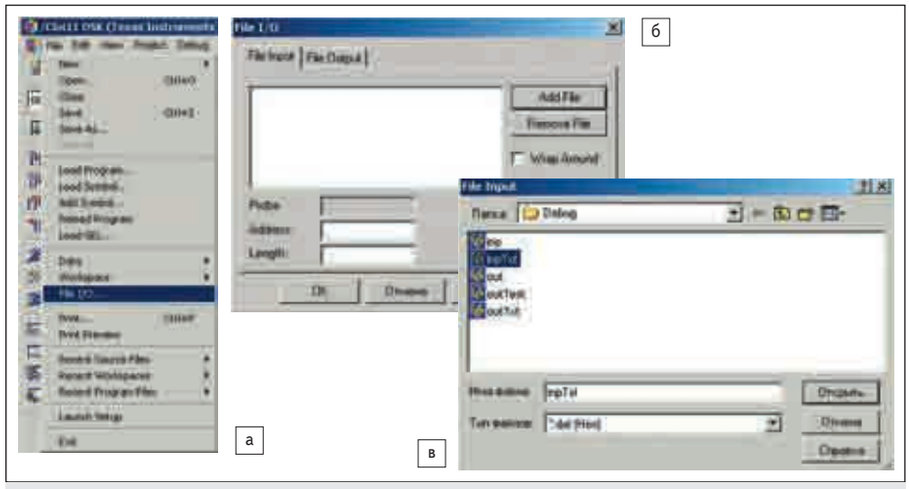

Рис. 4. Подключение тестовых файлов к проекту

```
#define LENBUFF  56   // Длина входного и выходного буферов.
```

Нажать клавишу *Add Probe Point* и перейти в окно *Break/Probe Points* (рис. 5б). Перейти на закладку *Probe Points*. Выбрать точку зондирования для подключения входного файла в списке *Probe Point*. Ее положение появится в позиции *Location*. В выпадающем списке *Connect To* указать имя тестового файла для подключения к выбранной точке. Нажать клавишу *Replace* для связи точки зондирования с файлом. Подтвердить ввод (нажать клавишу *OK*). В диалоговом окне *File* I/O в позиции *Probe* появится подтверждение успешного подключения файла (слово *Connected*). Перейти на закладку *File Output* (рис. 5в) и повторить аналогичную процедуру для выходного файла (выходной файл должен существовать!).

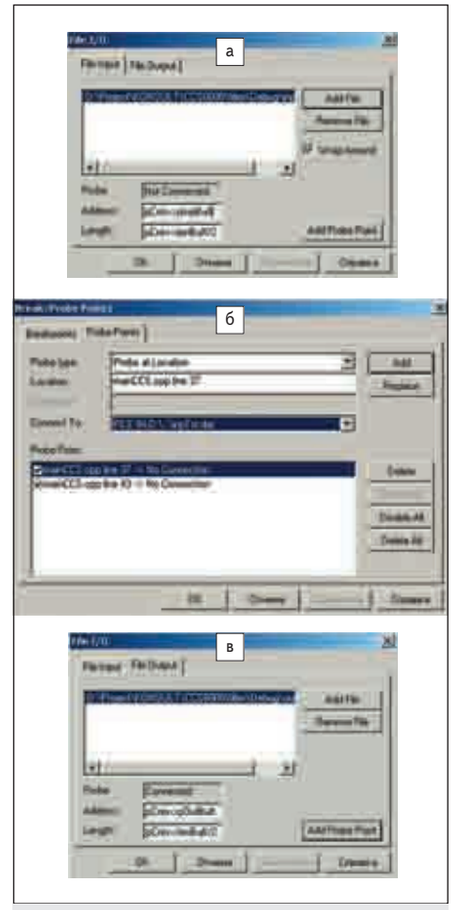

Рис. 5. Настройка подключения файлов

В процессе выполнения указанных выше действий должны появиться два окна (рис. 6) — это счетчики входного и выходного файлов. В них отображается процесс чтения входного и запись выходного файлов.

Рис. 6. Счетчики файлов

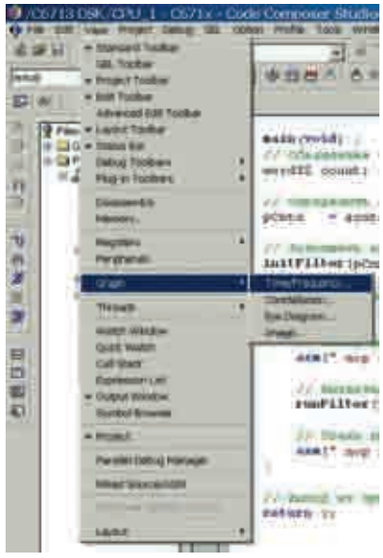

Рис. 7. Запуск окна визуализации данных

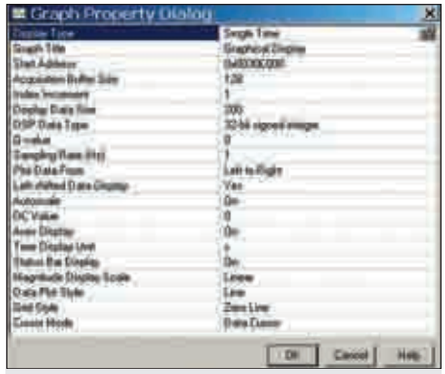

Рис. 8. Окно настройки визуализации данных

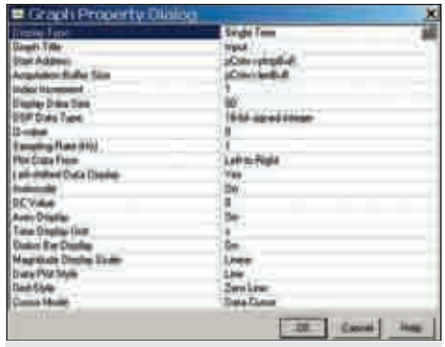

Рис. 9. Параметры визуализации данных для осциллограммы

Сохранить настройки рабочей среды (*File→Workspace→Save Workspace*), провести компиляцию и загрузку программного кода (нажав клавишу), запустить программу на выполнение (нажав клавишу). В результате получим выходной тестовый файл в текстовом формате. Для рассматриваемого примера несколько первых строчек этого файла показаны во второй колонке таблицы.

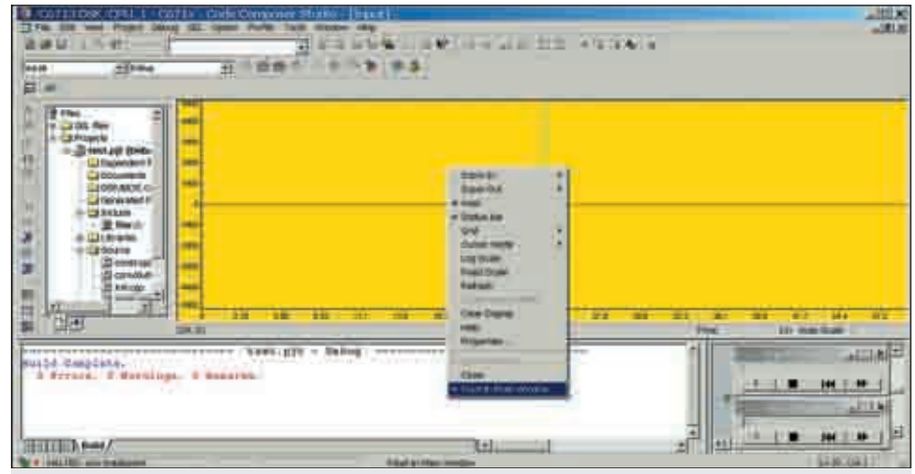

Рис. 10. Вид CCS после запуска окна визуализации данных

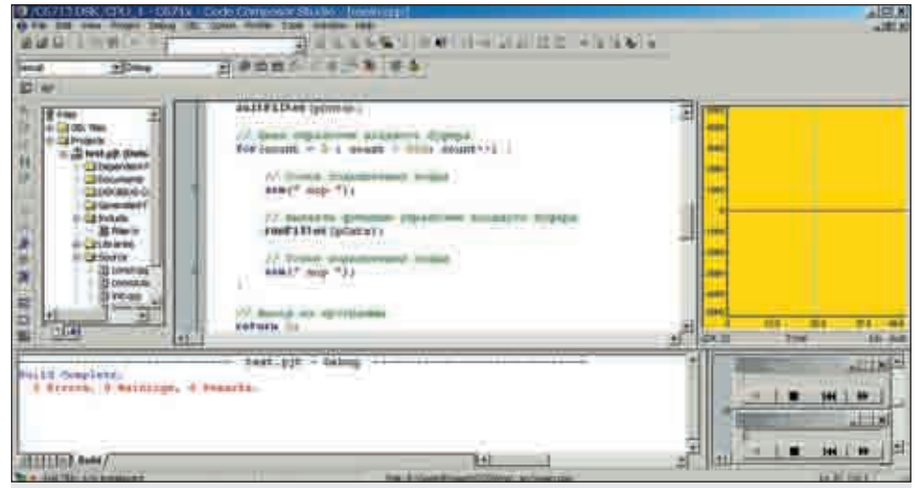

Рис. 11. Вид CCS после минимизации окна визуализации данных

Проведя преобразование выходного текстового файла в выходной бинарный файл и сравнив результат с тестовым файлом, полученным в VS, можно убедиться в их идентичности.

Для повторного запуска программы необходимо сбросить счетчики файлов, нажав клавишу на каждом счетчике (при этом выходной файл будет создан заново!).

Использование возможностей RTDX в данном случае позволяет значительно сократить программный код за счет исключения из него функций файлового ввода/вывода в стиле языка C, а также повысить скорость выполнения программы в режиме отладки.

Для облегчения процесса отладки CCS имеет возможность отображать отдельные области памяти в виде осциллограммы, изображения, спектра и т. д.

Рассмотрим, как реализовать отображение обрабатываемого сигнала в виде осциллограммы. Вначале необходимо запустить окно визуализации данных. Для этого в главном меню следует выбрать пункт *View*, в нем опцию *Graph->Time/Frequency*…(рис. 7).

В результате появится окно, где требуется ввести параметры отображения данных (рис. 8). В графе *Graph Title* можно указать наименование отображаемой области. В графе *Start Address* указывается начальный адрес отображаемых данных. В примере указан адрес входного буфера через контекстную структуру — *pCntx->pInpBuff*. В графе *Acquisition Buffer Size* — указывается размер отображаемого буфера (в примере это *pCntx->lenBuff*). Необходимо также указать количество отображаемых отсчетов (не превышающих размера буфера) в графе *Display Data Size*. Для примера выбрано значение 50. Все остальные параметры отображения можно оставить без изменений.

После заполнения указанных выше граф окно примет вид, показанный на рис. 9. Необходимо подтвердить ввод значений, нажав кнопку *OK*. После этого CCS примет вид, показанный на рис. 10. Окно визуализации данных занимает все рабочую область. Это не всегда удобно.

Рекомендуется выводить отображаемые данные в отдельном окне. Для этого нужно кликнуть правой кнопкой мыши в рабочей облас-

Рис. 12. Определение положения точки остановки

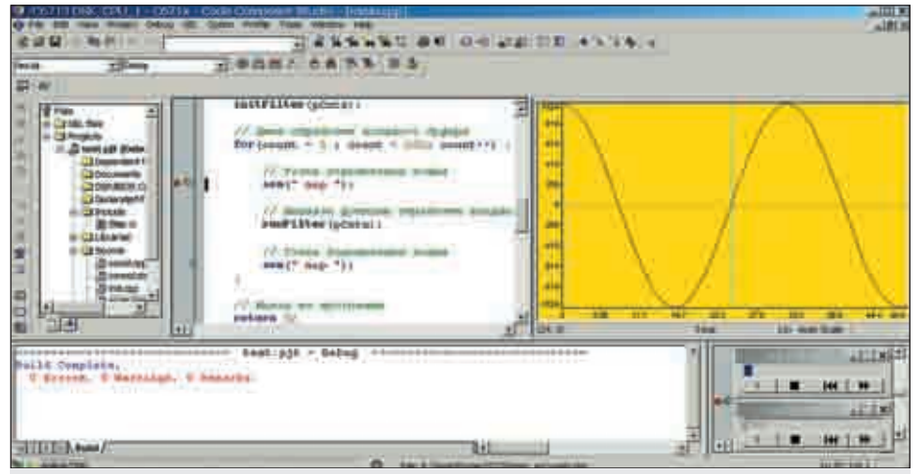

Рис. 13. Выполнение программного кода в режиме animating

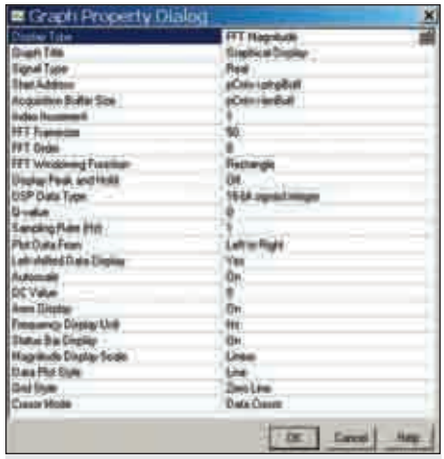

Рис. 14. Параметры визуализации данных для спектрограммы

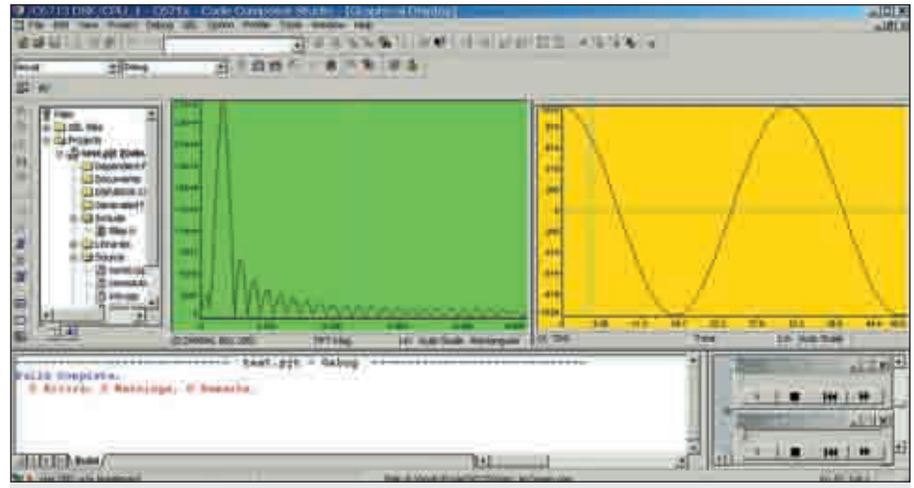

Рис. 15. Вид CCS при отображении осциллограммы и спектрограммы

ти и из контекстного меню выбрать опцию *Float Main In Window* (рис. 10). Затем CCS примет вид, показанный на рис. 11.

Изменения данных в окне визуализации происходит при остановке процесса выполнения программы. Для реализации данного положения нужно установить точку остановки в процессе выполнения программного кода. Это делается очень просто. Нужно установить курсор в том месте программы, где необходимо прервать ее выполнение. Затем нажать кнопку на панели быстрых клавиш. Напротив выбранного места остановки появится красная точка (рис. 12).

Далее надо откомпилировать проект, нажав кнопку . И запустить программу на выполнение в режиме *animating*, нажав кнопку . Вид CCS после выполнение всех действий показан на рис. 13.

Теперь рассмотрим, как реализовать отображение обрабатываемого сигнала в виде спектрограммы. Необходимо включить визуализацию данных (рис. 7). В появившемся окне (рис. 8) в графе *Display Type* указать *FFT Magnitude*. Остальные графы заполняются аналогично рассмотренному выше примеру отображения осциллограммы. В графе *Graph Title* — наименование отображаемой области. В графе *Start Address* — начальный адрес отображаемых данных. В графе *Acquisition Buffer Size* — размер отображаемого буфера. В графе *Display Data Size* — количество отображаемых отсчетов. Вид окна с установленными параметрами показан на рис. 14.

После подтверждения ввода параметров (нажатие кнопки *OK*) CCS примет вид, показанный на рис. 15.

Программой код не изменялся. Поэтому нет необходимости проводить его компиляцию. Запустить программу на выполнение без перезагрузки очень просто. Нужно установить программный указатель на начало программного кода, для чего в главном меню выбрать пункт *Debug*, а в нем опцию *Restart* (рис. 16). Программа готова к повторному запуску. Напомним, что при использовании визуализации данных в коде имеются точки остановки, поэтому запускать программу необходимо в режиме *animating* (нажать кнопку).

Если при запуске программы на выполнение окно визуализации исчезло, тогда его можно вернуть через главное меню CCS: *Window->Graphical Display* (рис. 17). Как уже отмечалось, производить визуализацию в главном окне неудобно. Кликнув правой кнопкой мыши в рабочей области и выбрав опцию *Float Main In Window* (рис. 18), мож-

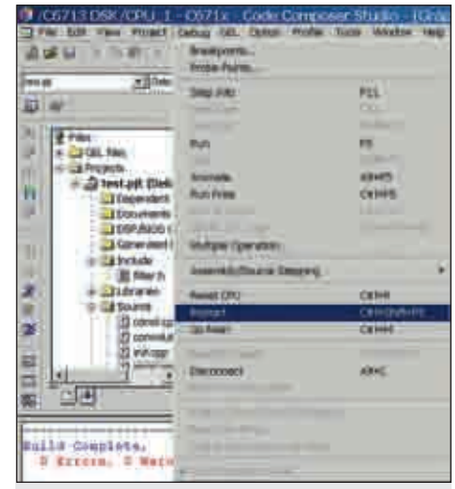

Рис. 16. Повторный запуск программного кода

Рис. 17. Вызов окна визуализации данных

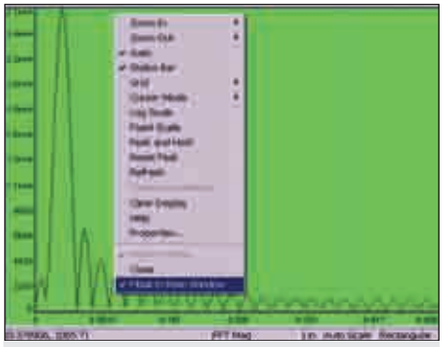

Рис. 18. Перевод визуализации данных из рабочей области в отдельное окно

но перевести визуализацию данных в отдельное окно (рис. 19).

Запустив программу на выполнение (напомним, что необходимо установить программный указатель на начало кода и запустить выполнение программы в режиме *animating*), можно наблюдать изменение входного сигнала и на осциллограмме и на спектрограмме.

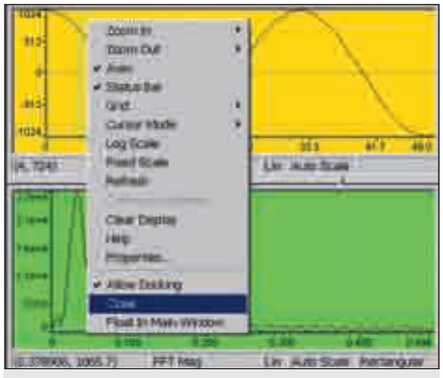

Рис. 20. Закрытие окна визуализации данных

Рис. 21. Открытие окна параметров профилирования программного кода

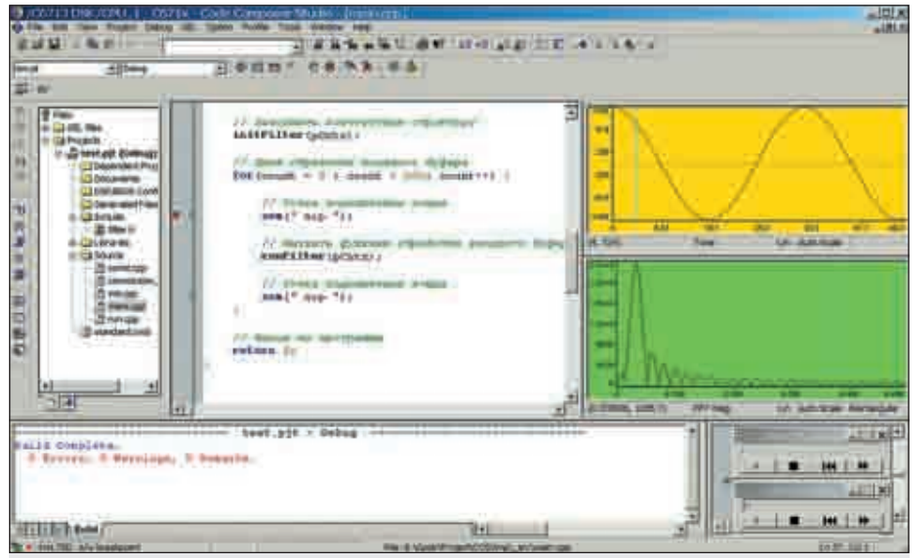

Рис. 19. CCS при визуализации входных данных в виде осциллограммы и спектрограммы

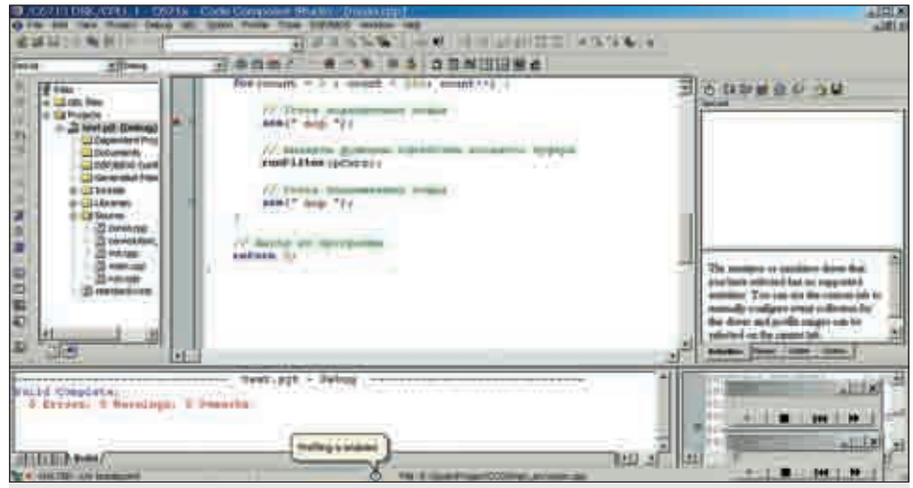

Рис. 22. Вид CCS с окном настройки режима профилирования программного кода

В дальнейшем визуализация не понадобится, и ее необходимо отключить. Соответственно, нужно кликнуть правой кнопкой мыши в окне визуализации и в контекстном меню выбрать опцию *Close* (рис. 20).

Для оценки производительности в CCS реализован механизм профилирования (определения времени выполнения) отдельных участков кода и функций.

Вначале необходимо открыть окно настройки параметров профилирования. Для этого в главном меню CCS выбрать пункт *Profile*, а в нем опцию *Setap* (рис. 21). В CCS появится окно настройки профилирования (рис. 22). Далее нажатием кнопки ⚙ в окне настройки профилирования включается режим профилирования (рис. 22).

Перейти на вкладку *Ranges* (рис. 23) и перетащить мышкой функции, для которых будет производиться профилирование, из папки *Disabled* в папку *Enabled* (рис. 22). В примере профилирование производится для функций *runFilter*() и *convolution*(). На вкладке

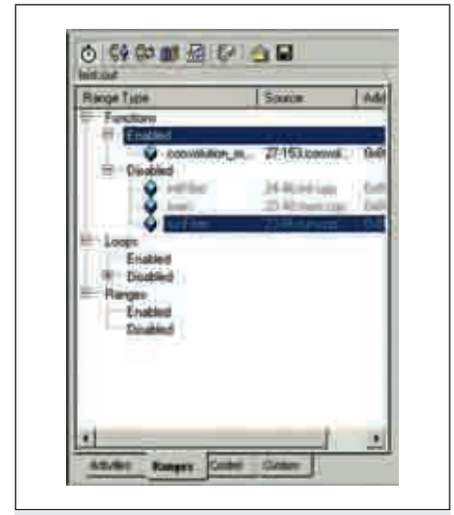

Рис. 23. Выбор функций для профилирования

Custom выбрать размерность профилирования. В рассматриваемом примере профили-

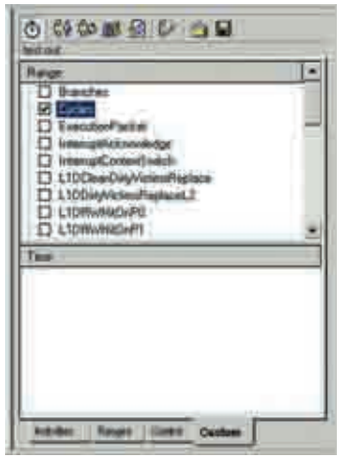

Рис. 24. Выбор размерности профилирования

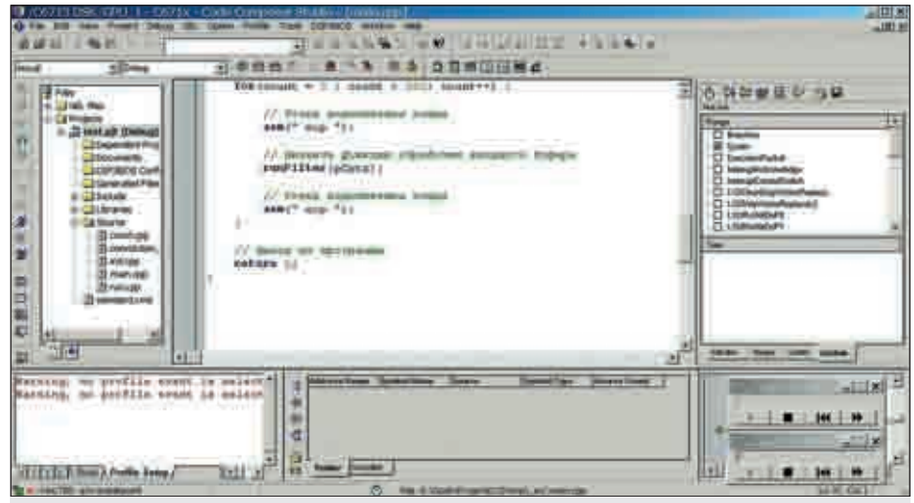

Рис. 26. Вид CCS в режиме профилирования

Рис. 25. Включение окна отображения результатов профилирования

рование будет производиться в тактах процессора. Для этого ставится галочка напротив позиции *Cycles* (рис. 24).

Необходимо отобразить окно, в котором указаны параметры профилирования. Это можно сделать через главное меню CCS: *Profile->Viewer* (рис. 25).

После этого CCS примет вид, показанный на рис. 26.

Можно выбрать параметры, которые будут отображаться в окне профилирования. Для этого надо кликнуть правой кнопкой мыши в этом окне и из контекстного меню выбрать опцию *Columns And Rows Setting* (рис. 27).

Затем в появившемся окне *Columns And Rows Setting* выбрать необходимые параметры отображения (рис. 28).

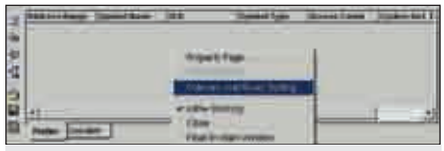

Рис. 27. Настройка окна отображения процесса профилирования

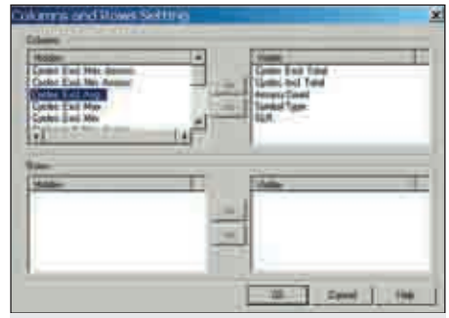

Рис. 28. Окно выбора отображаемых параметров профилирования

Теперь можно откомпилировать проект (нажать кнопку) и запустить программный код на выполнение (нажать кнопку). Окно отображения процесса профилирования изменится и будет выглядеть так, как показано на рис. 29.

На этом краткий обзор профилирования закончен. В дальнейшем изложении оно не понадобится, поэтому его необходимо выключить. Это производится следующим образом. Выключить режим профилирования, нажав кнопку в окне настройки режима профилирования и закрыть это окно, кликнув правой кнопкой мыши в области окна и выбрав в контекстном меню опцию *Hide* (рис. 30).

Закрыть панель профилирования, кликнув правой кнопкой мыши и выбрав *Close* (рис. 31).

Для дальнейшего изложения необходимо закрыть проект и привести внешний вид CCS в исходное состояние. Следует выбрать в главном меню пункт *Project*, а в нем опцию *Close* (рис. 32), закрыть все файлы в окне редактора и закрыть счетчики входного и выходного файлов.

Теперь можно приступить к созданию нового проекта, который сгенерирует библио-

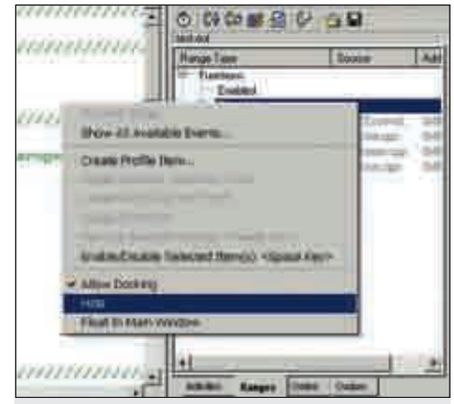

Рис. 30. Выключение режима профилирования

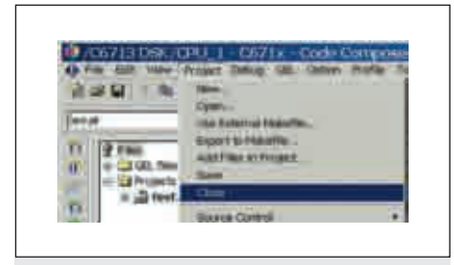

Рис. 32. Закрытие проекта

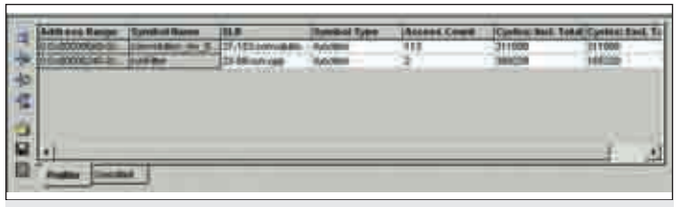

Рис. 29. Результат профилирования выбранных функций

Рис. 31. Закрытие окна отображения процесса профилирования

Рис. 33. Создание проекта библиотеки

теку с разработанным ранее алгоритмом фильтрации. Все действия по созданию проекта аналогичны тем, что описаны в предыдущих статьях цикла. Необходимо только указать тип проекта — «библиотека» (рис. 33). Кроме этого, при создании проекта библиотеки не нужно подключать управляющий файл (файл с расширением *.cmd*) и стандартную библиотеку.

После создания проекта библиотеки можно произвести настройку опций компилирования. Например, вызвав меню настройки опций компилирования (рис. 34), на вкладке *Archives* можно указать папку для размещения созданной библиотеки (рис. 35).

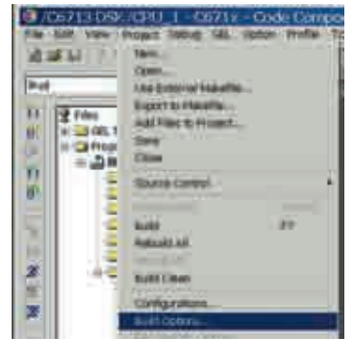

Рис. 34. Вызов окна редактирования опций компилирования

К созданному проекту библиотеки необходимо подключить файлы с исходным кодом, за исключением файла с функцией *main*(). Окно менеджера проекта примет вид, показанный на рис. 36.

Проведите компилирование проекта, нажав клавишу . В папке *_lib* появится файл библиотеки *lib.lib*. Закройте проект библиотеки.

Загрузите тестовый проект, созданный в начале статьи. Из файлов с исходным текстом кода оставьте только файл с функцией *main*(), все остальные — удалите из проекта. Подключите к проекту созданную библиотеку. Окно менеджера проекта после всех описанных действий примет вид, показанный на рис. 37.

Откомпилируйте проект и запустите его на выполнение. Сравните полученный результат с тестовым файлом. Убедитесь в их идентичности.

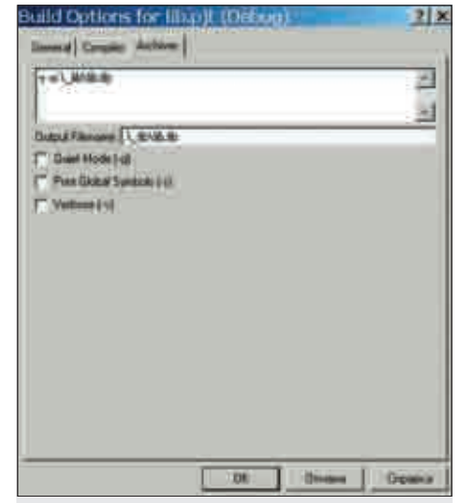

Рис. 35. Пример настройки опций компилирования

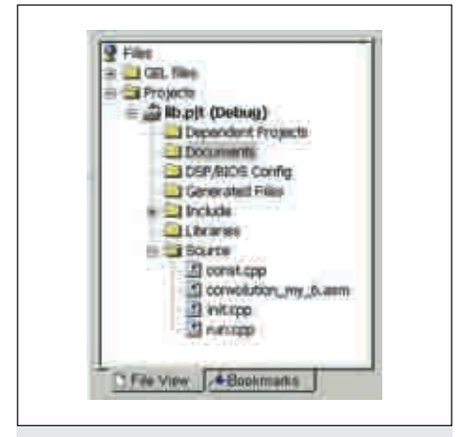

Рис.36. Вид менеджера проекта при создании библиотеки

Рис. 37. Вид менеджера проекта при тестировании библиотеки

В следующих статьях цикла по программированию DSP семейства TMS320C6000 будут рассмотрены вопросы использования DSP/BIOS, настройки периферийных модулей, использование прерываний, каналов DMA и т. д.