ЗАО "Спрут Технология"

Руководство пользователя по генератору постпроцессоров

423812, г. Набережные Челны,
а/я 438,
ЗАО СПРУТ-Технология

Содержание

1 Введение

1.1 Назначение генератора постпроцессоров

Генератор постпроцессоров является приложением для операционных систем семейства Windows.

Назначение генератора постпроцессоров – разработка файлов настройки постпроцессора на различные системы ЧПУ. Файл настройки постпроцессора на конкретную систему ЧПУ используется исполняющей системой для формирования соответствующей управляющей программы.

Для разработки файла настройки постпроцессора необходимо выполнить следующие действия:

- заполнить данные о станке и системе ЧПУ;
- описать структуру и формат кадра (сформировать список регистров);
- написать шаблоны или программы обработки технологических команд;
- сохранить файлы настройки постпроцессора.

Возможно как создание новых файлов настройки, так и редактирование существующих.

В среде генератора постпроцессоров кроме работы с данными о станке и системе ЧПУ, списком регистров и программами обработки команд, также возможны просмотр файлов технологических команд и тестовая генерация управляющих программ.

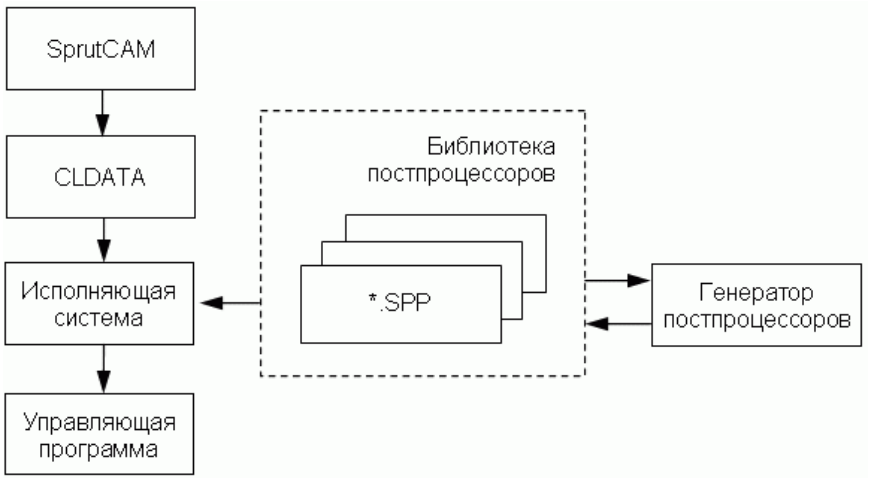
1.2 Набор файлов генератора постпроцессоров

- **Inp.exe** – исполняемый модуль генератора постпроцессоров.
- **InpD.dll** – исполняющая система постпроцессора.
- **CLDImport.dll** – библиотека импорта проектов SprutCAM.
- ***.spp** – файлы исходных текстов программ обработки технологических команд и настройки постпроцессора на конкретную систему ЧПУ.
- ***.stcx** – файлы проектов системы SprutCAM в формате XML.
- ***.stc** – файлы проектов системы SprutCAM.
- ***.mcd** – временные файлы связанные с проектом SprutCAM, содержат информацию о технологических командах.
- ***.inp**, ***.ppp** – файлы настройки для предыдущей версии постпроцессора.

2 Общая организация работы

2.1 Принцип работы генератора постпроцессоров

При помощи генератора постпроцессоров разрабатываются файлы настройки постпроцессора на различные системы ЧПУ (файлы с расширением ***.SPP**, так же возможен импорт файлов предыдущей версии постпроцессора – ***.INP**, ***.PPP**). В файле настройки постпроцессора содержится описание всех особенностей составления управляющих программ для указанной системы ЧПУ. Это описание используется исполняющей системой постпроцессора при формировании управляющей программы из файлов технологических команд (***.MCD**, ***.STC**, ***.STCX** файлов), которые, в свою очередь, могут быть сгенерированы, например, системой SprutCAM.



Для разработки файла настройки постпроцессора необходимо заполнить данные о станке и системе ЧПУ, описать структуру и формат кадра, а так же заполнить шаблоны и (или) написать программы обработки технологических команд.

Под данными о станке и системе ЧПУ понимается название станка и системы ЧПУ, максимальные перемещения по координатам и некоторые дополнительные данные.

Структура и формат кадра определяются упорядоченной последовательностью регистров и их параметрами. Регистры должны быть расположены в том порядке, в котором их идентификаторы и значения должны выводиться из программы обработки технологической команды в кадр управляющей программы.

Шаблоны включают в себя перечень регистров, составленных в требуемом порядке, необходимом для вывода в кадр управляющей программы для соответствующей технологической команды.

Программы обработки технологических команд пишутся на специальном проблемно-ориентированном языке и могут

содержать математические выражения и функции, операторы ввода/вывода, условные операторы, циклы, операторы перехода, вызовы подпрограмм, операторы формирования кадров управляющей программы и операторы работы с файлом технологических команд.

В среде генератора постпроцессоров производится:

- заполнение данных о станке и системе ЧПУ;
- описание структуры и формата кадра;
- формирование масок;
- написание программ обработки команд, их трансляция и отладка;
- просмотр файлов технологических команд;
- тестовая генерация управляющих программ.

Заполнение данных о станке и системе ЧПУ, описание структуры и формата кадра, формирование масок, написание программ обработки команд, их трансляция и отладка производятся в среде генератора постпроцессоров, где также возможны просмотр файлов технологических команд и тестовая генерация управляющих программ.

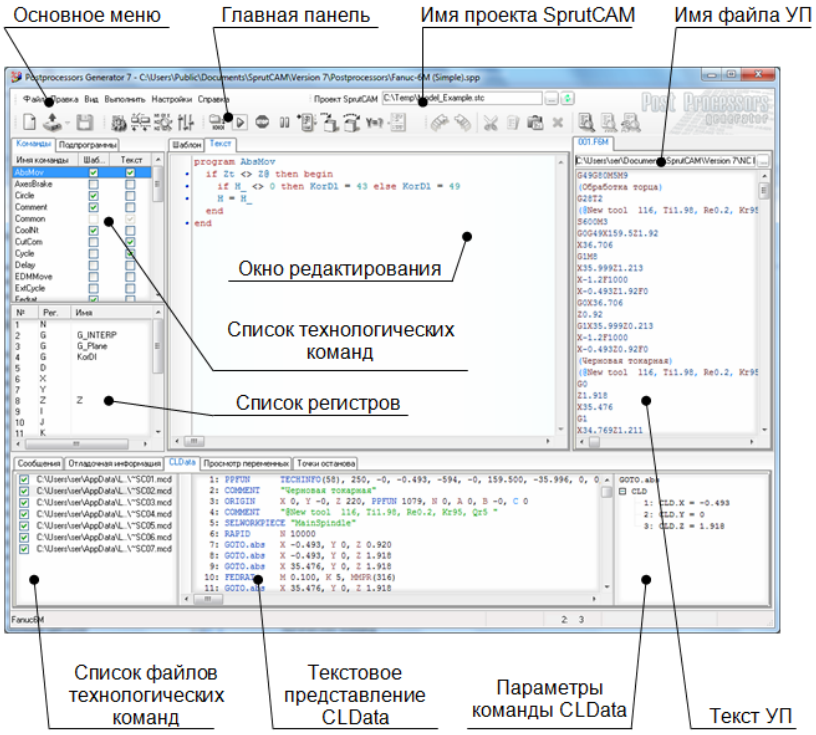
Исходные тексты программ обработки технологических команд, шаблоны, данные о станке и системе ЧПУ, маски, список и формат регистров сохраняются в файле с именем конкретного постпроцессора и расширением ***.SPP**. Данные из этого файла используются исполняющей системой постпроцессора при формировании управляющей программы для соответствующей системы ЧПУ.

Исполняющая система постпроцессора считывает данные о процессе обработки детали из файла технологических команд, анализирует код команды и если включена программа то передает управление в программу обработки этой команды (имя шаблона и программы обработки совпадает с именем команды, которую она обрабатывает), далее если включен шаблон, то формирует кадр управляющей программы по соответствующему шаблону. Параметры технологической команды передаются через предопределенный массив <**CLD**> и оператор <**CMD**>. При выполнении вызванной программы могут быть изменены значения регистров и внутренних переменных и сформированы кадры управляющей программы.

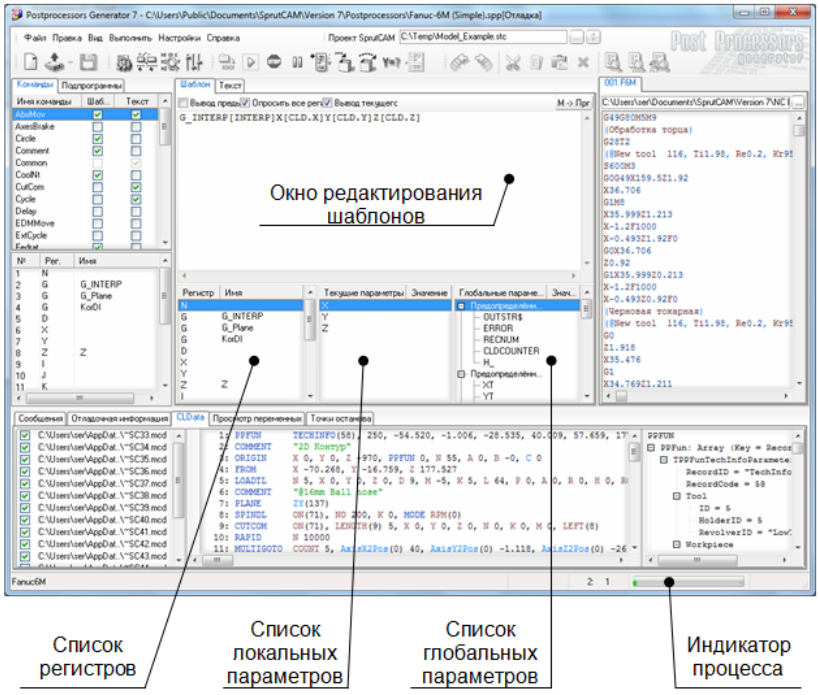
В программах обработки команд формирование кадра управляющей программы производится соответствующими операторами, причем в кадр выводятся идентификаторы и значения только тех регистров, которые были изменены после формирования предыдущего кадра.

2.2 Главное окно

Главное окно системы имеет вид:



В верхней части окна расположены <**Основное меню**> и < **Главная панель**>. Слева – список программ обработки команд и список регистров. В центре – переключающиеся страницы с редактором шаблонов и редактором исходного текста программ обработки технологических команд. На закладке <**Шаблон**>, кроме редактора, расположены списки регистров, локальных и глобальных параметров. Справа находится окно управляющей программы. Внизу – переключающиеся окна системных сообщений, отладочной информации, список файлов траектории движения инструмента и окно текстового представления этих файлов, перечня контролируемых переменных и список точек останова.



2.2.1 Основное меню

Основное меню состоит из шести пунктов. Некоторые пункты меню продублированы на главной панели и контекстных меню соответствующих окон.

- <**Файл**>
  - <**Создать**> – создание нового файла настройки постпроцессора. Перед созданием автоматически обновляется состояние системы: закрываются открытые файлы постпроцессора, очищается окно управляющей программы, меняется расширение файла управляющей программы. Функция также может быть вызвана из главной панели.
  - <**Открыть**> – открытие ранее сохраненного файла настройки постпроцессора. Перед открытием производится обновление состояния системы, очищается окно управляющей программы. Пункт меню продублирован в главной панели.
  - <**Открыть повторно**> – открытие файла настройки постпроцессора из списка ранее загруженных.
  - <**Сохранить**> – сохранение файла настройки постпроцессора под текущим именем. Если файл ранее не сохранялся, то запрашивается его имя. Вызов функции возможен и из главной панели.
  - <**Сохранить как**> – сохранение файла настройки постпроцессора под новым именем.
  - <**Открыть проект SprutCAM**> – открытие файлов технологических команд из проекта SprutCAM. Пункт меню

продублирован в главной панели.

- <**Выход**> – выход из генератора постпроцессоров. Если открытый постпроцессор не был сохранен, то будет предложено его сохранить.

- <**Правка**> – все функции этого меню доступны и из главной панели.
  - <**Отменить**> – отмена последнего исправления.
  - <**Вернуть**> – возврат последнего исправления.
  - <**Вырезать**> – перемещение выделенного текста или регистра в буфер обмена.
  - <**Копировать**> – копирование выделенного текста или регистра в буфер обмена.
  - <**Вставить**> – вставка ранее скопированного текста из буфера обмена.
  - <**Удалить**> – удаление выделенного текста.
  - <**Найти**> – поиск слова в тексте программы.
  - <**Найти далее**> – поиск следующего слова в тексте программ обработки технологических команд.
  - <**Заменить**> – поиск и замена введенного слова в тексте программ обработки технологических команд.
- <**Вид**>
  - <**Сообщения**> – показать окно системных сообщений.
  - <**Отладочная информация**> – показать окно отладочной информации.
  - <**CLData**> – показать список файлов технологических команд и их текстовое представление.
  - <**Просмотр переменных**> – показать перечень контролируемых переменных со значениями (значения переменных отображаются только если генератор постпроцессоров находится в режиме отладки).
  - <**Точки останова**> – показать окно со списком точек останова.
  - <**Общие параметры**> – открытие окна редактирования данных о станке и системе ЧПУ. Открытие окна возможно из главной панели.
  - <**Свойства регистра**> – открытие окна редактирования свойств регистра. Пункт меню продублирован в главной панели.
  - <**Обратная интерпретация**> – открытие окна редактирования данных для обратной интерпретации управляющей программы. Открытие окна возможно и из главной панели.
- <**Выполнить**> – все функции этого меню доступны и из главной панели.
  - <**Транслировать**> – трансляция программ обработки команд.
  - <**Выполнить**> – запуск формирования управляющей

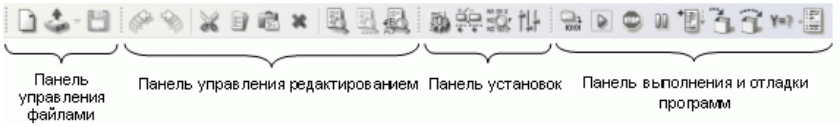
программы из файлов траектории инструмента. Если программы обработки команд не оттранслированы, то сначала производится их трансляция.

- <**Выполнить до курсора**> – выполнить программу обработки технологических команд до курсора.
- <**Шагнуть в**> – выполнить следующую команду программы обработки технологических команд с заходом в подпрограммы.
- <**Шагнуть через**> – выполнить следующую команду программы обработки технологических команд без захода в подпрограммы.
- <**Пауза**> – формирование управляющей программы.
- <**Прервать выполнение**> – прервать формирование управляющей программы.
- <**Контроль переменной**> – вычислить значение переменной или выражения. Вычисление возможно только в режиме отладки.
- <**Добавить переменную**> – добавить контролируемую переменную в список.
- <**Добавить точку останова**> – добавить точку останова в программе обработки технологических команд или файлах CLData.

- <**Настройки**> – функции этого меню доступны и из главной панели.
  - <**Каталоги**> – настройка путей к часто используемым каталогам.
  - <**Настройки редактора**> – настройка шрифтов и подсветки синтаксиса редактора программ, CLData, управляющей программы.
- <**Справка**>
  - <**Содержание**> – вывод оглавления справочной системы.
  - <**Справка**> – запуск справочной системы.
  - <**Учебник**> – запуск учебника по генератору постпроцессоров.
  - <**Написать письмо в службу технической поддержки**> – подготовка электронного письма для службы технической поддержки дилера.
  - <**Посетить сайт компании СПРУТ-Технология**> – загрузка WEB-страницы компании "СПРУТ-Технология". Адрес сервера: http://www.sprut.ru.
  - <**Написать письмо в компанию СПРУТ-Технология**> – подготовка электронного письма для службы технической поддержки компании "СПРУТ-Технология", e-mail: support@sprut.ru.
  - <**О программе**> – информация о генераторе постпроцессоров.

2.2.2 Главная панель

Главная панель расположена в верхней части экрана и состоит из четырёх инструментальных панелей:



- Панель управления файлами:
  -  – Создание нового файла настройки постпроцессора. Перед созданием обновляется состояние системы.
  -  – Открытие ранее сохраненного файла настройки постпроцессора. Перед открытием производится обновление состояния системы.
  -  – Сохранение файла настройки постпроцессора под текущим именем. Если постпроцессор ранее не сохранялся, то запрашивается его имя.

- Панель управления редактированием:
  -  – Отменить последнее изменение, сделанное в тексте программы обработки технологической команды или шаблона.
  -  – Вернуть последнее изменение, сделанное в тексте программы обработки технологической команды или шаблона.
  -  – Вырезать (переместить) выделенный текст в буфер обмена.
  -  – Копировать выделенный текст в буфер обмена.
  -  – Вставить текст из буфера обмена.
  -  – Удалить выделенный текст.
  -  – Поиск введённого сочетания символов в тексте программы.
  -  – Продолжить поиск.
  -  – Поиск и замена введённого сочетания символов в тексте программы.

- Панель установок:
  -  – Открытие окна редактирования данных о станке и системе ЧПУ.
  -  – Открытие окна определения регистров.
  -  – Открытие окна обратной интерпретации управляющей программы.
  -  – Открытие окна системных установок.

- Панель выполнения и отладки программ:
  -  – Запуск трансляции программ обработки команд.
  -  – Запуск формирования управляющей программы из файлов траектории инструмента. Если необходимо, то автоматически производится трансляция программ обработки команд.
  -  – Прервать формирование управляющей программы.
  -  – Остановить формирование управляющей программы.
  -  – Произвести формирование управляющей программы до текущей позиции в программе обработки технологической команды.
  -  – Выполнить следующую команду программы обработки технологических команд с заходом в подпрограмму.
  -  – Выполнить следующую команду программы обработки технологических команд без захода в подпрограмму.
  -  – Вычислить значение переменной или выражения. Вычисление возможно только в режиме отладки.
  -  – Добавить точку останова в программе обработки технологических команд или файлах CLData.

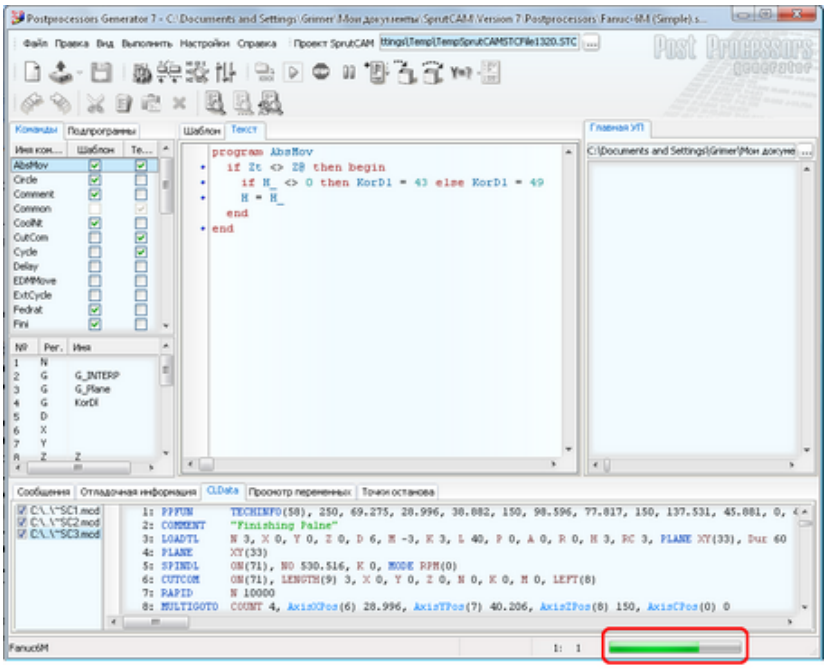
- 

– Загрузка файлов технологических команд из проекта SprutCAM.
  -  – Загрузка из системы SprutCAM обновленного списка

файлов CLData для открытого там проекта

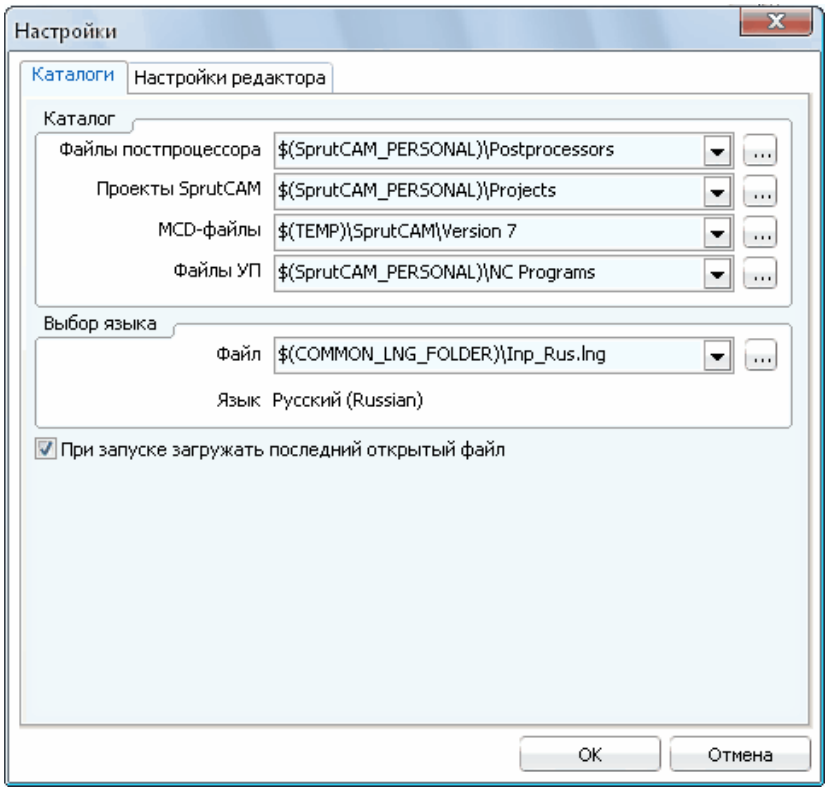
2.2.3 Индикатор процесса

Индикатор процесса появляется при выполнении системой длительных операций. К таким операциям относятся формирование управляющей программы, преобразование файла технологических команд в текстовый вид и др.



2.2.4 Системные установки

Окно системных установок вызывается нажатием кнопки  на главной панели или выбором пунктов **Установки** –> **Каталоги** в главном меню. В окне настраиваются пути по умолчанию для каталогов хранения файлов системы и выбор файла языковой поддержки.



Из каталога <**Файлы постпроцессора**> по умолчанию загружаются файлы постпроцессоров для различных систем ЧПУ (файлы ***.SPP**). Сохранение файлов постпроцессоров под новым именем по умолчанию производится в этом же каталоге.

Из каталога <**Проекты SprutCAM**> по умолчанию загружаются проекты системы SprutCAM (***.STC**). При открытии проекта создаются временные файлы технологических команд (***.MCD**) в соответствии с путями, указанными в этом проекте.

В папку <**MCD-файлы**> записываются временные файлы технологических команд (***.MCD**). После завершения работы все временные файлы удаляются.

В каталог <**Файлы УП**> по умолчанию сохраняются сгенерированные управляющие программы.

На панели <**Выбор языка**> находится панель выбора текущего языка.

Пути можно корректировать как вручную, так и при помощи диалогов выбора путей, которые запускаются нажатием кнопки ... .

В системе имеется предопределенная переменная, которая может быть использована при задании соответствующих каталогов:

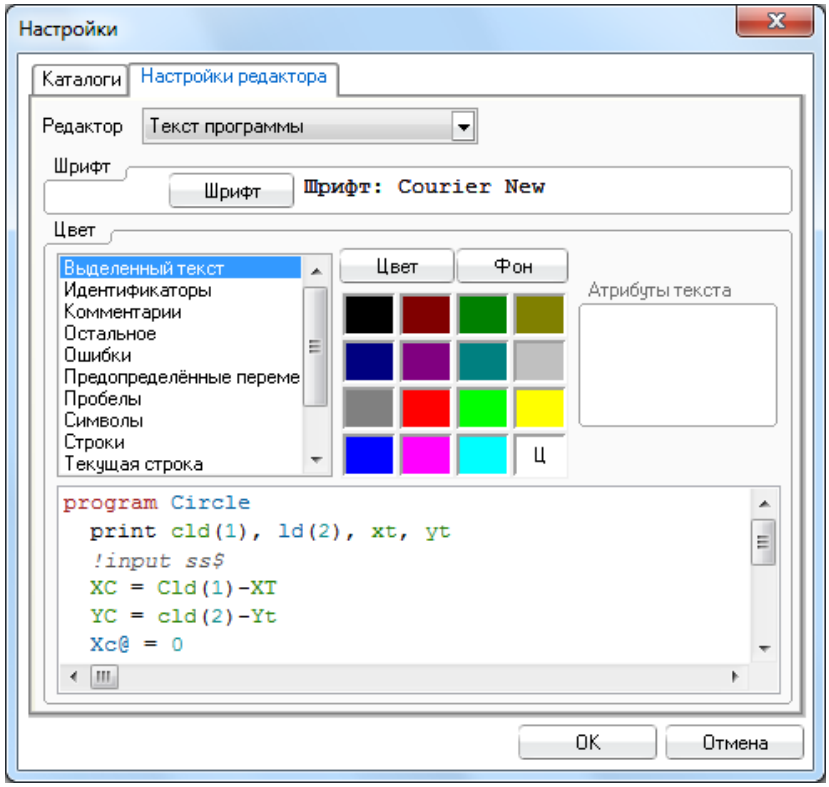
При определении реальных имен каталогов в процессе работы указанная переменная будет заменена соответствующим полным путем на момент запуска системы или на момент последнего редактирования системных установок.

Для загрузки последнего открытого ***.SPP** файла при запуске генератора постпроцессоров есть флажок <**При запуске загружать последний открытый файл**>.

2.2.5 Настройка редактора

Программы обработки технологических команд состоят из идентификаторов, чисел, зарезервированных переменных и функций, комментариев и т.д. Для каждой из этих групп элементов можно настроить свой цвет и стиль шрифта, цвет фона.

Окно настройки редактора вызывается выбором пунктов **Установки** –> **Настройки редактора** в главном меню. В окне настраивается шрифт и подсветка синтаксиса для редактора программ обработки технологических команд, окна отображения текста файлов **CLData** и окна отображения управляющей программы.



Для изменения настроек в выпадающем списке <**Редактор**> необходимо выбрать один из редакторов: <**Текст программы**>, <**Текст CLData**> или <**Текст управляющей программы**>. Для выбранного редактора на панели <**Шрифт**> задаётся название и размер шрифта.

На панели <**Цвет**> для каждой группы элементов можно назначить цвет текста, цвет фона текста, стиль шрифта.

Для быстрого изменения цвета фона всех групп элементов существует флажок <**Общий фон**>. При установке флажка цвет фона всех групп заменяется на цвет фона текущего элемента.

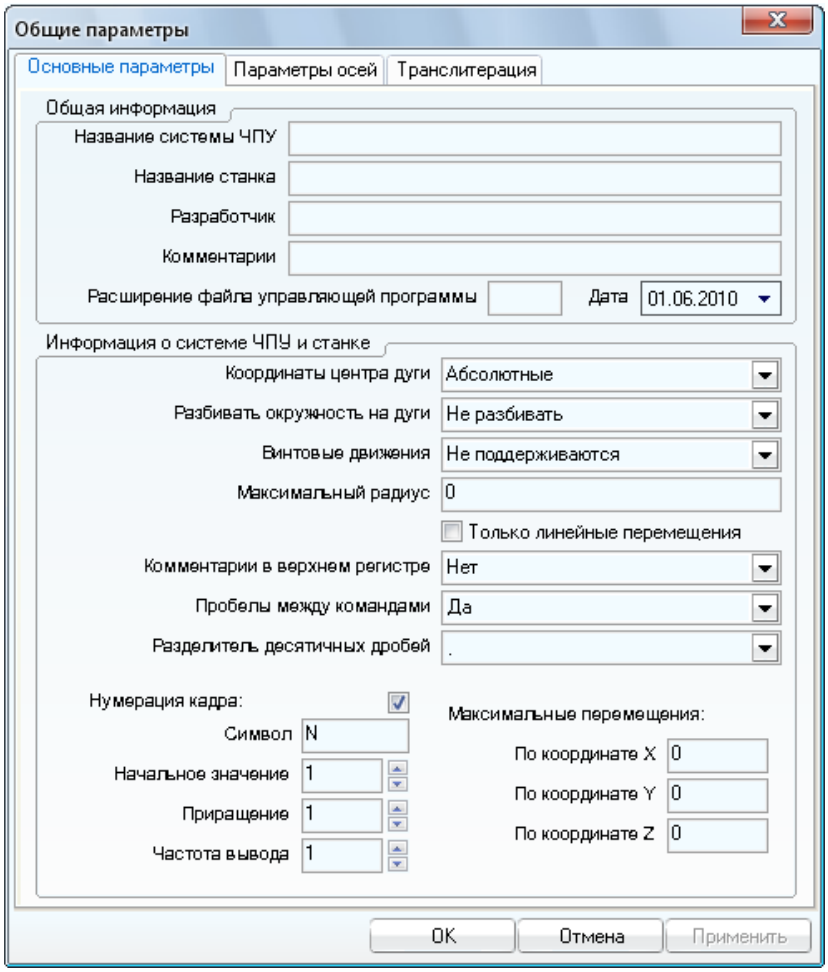
При закрытии окна кнопкой <**Ok**> все внесенные изменения

сохраняются. Если окно закрыто нажатием на кнопку <**Отмена**>, то изменения игнорируются.

2.2.6 Заполнение общих параметров

Основные параметры

Окно общих параметров открывается нажатием кнопки  на главной панели или выбором пунктов **Вид** – **Станок** в главном меню.



В верхней части страницы <**Общая информация**> вводятся название станка, <**Название системы ЧПУ**>, <**Название станка** >, <**Разработчик**>, <**Комментарии**> разработчика, <**Дата**> разработки постпроцессора и <**Расширение файла управляющей программы**>. Все управляющие программы, сгенерированные с использованием редактируемого файла настройки, будут записаны в файлы с указанным расширением.

<**Информация о системе ЧПУ и станке**> задаётся в соответствующей панели.

В поле <**Координаты центра дуги**> определяется способ задания центра окружности. Если выбран относительный способ задания центра, то координаты центра окружности в переменных <**XC**>, <**YC** >, <**ZC**> задаются относительно текущей точки. Если выбран абсолютный способ задания, то координаты центра задаются в абсолютных значениях.

В поле <**Разбивать окружности на дуги**> предлагается выбрать способ представления дуг в управляющей программе (четверть, половина или полная окружность). Постпроцессор во время работы фиксирует пересечение дугами квадрантов и, в случае необходимости, разбивает дуги на половины или четверти.

В поле <**Винтовые движения**> задаётся информация о поддержке системой ЧПУ винтовых движений. Если винтовые движения не поддерживаются, постпроцессор автоматически аппроксимирует, винтовые дуги на отрезки, и формирует команды линейных перемещений.

В поле <**Максимальный радиус**> вводится значение максимального радиуса дуги, поддерживаемого системой ЧПУ, при превышении которого, постпроцессор заменяет дуги линейными перемещениями. Значение 0 в этом поле отключает контроль, и система не будет аппроксимировать дуги при любом значении радиуса.

Для случая, когда система ЧПУ вообще не поддерживает круговую интерполяцию и винтовые движения, предусмотрен флажок <**Только линейные перемещения**>. В этом случае все дуги вне зависимости от полей <**Винтовые движения**> и < **Максимальный радиус**> разбиваются на линейные перемещения.

Предусмотрена возможность формировать <**Комментарии в верхнем регистре**> и расставлять <**Пробелы между командами** > управляющей программы.

Для формирования номера кадра имеется группа полей < **Нумерация кадра**>. Если соответствующая галочка на панели не стоит, то функция автоматической нумерации кадров выключена. Если эту галочку включить, то станут доступными ряд полей по настройке автоматической нумерации. В поле <**Символ**> вводится идентификатор регистра или произвольный символ, выводимый в кадр перед значением номера кадра. Если в данном поле указать идентификатор регистра, то появляется возможность дополнительного контроля над номером кадра за счет настроек формата регистра (например, если номер кадра выйдет за диапазон возможных значений регистра, то нумерация начнется заново). В случае автоматической нумерации кадры нумеруются, начиная с <**Начального значения**>. После каждого вывода кадра к номеру кадра добавляется <**Приращение**>. Кроме доступности величины приращения из окна <**Общие параметры**> она доступна и программно через предопределенную переменную <**BlockStep**>.

Номер кадра выводится с заданной частотой. Если <**Частота вывода**> равна 1, то номер кадра выводится в каждой команде формирования кадра, если 2 – через одну команду, 3 – через две команды и т.д. Если частота вывода равна нулю, то номер кадра не выводится.

<**Максимальные перемещения**> по основным координатам позволяют контролировать корректность управляющей программы. При превышении максимально допустимого перемещения по одной из координат будет выведено соответствующее предупреждение.

<**Разделитель десятичных дробей**> символ, который будет выводиться в кадры УП в качестве разделителя десятичных дробей в числах.

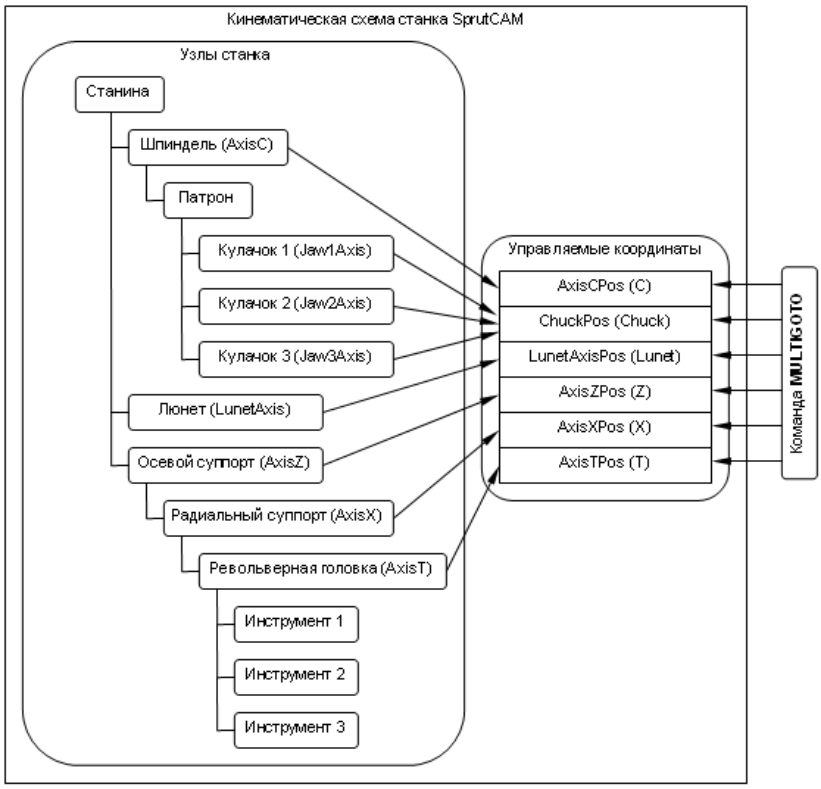
При закрытии окна кнопкой <**ОК**> все внесенные изменения сохраняются. Нажатие на кнопку <**Применить**> сохраняет изменения, но окно остаётся активным. Если окно закрыто нажатием на кнопку <**Отмена**>, то изменения игнорируются.

Заполнение параметров управляемых координат станков

Система SprutCAM генерирует траектории в координатах станка, заданных кинематической схемой оборудования. Генератор постпроцессоров предоставляет все необходимые инструменты для создания и отладки постпроцессоров для многокоординатных станков.

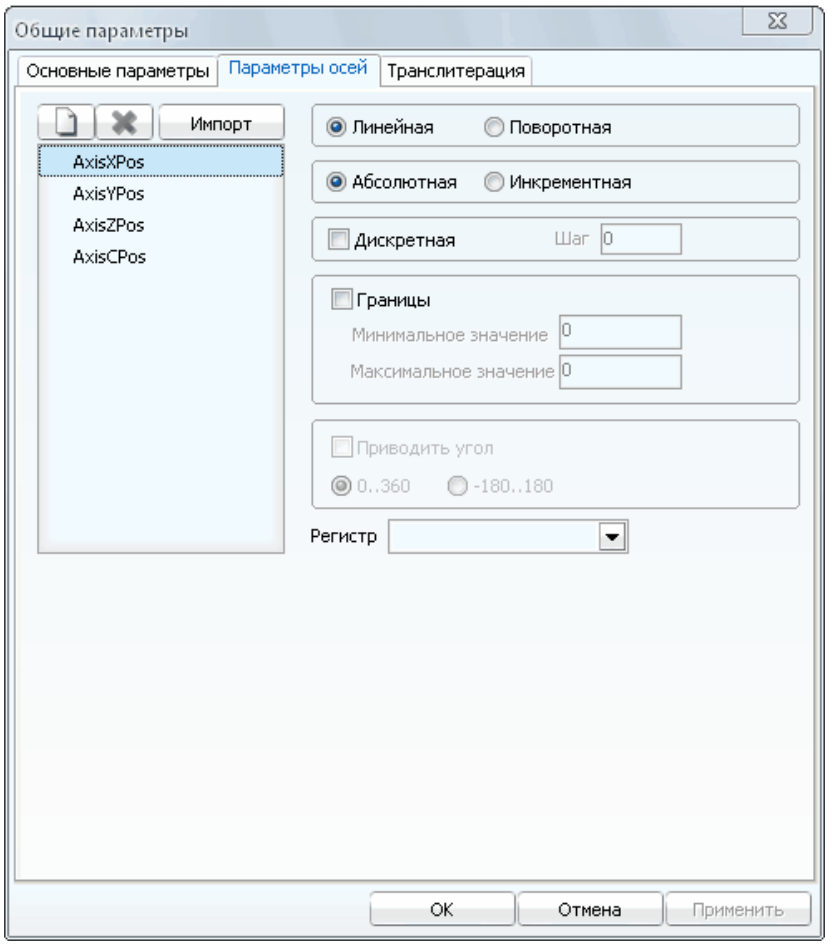
Траектория, рассчитанная и переданная в построцессор системой SprutCAM, представляет собой последовательность технологических команд CLDATA. Для управления перемещениями узлов традиционных станков достаточно команд на перемещение линейных координат X, Y и Z (<**GOTO**>), а также поворотной координаты A, B или C (<**ROTABL**>). Управление же узлами более сложных станков осуществляется при помощи технологической команды <**MULTIGOTO**>.

Особенностью команды <**MULTIGOTO**> является изменяющийся состав координат станка, перемещения по которым она осуществляет. <**MULTIGOTO**> может перемещать как линейные координаты – <**MULTIGOTO X Y Z**>, так и поворотные оси – <**MULTIGOTO A**>, заменяя, таким образом, команды <**GOTO**> и <**ROTABL**>. Кроме того, в одной команде <**MULTIGOTO**> могут быть обозначены перемещения, как по линейным, так и нескольким поворотным осям – <**MULTIGOTO X Z A C**>. Еще более актуальной особенностью команды <**MULTIGOTO**> является возможность управлять координатами специфичными для конкретного станка – зажимом кулачков патрона, тормозом шпинделя, перемещением люнета или магазина и т.д. (<**MULTIGOTO Lunet**>, <**MULTIGOTO Chuck**>). Состав координат станка, управление которыми может осуществлять команда, зависит от кинематической схемы оборудования.



Программный доступ к параметрам команды многокоординатного перемещения <**MULTIGOTO**> может быть осуществлен как при помощи универсального массива <**CLD**> или оператора <**Cmd**>, так и при помощи специально предназначенного для этой команды предопределенного массива <**GMA**>.

Каждый из элементов в массиве <**GMA**> всегда соответствует управляемой координате в кинематической схеме соответствующего станка. Список возможных элементов массива <**GMA**> (управляемых координат) определяется в постпроцессоре для каждого станка однократно. Он редактируется на вкладке < **Параметры осей**> окна <**Общие параметры**>. Набор управляемых координат и их характеристики можно как импортировать из системы SprutCAM, так и задать вручную.

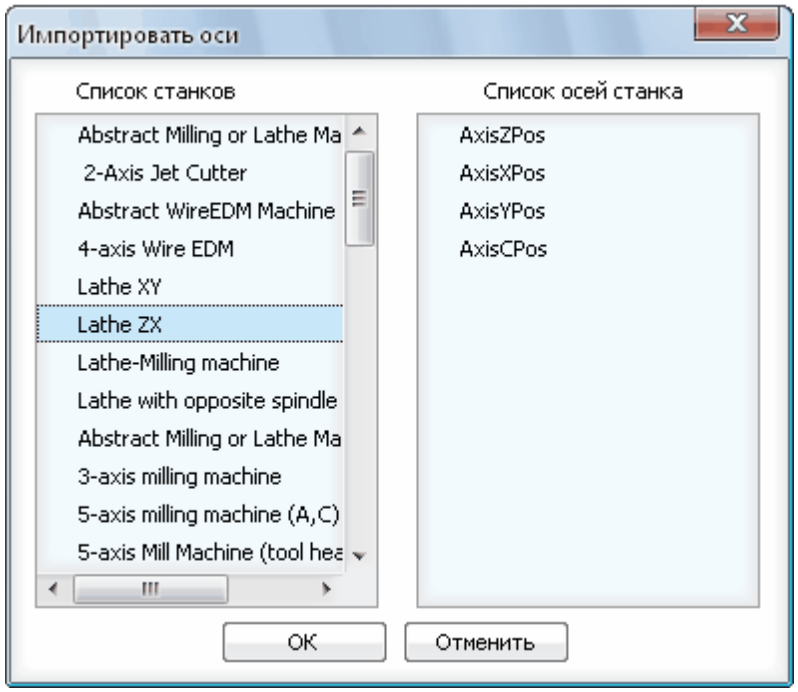


В списке слева отображается полный список управляемых координат, доступный в данном постпроцессоре. Выделяя нужную координату в списке можно отредактировать ее параметры, отображаемые в правой части окна.

- <**Линейная/Поворотная**> – указывает является ли ось станка линейной или поворотной.
- <**Относительная/Абсолютная**> – определяет способ занесения координат в массив <**GMA**>: в абсолютных значениях или в приращениях относительно предыдущего значения.
- <**Дискретность**> – устанавливает дискретность соответствующей оси станка.
- <**Границы**> – назначает максимальную и минимальную границы значений по соответствующей координате.
- <**Приводить угол к диапазону**> – параметр, доступный только для поворотных осей, позволяющий приводить периодические неограниченные в SprutCAM значения углов к указываемому диапазону.
- <**Регистр**> – позволяет выбрать регистр, с которым ассоциируется данная управляемая координата.

Назначение кнопок окна редактирования:

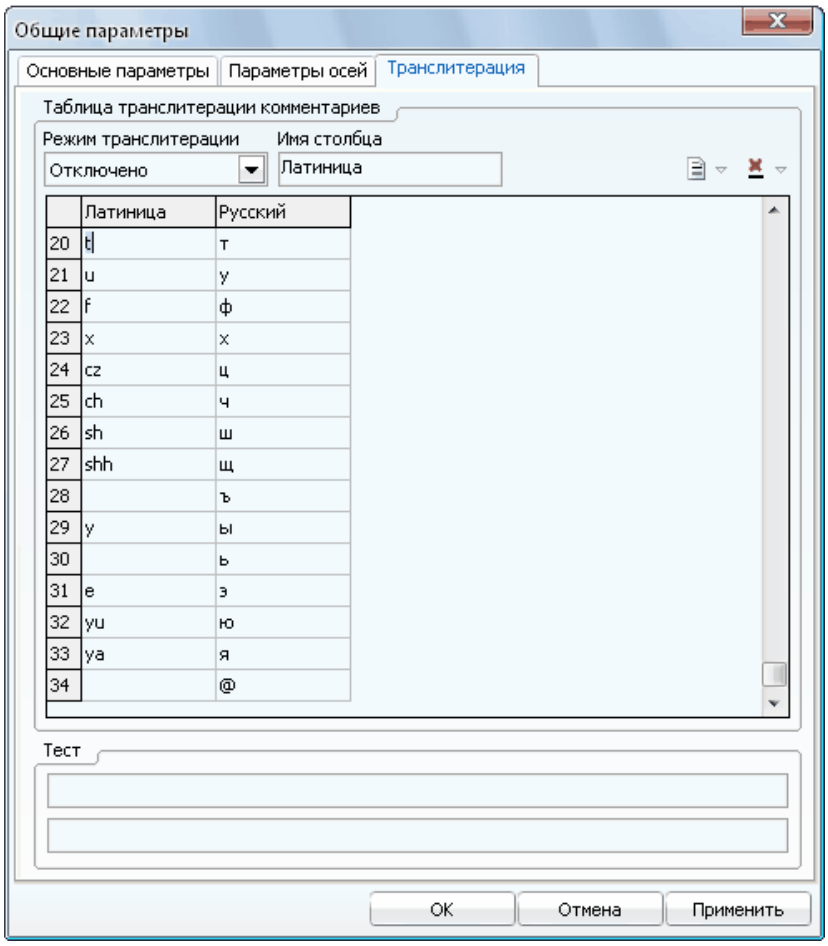
-  <**Новая ось**> – добавление новой координаты в список.
-  <**Удалить ось**> – удаление выделенной координаты из списка.
- <**Импорт**> – импорт списка управляемых координат из кинематической схемы станка SprutCAM. При щелчке по кнопке открывается окно со списком доступных станков (слева), выделяя в котором нужный станок, в правой части окна можно увидеть список его управляемых координат. Добавление списка производится нажатием на кнопку <**ОК**>, выход без добавления списка осуществляется нажатием на кнопку <**Отменить**>:



Транслитерация комментариев

Функция транслитерации комментариев в генераторе постпроцессоров позволяет автоматизировать процедуру преобразования строк комментария (содержащихся в технологической команде <**COMMENT**>), написанных с использованием символов, не поддерживаемых стойкой ЧПУ (например, при использовании языка с набором символов, отличающимся от кодовой таблицы ASCII) в строки, содержащие символы, понимаемые стойкой ЧПУ.

Управление режимом транслитерации осуществляется через вкладку <**Транслитерация**> окна <**Общие параметры**>.



Поле <**Режим транслитерации**> окна определяет активный столбец в таблице транслитерации, т.е. определяет исходный язык комментариев. По умолчанию выбран пункт <**Отключено**>, при котором преобразование не выполняется.

Столбцы таблицы транслитерации определяют набор доступных для преобразования языков. Первый столбец с именем <**Latin**> является ключевым, т.е. он содержит символы, которые будут подставляться вместо соответствующих символов в активном столбце, определяемом режимом транслитерации. Таким образом, строки таблицы устанавливают наборы символов для каждого из доступных языков.

Таблица транслитерации является редактируемой. Содержимое ячеек таблицы может быть изменено в любой момент. Редактирование заголовка выделенного столбца осуществляется в поле <**Имя столбца**>. Для добавления и удаления новых столбцов и строк предназначены кнопки в правом верхнем углу окна:

- <**Новая строка**> – добавление новой строки после выделенной;
- <**Новый столбец**> – добавление нового столбца после выделенного;
- <**Удалить строку**> – удаление выделенной строки;
- <**Удалить столбец**> – удаление выделенного столбца.

В нижней части окна содержится область <**Тест**>, предназначенная для тестирования и позволяющая опробовать выбранный режим транслитерации. В верхнее поле этой области вводится исходная строка, а в нижнем поле при этом отображается результат транслитерации.

2.2.7 Обязательное заполнение пользователем информации о системе ЧПУ и станке

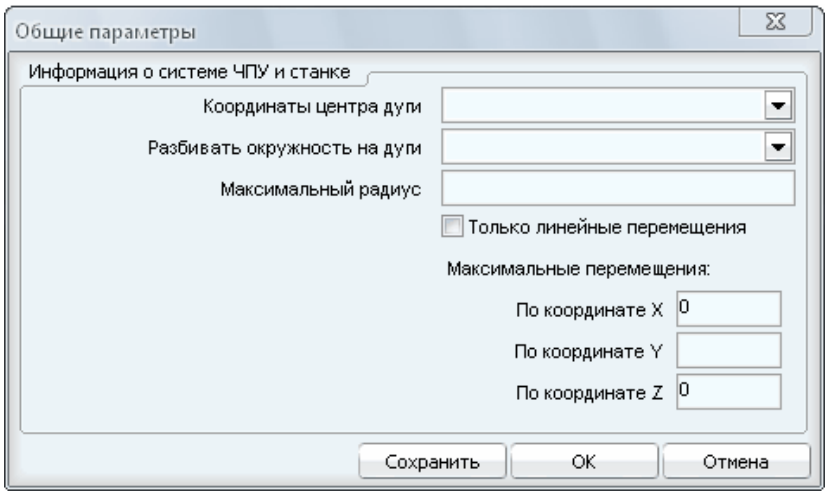
Для того чтобы перед началом работы с постпроцессором пользователь обязательно заполнил данные о системе ЧПУ и станке необходимо в текстовом редакторе внести изменения в раздел файла постпроцессора <**[Common definitions]**>. Обычно данный раздел выглядит следующим образом:

```
[Common definitions]
Spaces N            ! Y / N
CtrInc Y            ! Y / N
ArcDivision N       ! N / 90 / 180
Helical Y           ! Y / N
MaxRad 1000         ! >= 0
LinearMovement N    ! Y / N
UpperCaseComment N  ! Y / N
Sequence N 1 1 1    ! Symbol Frequence StepValue StartValue
Xmax 0              ! >= 0
Ymax 0              ! >= 0
Zmax 0              ! >= 0
```

При изменении значения любого параметра на "?" значение параметра будет запрашиваться у пользователя перед началом работы до первого сохранения файла. Например, при подобном заполнении раздела:

```
[Common definitions]
Spaces N            ! Y / N
CtrInc ?            ! Y / N
ArcDivision ?       ! N / 90 / 180
Helical Y           ! Y / N
MaxRad ?            ! >= 0
LinearMovement ?    ! Y / N
UpperCaseComment N  ! Y / N
Sequence N 1 1 ?    ! Symbol Frequence StepValue StartValue
Xmax 0              ! >= 0
Ymax ?              ! >= 0
Zmax 0              ! >= 0
```

При загрузке файла постпроцессора пользователю будет предложено заполнить следующие данные:



При сохранении данные запишутся в файл постпроцессора, и в будущем это окно появляться не будет. Для использования заполненных данных до следующей загрузки файла необходимо нажать на кнопку <**OK**>. При отмене файл постпроцессора загружен не будет.

Ввод дополнительной информации

При необходимости ввода дополнительной информации разработчик постпроцессоров может формировать перечень параметров обязательных для заполнения пользователем. В этом случае в окне запроса информации появится новая группа, в которой отобразится перечень. Параметры могут быть двух видов: выпадающий список и поле для ввода. С каждым параметром связана глобальная переменная постпроцессора. После ввода пользователем параметров каждая такая переменная принимает начальное значение в соответствии с выбранным или введённым параметром.

Для добавления дополнительных параметров в окно запроса информации необходимо в текстовом редакторе внести изменения в раздел файла постпроцессора <**[Initial questions block]**>. Формат раздела:

```
[Initial questions block]
Имя_переменной   Значение_пер          Описание_параметра_или_запрос
Значение_1       Описание_значения_1
Значение_2       Описание_значения_2
...
Значение_N       Описание_значения_N
Имя_переменной   Значение_пер          Описание_параметра_или_запрос
Имя_переменной   Значение_пер          Описание_параметра_или_запрос
Значение_1
[Initial questions block end]
```

Здесь:

- <**Имя_переменной**> – имя существующей в постпроцессоре глобальной переменной. Переменная может быть трёх типов: <**String**>, <**Real**>, <**Integer**>;

- <**Значение_пер**> – стартовое значение, которое примет глобальная переменная при выполнении постпроцессора. Если вместо значения подставить знак "?", то такая переменная попадёт в окно запроса информации в качестве параметра;
- <**Описание_параметра_или_запрос**> – данное сообщение появится в описании запрашиваемого параметра, сообщение должно быть заключено в кавычки;
- <**Значение_N**> – значение того же типа что и глобальная переменная;
- <**Описание_значения_N**> – описание значения, описание должно быть заключено в кавычки.

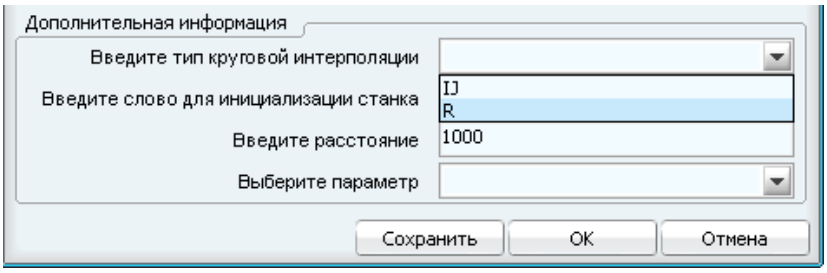
Значения строковых переменных должны быть заключены в кавычки.

Например, при подобном заполнении раздела:

```
[Initial questions block]
Intrp  ?     "Введите тип круговой интерполяции"
       1     "IJ"
       2     "R"
Init$  ?     "Введите слово для инициализации станка"

L      ?     "Введите расстояние"
       1000
P$     ?     "Выберите параметр"
       "A"   "первый параметр"
       "B"   "второй параметр"
[Initial questions block end]
```

При загрузке файла постпроцессора пользователю будет предложено заполнить следующие данные:



После заполнения и сохранения данных пользователем раздел в файле постпроцессора принимает следующий вид:

```
[Initial questions block]
Intrp 1                      "Введите тип круговой интерполяции"
      1                      "IJ"
      2                      "R"
Init$ "G9G1G20G40G90F4000Z0M5" "Введите слово для инициализации станка"
L     995                    "Введите расстояние"
      1000
P$    "B"                    "Выберите параметр"
```

```
      "A"                    "первый параметр"
      "B"                    "второй параметр"
[Initial questions block end]
```

В дальнейшем при загрузке данного файла постпроцессора окно заполнения дополнительной информации появляться не будет, а переменные <**Intrp**>, <**Init$**>, <**L**> и <**P$**> при инициализации будут принимать значения 1, "G9G1G20G40G90F4000Z0M5", 995 и "B" соответственно.

2.2.8 Определение структуры и формата кадра (формирование списка регистров)

Для определения структуры и формата кадра управляющей программы необходимо сформировать список регистров и определить их параметры.

Кадр управляющей программы состоит из слов, каждое слово содержит адрес и значение. <**Адрес**> – буква (иногда несколько букв), <**Значение**> – число представленное в определенном формате. С каждым адресом в постпроцессоре связан определенный <**Регистр**>.

Понятие <**Регистра**> в постпроцессоре объединяет следующие свойства:

- идентификатор регистра (адрес в управляющей программе);
- формат вывода значения этого адреса в кадр;
- текущее значение, соответствующее адресу;
- предыдущее значение, соответствующее адресу;
- имя регистра (переменная постпроцессора, связанная с адресом, через которую производится доступ как к текущему, так и к предыдущему значению регистра).

Постпроцессор имеет средства автоматического формирования кадра управляющей программы. Происходит это так: перебираются регистры по порядку, начиная с первого; если старое и текущее значения регистра различаются, то регистр выводится в кадр, а его старое значение приравнивается текущему. Если старое и текущее значения регистра равны, то регистр в кадр не выводится.

При выводе регистра в кадр управляющей программы сначала записывается его идентификатор, а затем текущее значение регистра. Число выводится в кадр в соответствии с описанным в регистре форматом (количество знаков целой части числа, количество знаков дробной части числа, наличие десятичной точки, знака, лидирующих и незначащих нулей).

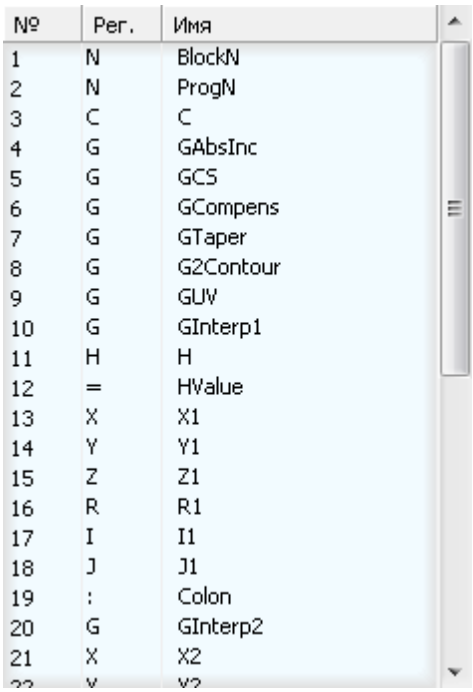
Таким образом, для того, чтобы определить структуру и формат кадра управляющей программы необходимо заполнить список регистров и задать их свойства. Регистры должны быть расположены в том же порядке, в котором их идентификаторы и значения должны появляться в кадре управляющей программы. Данное правило действительно для программ обработки

технологических команд. При формировании кадра по шаблону порядок вывода значений соответствует порядку регистров в шаблоне.

Примечание: *Разные регистры не должны иметь одинаковые имена, но у них могут быть одни и те же идентификаторы (адреса слов в кадре управляющей программе). Это позволяет создавать отдельный регистр для каждой группы функций одного типа.*

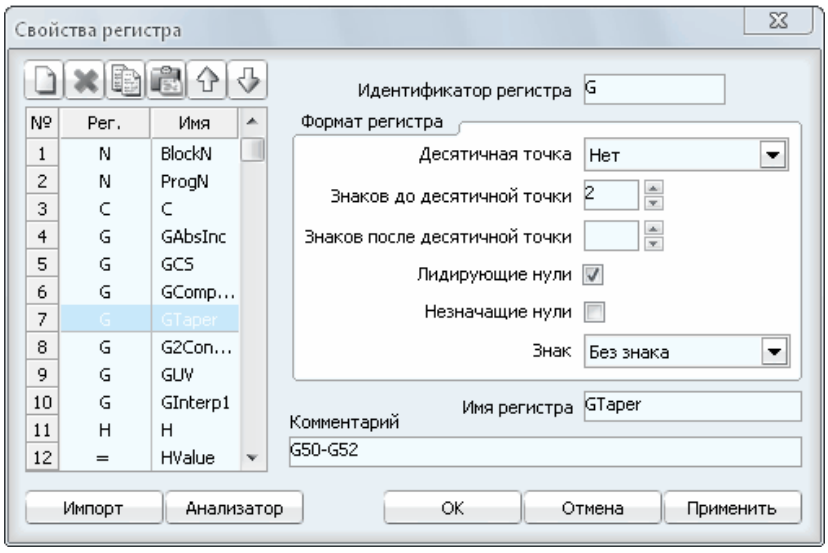
Например, для подготовительной функции переключения абсолютной/относительной системы отсчета создается регистр с именем <**ABSOLUTE**> и идентификатором <**G**>. А для подготовительных функций позиционирования, линейной интерполяции и интерполяции по окружности создается регистр с именем <**INTERP**> и таким же идентификатором <**G**>. Это позволит, во-первых, при необходимости, вывести обе эти команды в один кадр управляющей программы (например, <**N190 G91 G1 X50 Y30** >). А, во-вторых, по текущим значениям регистров <**ABSOLUTE**> и <**INTERP**> отслеживать текущий режим перемещений (позиционирование, линейная или круговая интерполяция) и в какой системе отсчета (абсолютной или относительной) задаются координаты.

Список регистров отображается в левой части главного окна.



№	Рег.	Имя
1	N	BlockN
2	N	ProgN
3	C	C
4	G	GAbsInc
5	G	GCS
6	G	GCompens
7	G	GTaper
8	G	G2Contour
9	G	GUV
10	G	GInterp1
11	H	H
12	=	HValue
13	X	X1
14	Y	Y1
15	Z	Z1
16	R	R1
17	I	I1
18	J	J1
19	:	Colon
20	G	GInterp2
21	X	X2

Окно редактирования свойств регистра открывается двойным нажатием левой клавиши мыши на строке регистра, либо нажатием кнопки  на панели инструментов, либо выбором соответствующего пункта из главного меню: **Вид** –> **Свойства регистра**.



В левой части окна отображается список регистров. Выше находятся кнопки редактирования списка регистров:

- – создать новый регистр, созданный регистр помещается после текущего;
- – удалить текущий регистр;
- – скопировать текущий регистр в буфер обмена;
- – вставить регистр из буфера обмена после текущего;
- – переместить текущий регистр вверх по списку;
- – переместить текущий регистр вниз по списку.

Поменять порядок расположения регистров можно, нажав левую клавишу мыши на номере регистра и, не отпуская левой клавиши, переместить его в нужную позицию.

В правой части окна отображаются и могут быть отредактированы свойства регистров, которые выделены в списке слева.

- <**Идентификатор регистра**> – символы, выводящиеся в кадр управляющей программы перед значением регистра (адрес слова в кадре управляющей программы).
- <**Десятичная точка**> – может иметь несколько значений:
  - <**Отсутствует**>;
  - <**Присутствует**> – выбор этого варианта значит, что точка будет стоять только тогда, когда значение регистра будет иметь дробную часть;
  - <**Присутствует всегда**>.
- <**Знаков до десятичной точки**> – максимальное количество знаков в целой части числа.
- <**Знаков после десятичной точки**> – количество знаков в дробной части числа.

- <**Лидирующие нули**> и <**Незначащие нули**> – управление выводом нулей перед и после значения регистра.
- <**Знак**> – управление выводом знака значения регистра. Возможны:
  - <**Без знака**>;
  - <**Только "-"**>;
  - <**Всегда "-" и "+"**>.
- <**Имя регистра**> – имя, с которым оперируют программы обработки команд.
- <**Комментарий**> – комментарий к регистру.

Кнопка <**Импорт**> предназначена для импорта списка регистров из постпроцессоров форматов <**SPP**>, <**PPP**> (формат старой версии), а так же из постпроцессоров SurfCAM.

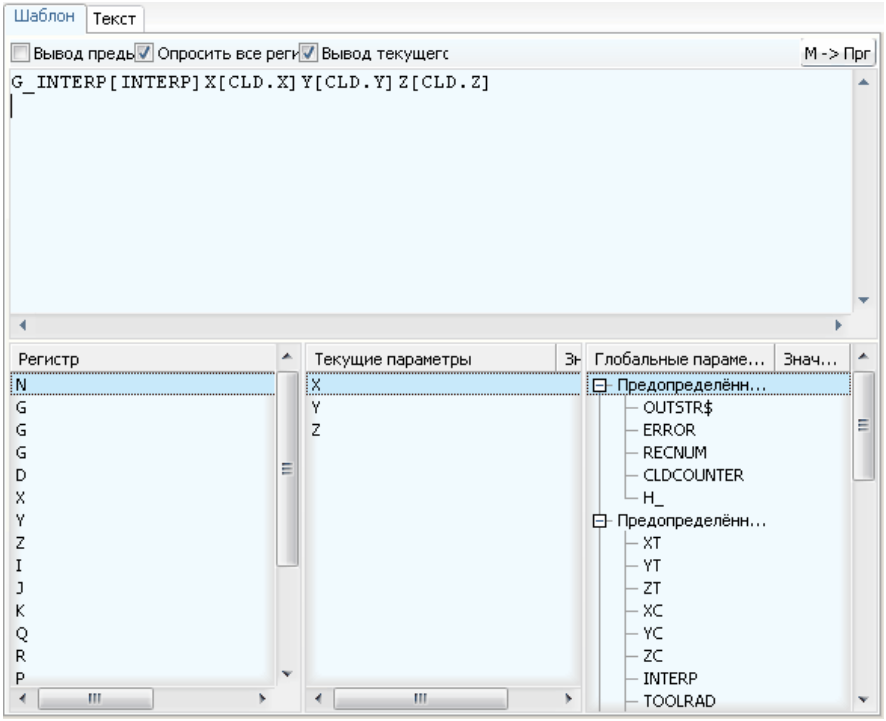
Для формирования списка регистров на основе управляющей программы предназначена кнопка <**Анализ**>. При этом анализируется текст указанной управляющей программы и все найденные адреса добавляются в список регистров.

При закрытии окна кнопкой <**ОК**> все внесенные изменения сохраняются. Если окно закрыто нажатием кнопки <**Отмена**>, то изменения игнорируются. При нажатии на кнопку <**Применить**> все внесённые изменения сохраняются, окно редактирования регистров не закрывается.

2.2.9 Шаблоны трансляции технологических команд

Использование шаблонов позволяет просто и быстро формализовать процесс трансляции команд CLData в кадры управляющей программы. Каждой команде соответствует свой шаблон, в котором указывается, какие адреса должны появится в кадре и откуда должны быть подставлены значения, для этих адресов.

Редактор шаблонов расположен на закладке <**Шаблон**>.

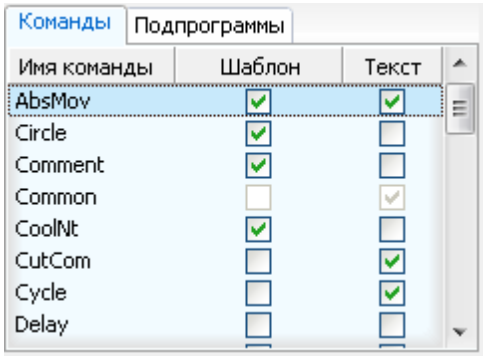


Для упрощения процесса создания шаблонов в нижней части окна находятся список регистров, список параметров текущей команды и список глобальных параметров процесса обработки.

Двойной щелчок по любому элементу этих списков приводит к его подстановке в текст шаблона.

Над полем текста шаблона расположена панель флажков управления процессом обработки шаблона.

Активизация шаблона выполняется установкой соответствующего флажка на панели списка технологических команд.



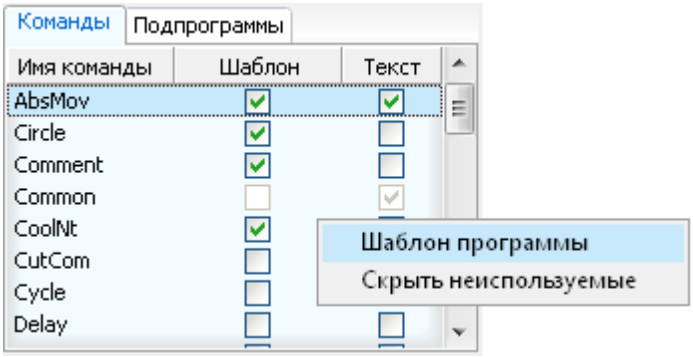
Правила формирования шаблонов подробно описаны в главе Шаблоны.

Команды CLData, которые не обрабатываются данным постпроцессором можно скрыть, выбрав соответствующий пункт контекстного меню окна команд.

2.2.10 Программы обработки технологических команд

Так же как и шаблоны, программы обработки технологических команд предназначены для формализации процесса формирования кадров управляющей программы. Они могут дополнять шаблоны или использоваться вместо шаблонов. Каждый из этих способов трансляции технологических команд имеет свои достоинства и недостатки. Программы – очень гибкий и мощный инструмент формализации любой, самой сложной логики процесса трансляции. Однако, овладение этим инструментом предполагает наличие хотя бы самых минимальных навыков программирования.

Каждой технологической команде соответствует своя программа обработки. Команды CLData, которые не обрабатываются данным постпроцессором можно скрыть, выбрав соответствующий пункт контекстного меню окна команд.



Программы обработки технологических команд пишутся на специальном проблемно-ориентированном языке и могут содержать математические выражения и функции, операторы ввода/вывода, условные операторы, циклы, операторы перехода, вызовы подпрограмм, операторы формирования кадров управляющей программы. Язык написания программ обработки подробно описан в главе <**Описание языка**>.

Каждая программа обработки технологической команды начинается с заголовка, состоящего из слова <**PROGRAM**> и имени программы, а заканчивается словом <**END**>. Имя программы совпадает с именем команды, которую она обрабатывает. Параметры технологической команды доступны в программе через предопределенный массив <**CLD**> и оператор <**Cmd**>.

Активизация программ обработки технологических команд выполняется установкой соответствующего флажка на панели списка технологических команд.

Не активизированная программа не транслируется и не выполняется даже при наличии текста. Исключение составляет программа <**COMMON**>, которая транслируется и выполняется всегда первой, один раз. Программа <**COMMON**> предназначена для определения глобальных переменных, т.е. переменных которые доступны из любой программы и подпрограммы.

При двойном нажатии в поле списка технологических команд для

пустой программы формируется заготовка.

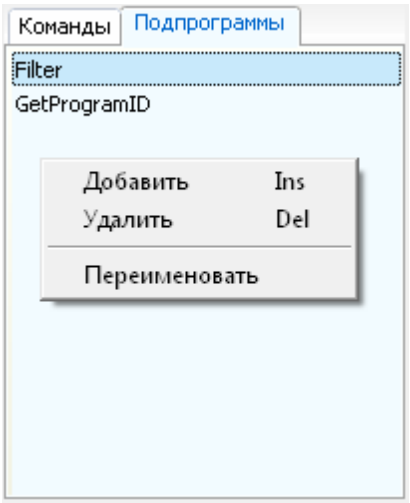
При нажатии на правую кнопку мыши в поле списка подпрограмм появляется контекстное меню, в котором содержатся пункты:

- <**Добавить**> – добавить новую подпрограмму, открывается окно ввода имени подпрограммы;
- <**Удалить**> – удалить текущую подпрограмму;
- <**Переименовать**> – переименовать текущую подпрограмму, открывается окно ввода имени подпрограммы.

В центральной части экрана расположено окно редактирования программ. При смене текущей команды в окно редактирования загружается исходный текст соответствующей программы (или подпрограммы).

2.2.11 Подпрограммы

Закладка <**Подпрограммы**>, расположенная в главном окне рядом со списком технологических команд содержит список дополнительных подпрограмм, которые пишутся на том же языке, что и обработчики технологических команд. Набор подпрограмм содержит одну стандартную подпрограмму <**Filter**>, а также может содержать произвольное количество определяемых пользователем вспомогательных подпрограмм. Добавлять новые, удалять имеющиеся подпрограммы, а также изменять их имена можно через контекстное меню данного окна. В центральной части экрана расположено окно редактирования текста подпрограмм. Исходный текст подпрограммы загружается в окно редактирования при выделении имени нужной подпрограммы в списке слева.



Имя подпрограммы может быть любым, не совпадающим с именами программ и уже имеющимися подпрограммами.

Так же как и программы обработки технологических команд подпрограммы предназначены для формализации процесса формирования кадров управляющей программы. В подпрограммы

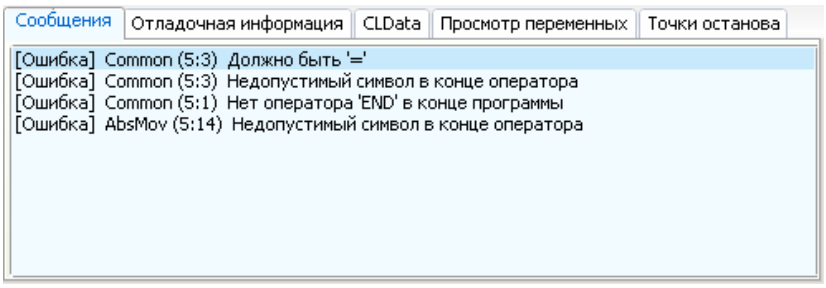
можно выделять повторяющиеся части кода программ и затем вызывать их из произвольных мест в обработчиках технологических команд. Синтаксис определения подпрограмм описан в разделе Описание языка. Подпрограммы.

2.2.12 Трансляция программ обработки технологических команд

Трансляция программ обработки технологических команд производится нажатием кнопки  на главной панели, выбором пункта главного меню **Выполнить** –> **Транслировать** или нажатием кнопок [**Ctrl-F9**] на клавиатуре.

Примечание: *Транслируются только активизированные программы.*

Список сообщений системы о результатах трансляции активизированных программ расположен в нижней части главного окна на закладке <**Сообщения**>.



Ошибки, возникающие в результате трансляции программ, заносятся в список сообщений системы с префиксом <**[Ошибка]**>. Двойное нажатие левой клавишей мыши на такое сообщение приведет к загрузке исходного текста программы обработки команды и установке курсора в позицию ошибки.

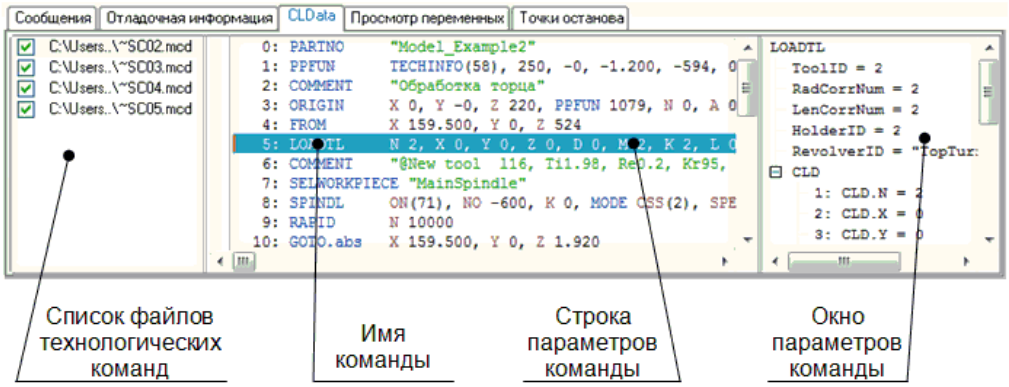
2.2.13 Работа с файлами технологических команд

Файлы технологических команд могут быть загружены из проекта системы SprutCAM (***.stc**, ***.stcx**). Для загрузки необходимо нажать кнопку на главной панели, или выбрать соответствующий пункт в главном или контекстном меню.

После загрузки поле <**Проект SprutCAM**> на главной панели содержит имя проекта, из которого добавлены файлы операций.

В нижней левой части главного окна, на закладке <**CLData**>, расположен список файлов технологических команд. Каждая строка списка соответствует одной операции. Флажками отмечены операции, которые будут участвовать в формировании

управляющей программы.



Список операций может быть отредактирован через контекстное меню, выпадающее при нажатии на правую кнопку мыши. Меню состоит из пунктов:

- <**Добавить**> – добавить файл технологических команд;
- <**Удалить**> – удалить выделенный файл;
- <**Удалить все**> – удалить все файлы.
- <**Обновить CLData**> – загрузить из системы **SprutCAM** обновленный список файлов CLData для открытого там проекта.

В центре на закладке <**CLData**> расположено окно текстового представления технологических команд (команд CLData). В нем отображается список команд для файла, который выделен в списке слева. Каждая технологическая команда в тексте CLDATA – это отдельная строка, состоящая из имени команды и ее параметров. При двойном нажатии левой клавиши мыши на имени команды в редактор шаблонов загружается текст шаблона, соответствующего команде, а в окно редактирования программы – программа обработки этой команды.

Справа от списка команд расположено окно параметров технологической команды. Оно содержит список имен параметров и их значений для выделенной команды. Чтобы в коде программы быстро получить значение того или иного параметра, необходимо выделить соответствующий параметр в данном окне и в контекстном меню (открывается правой кнопкой мыши) выбрать пункт <**Добавить параметр в код**>, либо сделать двойной щелчок левой кнопкой мыши. Параметры доступны в шаблонах через список параметров текущей команды, а в программах ещё и через массив <**CLD**> и оператор <**Cmd**>.

2.2.14 Тестовая генерация управляющих программ

Запуск тестовой генерации управляющей программы производится нажатием кнопки  на главной панели, выбором пункта главного меню **Выполнить** –> **Выполнить** или нажатием кнопки [**F9**] на

клавиатуре.

Перед генерацией управляющей программы производится трансляция ещё не оттранслированных программ. При возникновении ошибок трансляции процесс прерывается.

Исходными данными для тестовой генерации являются загруженные в постпроцессор файлы технологических команд. Управляющая программа формируется на основании тех операций, которые отмечены флажком в списке загруженных технологических команд.

Генерация управляющей программы происходит следующим образом:

1. Считывается очередная команда из файла технологических команд;
2. Анализируется код команды, в соответствии с параметрами команды заполняется предопределённый массив <**CLD**> и предопределённый переменные;
3. Если активизирована программа обработки технологической команды, то вызывается программа обработки этой команды.
4. Если активизирован шаблон для этой команды, то формируется выходная строка по шаблону.

Примечание: *Таким образом, для каждой технологической команды могут быть активизированы и программа обработки и шаблон. В этом случае сначала выполнится программа обработки технологической команды, затем обработается шаблон. Если и шаблон и программа обработки не активизированы, то технологическая команда обработана не будет.*

5. Если в результате вышеперечисленных действий сформирован кадр управляющей программы, то выполняется подпрограмма <**Filter**>. Типичное назначение этой подпрограммы – замена фрагментов кадра перед окончательным выводом в управляющую программу.

Формирование кадра управляющей программы производится соответствующими операторами в программах обработки технологических команд или по шаблону. Причем, в случае с программой, в кадр выводятся идентификаторы и значения только тех регистров, значения которых были изменены после формирования предыдущего кадра.

Управляющая программа выводится в файл, указанный в поле имени файла управляющей программы и в окно управляющей программы, которое находится в правой части главного окна. Если это поле пустое, то управляющая программа выводится только в окно управляющей программы и не сохраняется в файле. Если управляющая программа содержит подпрограммы, то они также могут выведены в отдельные файлы, в этом случае появятся дополнительные закладки, переключившись на которые можно просмотреть сгенерированные тексты подпрограмм.

Отладочная информация, формируемая в программах обработки команд, выводится в окно отладочной информации.

Если в процессе генерации управляющей программы возникли ошибки, то информация о них выводится в окно сообщений.

После безошибочной генерации управляющей программы, при

двойном нажатии левой клавиши мыши на имени команды в окне <**CLData**>, в окне просмотра управляющей программы выделяются кадры управляющей программы, сгенерированные выбранной командой <**CLData**>. А так же при двойном нажатии на строке в окне просмотра управляющей программы, в окне <**CLData**> выделяется команда, при обработке которой был сгенерирован выбранный кадр управляющей программы.

2.2.15 Отладка программ

Интегрированный отладчик позволяет отслеживать работу программы, выполняя ее по шагам и контролируя соответствие текста программы и результатов ее работы, входить при этом в подпрограммы либо выполнять их как один шаг, контролировать в процессе пошагового выполнения программы значения переменных, просматривать результаты работы программы, связанные с выводом сообщений. Использование отладчика значительно снижает трудоемкость разработки программ, облегчает поиск и устранение ошибок.

Функции отладчика:

- запуск программы на выполнение – [**F9**];
- выполнить программу до текущего оператора – [**F4**];
- выполнить очередной оператор программы без входа в подпрограмму – [**F8**];
- установка и снятие точки останова выполнения программы – [**Ctrl+F8**];
- выполнить очередной оператор программы с входом в подпрограмму – [**F7**];
- прервать отладку программы – [**Ctrl+F2**];
- поместить новую переменную в окно переменных – [**Ctrl+F7**];
- вычисление выражений – [**Ctrl+F4**].

При запуске программы на выполнение происходит автоматическая проверка на необходимость перетрансляции всех модифицированных программ. Если необходимо – выполняется трансляция. Если на этапе трансляции не было выявлено ошибок – программа выполняется.

Начать выполнение программы в режиме отладки можно любой из команд: [**F4**], [**F7**], [**F8**] или [**F9**] в сочетании с точками останова в программах проекта.

Команда [**F4**] означает выполнение программы до оператора, находящегося на строке, на которой стоит курсор редактора. Если текущая строка не содержит выполняемого оператора – программа выполняется до своего окончания.

Строка, на которой прервалось выполнение программы, выделяется цветом, как курсор отладчика.

Команды [**F7**] и [**F8**] приводят к выполнению очередного оператора программы, на котором стоит текущая строка отладчика, если режим отладки уже активизирован. Иначе текущая строка отладчика установится на первом выполняемом операторе первой программы. Команда [**F7**], в отличие от [**F8**], позволяет войти в подпрограмму, если очередной оператор – вызов подпрограммы.

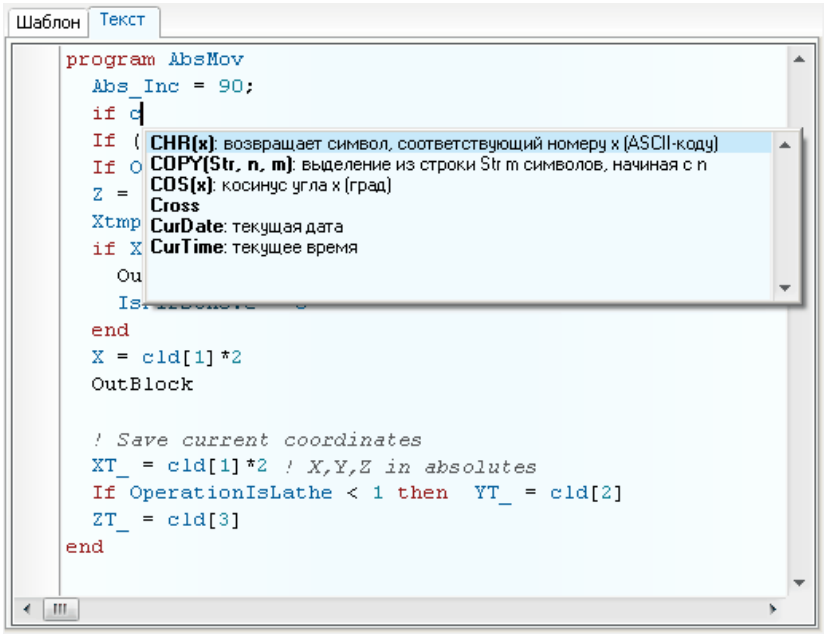
Комбинация клавиш для установки/снятия точки останова: [**Ctrl+F8**]. Когда программа дойдет до точки останова, выполнение ее будет остановлено с выходом в отладочную среду в месте нахождения точки останова.

В любой точке процесса отладочного выполнения программы можно прервать отладку командой [**Ctrl+F2**] или продолжить выполнение программы без отладки командой [**F9**].

Изменение количества строк в отлаживаемой программе в процессе отладки приведет к неправильной индикации текущего выполняемого оператора.

Поместить переменную или выражение в окно просмотра можно командой [**Ctrl+F7**]. Если команда исполняется в режиме редактирования, то можно получить имя переменной из текста программы. Для этого курсор редактора подводится к имени переменной и выполняется команда [**Ctrl+F7**]. Во время отладки, при наведении указателя мыши на переменную в окне редактирования программы, во всплывающем окне будет показано значение этой переменной.

При редактировании текста подпрограмм и программ обработки технологических команд для ввода имен предопределенных функций и переменных удобно пользоваться всплывающим окном-подсказкой. Его можно вызвать в любой момент редактирования путем нажатия сочетания клавиш [**Ctrl+пробел**]. Появившееся окно содержит список стандартных функций, имена которых совпадают с уже введенной в редакторе частью слова, а также показывает краткое описание этих функций. При двойном щелчке мышью по одной из строк списка выделенная в нем функция будет добавлена в текущую позицию редактора.



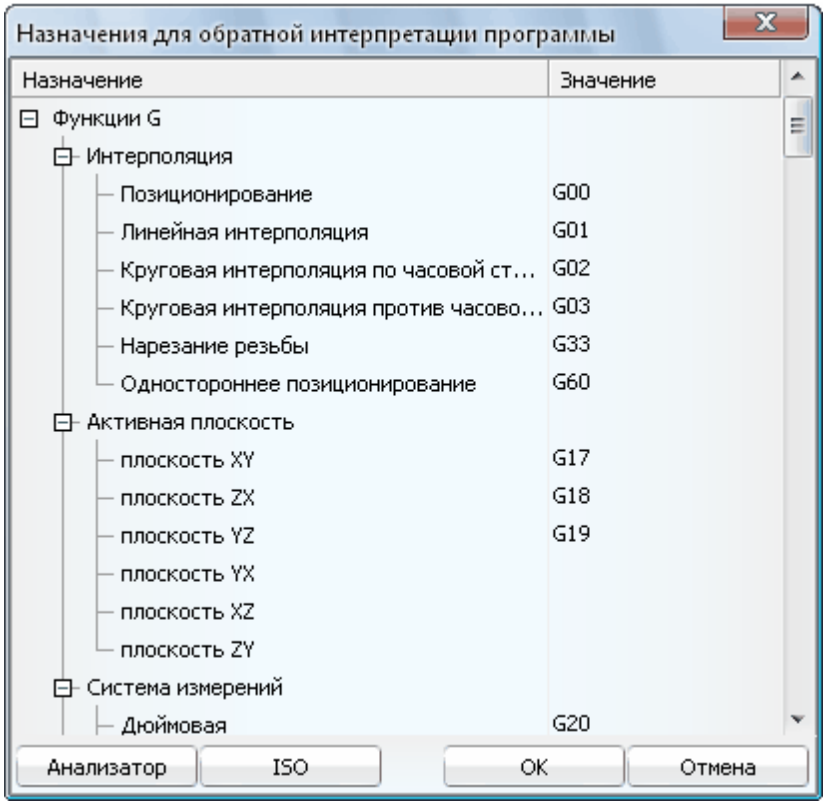
2.2.16 Обратная интерпретация программ

Для генерации управляющей программы не требуется обязательное определение параметров обратной интерпретации.

Необходимость в обратной интерпретации возникает при моделировании процесса обработки по готовой управляющей программе. Эту функцию выполняет программа **NCTuner**.

Окно редактирования назначений для обратной интерпретации вызывается из пункта главного меню **Вид** –> **Обратная интерпретация**. Так же это окно можно вызвать с панели инструментов кнопкой  .

Перечень назначений разбит на группы и представлен в виде дерева. Заполнение большего числа назначений приводит к более точному моделированию процесса обработки.



<**Анализатор**> автоматически формирует назначения по заполненным шаблонам технологических команд.

Есть возможность автоматически заполнить список назначений в соответствии со стандартом ISO (например, система ЧПУ Fanuc 6M). Это делается с помощью кнопки <**ISO**>.

При закрытии окна кнопкой <**OK**> все внесенные изменения сохраняются. Если окно закрыто нажатием на кнопку <**Отмена**>, то

изменения игнорируются.

3 Шаблоны

С помощью шаблонов можно описать вид кадров, которые следует сформировать при обработке очередной команды CLData.

Каждой команде CLData соответствует свой набор шаблонов. Шаблон представляет собой текст, близкий по виду строкам управляющей программы.

Например, при обработке команды CLData:

```
GOTO.abs X 19.599, Y -73.200, Z 28.030
```

следует сформировать кадр:

```
G1 X19.599 Y-73.200 Z28.030 F200
```

Для этого можно использовать шаблон:

```
G[1] X[XT] Y[YT] Z[ZT] F[FT]
```

Здесь:

- <**XT**>, <**YT**>, <**ZT**> – текущие значения положения инструмента;
- <**1**> – тип интерполяции;
- <**FT**> – значение подачи.

В шаблоне можно использовать значения переменных и регистров, математические выражения и функции, а так же вставлять модификаторы для управления выводом значений в управляющую программу.Так же в шаблонах можно использовать предопределенный массив <**GMA**>.

3.1 Состав шаблона

Шаблон может состоять из нескольких строк. В общем случае формат строки шаблона выглядит следующим образом:

```
ЭЛЕМЕНТ1 ЭЛЕМЕНТ2... ЭЛЕМЕНТN
```

3.1.1 Элемент шаблона

Каждый элемент шаблона выводится в кадр управляющей программы в том виде, в котором он содержится в шаблоне. Например:

- Шаблон:

```
G0 быстрое перемещение
```

- Кадр УП:

```
G0 быстрое перемещение
```

Если элемент шаблона заключен в квадратные скобки: "[" и "]", то в таком случае выполняется подстановка значения или переменной стоящей в скобках. Например:

- Шаблон:

```
[XT] [200]
```

- Кадр УП:

```
100.12456 200
```

В примере значение переменной <**XT**> равно 100.12456

В квадратных скобках допустимо использовать:

- Все переменные подпрограммы <**COMMON**>.
- Предопределённые переменные: <**XT**>, <**YT**>, <**ZT**>, <**XC**>, <**YC**>, <**ZC**>, <**INTERP**>, <**TOOLRAD**>, <**CLDATA$**>, <**ARCPLANE**>, <**XP**>, <**YP**>, <**ZP**>, <**FEED**>, <**TLCOMP**>, <**TRCOMP**>, <**FROMX**>, <**FROMY**>, <**FROMZ**>, <**CURCODE**>, <**NCNAME$**>, <**NCPATH$**>, <**BLOCKSTEP**>.
- Предопределённые функции: <**FLAGIN**>, <**CROSS**>, <**NEXTTOOLNUM**>, <**CURDATE**>, <**CURTIME**>.
- Все параметры текущей технологической команды, передаваемые через массив <**CLD**>.

В управляющей программе числа могут иметь различные представления. Формат вывода числа в кадр характеризуется параметрами:

- Десятичная точка. Может отсутствовать, присутствовать, в случае если число дробное или присутствовать всегда.
- Максимальное количество знаков в целой части числа.
- Количество знаков в дробной части числа.
- Наличие лидирующих нулей и наличие незначащих нулей.
- Знак значения регистра. Возможны варианты:

- без знака,
- только "-",
- всегда "-" и "+".

При занесении значения в кадр управляющей программы используется формат вывода чисел по умолчанию:

- Лидирующие и незначащие нули – отсутствуют.
- Если число дробное, присутствует десятичная точка.
- Если число отрицательное, присутствует знак.
- Знаков до и после десятичной точки столько, сколько необходимо для полного отображения числа.
- Идентификатор элемента выводится как текст.

3.1.2 Регистры в шаблонах

Для задания формата вывода значений, в соответствии с требованиями системы ЧПУ, используются регистры. Понятие <**Регистра**> в постпроцессоре объединяет следующие свойства:

- Идентификатор регистра;
- Текущее значение;
- Предыдущее значение;
- Формат вывода значения в кадр;
- Имя регистра;

Подробнее о регистрах см. главу <Определение структуры и формата кадра (формирование списка регистров)>.

Если перед открывающей квадратной скобкой указать имя или идентификатор регистра, то при выводе значения в кадр управляющей программы будет использован формат указанного регистра.

При разборе шаблона предполагается что слово, стоящее перед открывающей квадратной скобкой является ссылкой на регистр. Исполняющая система ищет регистр сначала по имени, затем, при не нахождении, по идентификатору регистра. В процессе поиска по идентификатору регистры перебираются в соответствии с порядком, определённым в списке регистров. При наличии в перечне нескольких регистров с одинаковым идентификатором результатом поиска будет регистр, находящийся в списке выше. Например:

- Шаблон:

```
G_INTERP[1] X[XT] Y[YT] Z[ZT] F[200]
```

- Кадр УП:

```
G1 X100.100 Y-245.100 Z-010.560 F200
```

Если, при завершении поиска, регистр найден не был, то значение выводится в формате по умолчанию, а перед значением выводится слово, стоящее в шаблоне перед открывающей квадратной скобкой. Например:

- Шаблон:

```
G_INTERP[1] X[XT] YYY[YT] Z[ZT] F[200]
```

- Кадр УП:

```
G1 X100.100 YYY-245.100034 Z-010.560 F200
```

В примере слово <**YYY**> в списке регистров отсутствует.

Если в результате поиска регистр найден, то старому значению регистра присваивается текущее значение, а текущему подставляемое. Затем исполняющая система сравнивает старое и текущее значение регистра, и если они различны, то идентификатор регистра и его значение выводится в кадр управляющей программы в соответствии с описанным в регистре форматом.

Например, пусть предыдущее значение регистра <**F**> равно 200:

- Шаблон:

```
G_INTERP[1] X[XT] Y[YT] Z[ZT] F[200]
```

- Кадр УП:

```
G1 X100.100 Y-245.100 Z-010.560
```

Так как старое и новое значение регистра <**F**> совпали, то при формировании кадра управляющей программы идентификатор и значение регистра <**F**> не вывелось.

3.1.3 Модификатор

В некоторых случаях требуется вывести новое значение регистра, вне зависимости от того изменилось ли оно, или изменить значение регистра и не выводить его в текущем кадре. Для этого предназначены модификаторы <**On**> и <**Off**>.

Модификатор указывается в квадратных скобках через запятую после значения.

Модификатор <**On**>. При указании этого модификатора значение регистра обязательно попадёт в кадр управляющей программы.

Например, пусть предыдущее значение регистра <**G_PLANE**>

равно 17:

- Шаблон:

```
G_PLANE[17, On] X[XT] Y[YT]
```

- Кадр УП:

```
G17 X100.123 Y200.456
```

Для вывода текущего значения регистра (без занесения) допустимо использование модификатора <**On**> без подставляемого значения. Например:

- Шаблон:

```
G_PLANE[On] X[CLD.X] Y[CLD.Y]
```

- Кадр УП:

```
G17 X100.123 Y200.456
```

Модификатор <**Off**>. При указании этого модификатора в регистр занесётся значение, но в кадр управляющей программы регистр не попадёт. Например:

- Шаблон:

```
G49G80M5M9 G_PLANE[17,Off]
```

- Кадр УП:

```
G49G80M5M9
```

Для подавления вывода текущего значения регистра (без занесения) допустимо использование модификатора <**Off**> без подставляемого значения.

3.1.4 Выражение

Наравне со значением в шаблон можно подставлять выражение. Под выражением понимается точное описание операций, результатом которых является числовое значение. Математические выражения аналогичны математическим формулам.

Синтаксически математическое выражение представляет собой комбинацию чисел, числовых переменных и числовых функций, разделенных знаками математических операций и круглыми скобками. Простейшими случаями математического выражения являются число и числовая переменная.

Реализованы следующие операции:

- <**+**> – сложение;
- <**-**> – вычитание;
- <*****> – умножение;
- <**/**> – деление
- <**^**> – возведение в степень.

Два знака арифметических действий нельзя располагать рядом.

Кроме этих операций, в математических выражениях допустимы следующие стандартные функции:

- <**SIN(x)**> – синус угла x (град);
- <**COS(x)**> – косинус угла x (град);
- <**TAN(x)**> – тангенс угла x (град);
- <**ATN(x)**> – арктангенс x в град;
- <**ASIN(x)**> – арксинус x в град;
- <**ACOS(x)**> – арккосинус x в град;
- <**SQR(x)**> – квадратный корень из x;
- <**ABS(x)**> – вычисление абсолютной величины x;
- <**SGN(x)**> – вычисление знака x;
- <**ROUND(x, y)**> – округление x до y знаков после запятой;
- <**LOG(a, b)**> – логарифм b по основанию a;
- <**LG(x)**> – десятичный логарифм x;
- <**LN(x)**> – натуральный логарифм x.

В выражении так же доступны предопределённые переменные и функций, а так же все параметры текущей технологической команды, передаваемые через массив <**CLD**>. Например:

- Шаблон:

```
X[2+5/2+2*(sin(45))] Y[CLD.Y*2] F[FEED]
```

- Кадр УП:

```
X005914 Y001234 F200
```

В этом примере выражение первого элемента: 2+5/2+2*(sin(45)) равно 5.91421356237697. При выводе в кадр управляющей программы среди регистров был найден идентификатор <**X**> и формат вывода значения был взят из свойств регистра. В выражении второго элемента присутствует параметр технологической команды, переданный через <**CLD**> массив – <**CLD.Y**> (его значение 0.617) умноженный на 2. В третьем элементе в качестве параметра принимает участие предопределённая

переменная <**FEED**>, в которой содержится текущее значение подачи.

3.1.5 Вложенный шаблон

Иногда требуется вывести значение регистра, только если значение другого регистра изменилось. В таких случаях используется вложенный шаблон. Например:

- Шаблон:

```
Z[CLD.Z, G[43]]
```

- Кадр УП:

```
G43 Z0.123
```

Значение регистра <**Z**> было изменено и в регистр <**G**> занесено значение 43.

Во вложенном шаблоне допустимо использовать модификаторы, выражения и другие вложенные шаблоны. Регистры вложенного шаблона помещаются в кадр управляющей программы в соответствии с положением в списке регистров. Например:

- Шаблон:

```
X[CLD.X] Y[CLD.Y] Z[CLD.Z, G[43]]
```

- Кадр УП:

```
G43 X6.141 Y-4.234 Z0.123
```

При формировании кадра управляющей программы изменённый регистр <**G**>, в соответствии со списком регистров, был помещён в начало кадра.

3.1.6 Разделители элементов шаблона

В общем случае строка шаблона состоит из нескольких элементов. Для удобства написания между элементами можно вставлять пробелы. Эти пробелы никак не влияют на формат кадра управляющей программы.

Для управления пробелами между словами в кадре управляющей программы можно воспользоваться признаком <**Расставлять пробелы между командами**> (признак доступен для изменения

на форме <**Параметры системы ЧПУ**>, главное меню **Вид** –> **Станок**). Если признак включен (т.е. расставлять пробелы), то пробел добавляется, даже если между элементами отсутствует разделитель. Например:

- Шаблон:

```
" G[2] X[100.101]Y[234.89] Z[45.67]"
```

- Кадр УП:

```
" G2 X100.101 Y243.890 Z045.670"
```

В кадре управляющей программы между элементами 2 (X100.101 и 3 (Y243.890) добавлен пробел. Количество пробелов перед элементом 1 и между элементами 1 и 2, 3 и 4 осталось без изменения.

Если признак <**Расставлять пробелы между командами**> выключен, то удаляются все пробелы между элементами, независимо от их количества. Например:

- Шаблон:

```
G_INTERP[INTERP] X[CLD.X] Y[CLD.Y] Z[CLD.Z]
```

- Кадр УП:

```
G1X-49.47Y-6.513Z.033
```

В кадре управляющей программы удалены все пробелы.

3.1.7 Занесение значений в переменные

При необходимости из шаблона, во все предопределённые переменные и переменные подпрограммы <**COMMON**> можно занести требуемые значения. Значения задаются как для числовых, так и для строковых переменных. Причём строковым переменным можно присваивать только константы, в то время как числовым разрешается присваивать любые выражения, которые могут содержать другие переменные или даже функции.

Задавать значения переменных можно в любой строке шаблона, при этом для данной строки кадр сформирован не будет, а в соответствующие переменные будут занесены требуемые значения. Разделителем между переменными является знак <**;**>. Например:

- Шаблон:

```
INTERP=0
XP=XT
```

```
YP=YT-Y1+2
CLDATA$="текст"
G_INTERP[INTERP] X[XT] Y[YT] Z[ZT] ([CLDATA$])
```

- Кадр УП:

```
G0 X100.122 Y231.567 Z010.546 (текст)
```

3.1.8 Отложенные шаблоны

Иногда возникает ситуация при которой необходимо использовать текущие параметры текущей команды CLData при обработке другой команды. Вместо сохранения необходимых параметров во временные переменные в этом случае используются отложенные шаблоны. Текущие параметры таких шаблонов сохраняются, но их выполнение откладывается до обработки требуемой команды CLData. При выполнении отложенного шаблона все значения текущих параметров заменяются сохранёнными значениями.

Отложенный шаблон должен быть заключён в фигурные скобки: <**{** > и <**}**>, внутри фигурных скобок обязательно наличие имени одной из команд CLData, по приходу которой отложенный шаблон выполняется. Строки, расположенные до названия команды CLData, выполняются перед выполнением команды, строки, расположенные после названия команды CLData, выполняются после выполнения команды CLData. Например:

- Шаблон команды <**SafPos**>:

```
{
G_INTERP[0] X[XT] Y[YT] Z[CLD.Z]
G_FUNC[28] X[CLD.X] Y[CLD.Y]
LoadTL
G_INTERP[1] G_FUNC[29] G_LENGTHCOMPENS[43] X[XT] Y[YT]
G_LENGTHCOMPENS[49] Z[ZT]
}
```

- Шаблон команды <**LoadTL**>:

```
H[CLD.N] T[CLD.N]
```

- Шаблон команды <**AbsMov**>:

```
X[CLD.X] Y[CLD.Y] Z[CLD.Z]
```

- Команды CLData:

```
GOTO.abs X 20.0000,Y 20.0000,Z 20.0000
SAFPOS X 10.0000,Y 11.0000,Z 12.0000,N 0
LOADTL N 1,X 0.0000,Y 0.0000,Z 0.0000,D 1.0000,M 0,K 0,L 0.0000,
       P 0.0000,A 0.0000, R 0.0000,PLANE XY(33),Dur 0.0000
```

- Кадр УП:

```
X20.0000 Y20.0000 Z20.0000
```

```
G0 Z0.0000
G28 X0.0000 Y0.0000
H1 T1
G1 G29 G43 X20.0000 Y20.0000
G49 Z20.0000
```

В данном случае в момент выполнения команды <**SafPos**> создаётся отложенный шаблон. Значения параметров <**CLD.X**>, <**CLD.Y**>, <**CLD.Z**> команды <**SafPos**> сохраняются, и выполнение шаблона откладывается до обработки команды <**LoadTL**>. Значения переменных <**XT**>, <**YT**>, <**ZT**> не сохраняются, т.к. это не параметры команды CLData. Перед обработкой команды <**LoadTL**> выполняется шаблон:

```
G_INTERP[0] X[XT] Y[YT] Z[CLD.Z]
G_FUNC[28] X[CLD.X] Y[CLD.Y]
```

Затем выполняется команда

```
LoadTL: H[CLD.N] T[CLD.N]
```

Далее выполняется шаблон:

```
G_INTERP[1] G_FUNC[29] G_LENGTHCOMPENS[43] X[XT] Y[YT]
G_LENGTHCOMPENS[49] Z[ZT]
```

После чего действие отложенного шаблона завершается.

Если перед закрывающей фигурной скобкой отложенного шаблона указать идентификатор <**Keep**>, то отложенный шаблон будет вновь выполнен при обработке следующей требуемой команды CLData (в предыдущем примере это <**LoadTL**>). Например:

- Шаблон команды <**SafPos**>:

```
{
G_INTERP[0] X[XT] Y[YT] Z[CLD.Z]
G_FUNC[28] X[CLD.X] Y[CLD.Y]
LoadTL
G_INTERP[1] G_FUNC[29] G_LENGTHCOMPENS[43] X[XT] Y[YT]
G_LENGTHCOMPENS[49] Z[ZT]
Keep
}
```

В данном примере после однократного выполнения действие отложенного шаблона не прекращается, и при обработке второй и последующих команд <**LoadTL**> будет выполнен отложенный шаблон.

3.1.9 Использование массива <GMA> в шаблонах

Предопределенный массив <**GMA**> предназначен для удобного обращения с параметрами команды многокоординатного перемещения <**MULTIGOTO**> внутри процедуры обработки, а также при использовании шаблонов обработки этой команды. Массив <**GMA**> также как и массив <**CLD**> заполняется каждый раз перед обработкой команды. Но если массив <**CLD**> является универсальным и может использоваться для любой команды, то <**GMA**> следует использовать только для команды <**MULTIGOTO**>.

Каждый из элементов в массиве <**GMA**> всегда соответствует управляемой координате в кинематической схеме соответствующего станка. В элемент массива из технологической команды <**MULTIGOTO**> заносятся, например, текущее и предыдущие значения координаты, направление вращения для поворотных осей и др. Ниже приведен пример доступа к элементам массива из текста программы обработки технологической команды:

```
GMA["AxisAPos"].OutFlag
GMA["AxisAPos"].Vn
GMA["AxisAPos"].Vp
GMA["AxisAPos"].Axis
GMA["AxisAPos"].Reg
GMA["AxisAPos"].TurnCount
GMA["AxisAPos"].Dir
```

В данном примере <**AxisAPos**> – это строка, соответствующая имени какой-то конкретной координаты в схеме станка SprutCAM.

При использовании массива <**GMA**> в шаблонах синтаксис немного отличается от того, который используется в языке программирования в обработчиках технологических команд. Единственное отличие состоит в том, что ключевое слово <**GMA**> и квадратные скобки опускаются. Таким образом, внутри шаблона команды <**MULTIGOTO**> для доступа к свойству какой-либо координаты станка необходимо в кавычках указать имя координаты, а затем через точку имя свойства. Например, текущее значение координаты с именем <**AxisXPos**> может быть получено как <**"AxisXPos".Vn**>.

Доступ к другим свойствам элемента массива <**GMA**> осуществляется аналогично.

- <**"AxisName".OutFlag**> – наличие координаты в текущей команде <**MULTIGOTO**>.
- <**"AxisName".Vn**> – текущее значение координаты.
- <**"AxisName".Vp**> – предыдущее значение координаты.
- <**"AxisName".Axis**> – имя координаты в кинематической схеме станка.
- <**"AxisName".Reg**> – имя регистра, сопоставленного с координатой.
- <**"AxisName".TurnCount**> – количество полных оборотов при текущем изменении поворотной координаты.
- <**"AxisName".Dir**> – направление изменения поворотной

координаты.

Здесь <**AxisName**> – идентификатор координаты станка.

Приведенный ниже пример шаблона заносит из технологической команды <**MULTIGOTO**> в регистры <**X**>, <**Y**> и <**Z**> текущие значения координат <**AxisXPos**>, <**AxisYPos**> и <**AxisZPos**> соответственно.

```
X[“AxisXPos”.Vn] Y[“AxisYPos”.Vn] Z[“AxisZPos”.Vn]
```

3.2 Управление шаблоном

3.2.1 Переключатели шаблона

С каждым шаблоном связаны три переключателя:

1. <**Вывод предыдущего кадра**>. Включение признака приведёт к тому, что до обработки шаблона сформируется кадр, в который попадут все регистры, старые и новые значения которых не равны.
2. <**Опросить все регистры**>. Включение признака приведёт к тому, что генератор постпроцессоров помимо регистров, перечисленных в шаблоне, включит в кадр управляющей программы те регистры, старые и новые значения которых не совпадают. Если такие регистры существуют, а это может быть, например, если в предыдущей маске для одного из регистров был применён модификатор <**Off**>, то данный регистр занимает местоположение в кадре управляющей программы в соответствии с местом в списке регистров. Например:

- Перечень регистров:

```
G_INTERP, G_PLANE, X, Y, Z
```

- Шаблон <**Plane**>:

```
G_PLANE[ARCPLANE,Off]
```

- Шаблон <**AbsMov**>:

```
G_INTERP[1] X[CLD.X] Y[CLD.Y] Z[CLD.Z]
```

- Команды CLData:

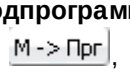
```
PLANE XY(33)
GOTO.abs X100 Y100 Z100
```

- Кадр УП:

```
G1 G17 X100.000 Y100.000 Z100.000
```

3. <**Вывод текущего кадра**>. При выключении признака в регистры, перечисленные в шаблоне, будут занесены значения, но кадр управляющей программы сформирован не будет;

3.2.2 Преобразование шаблона в подпрограмму

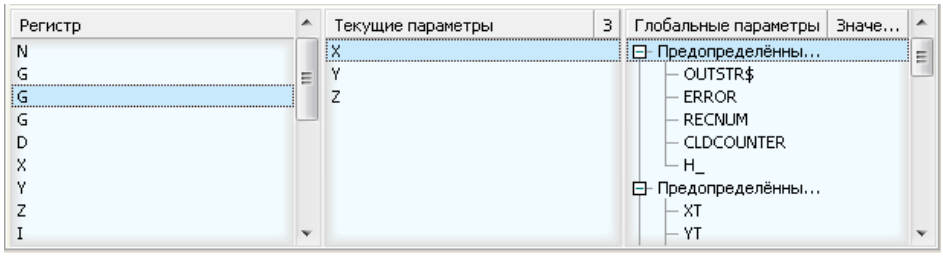
При редактировании шаблона может возникнуть ситуация, когда возможностей шаблона недостаточно. Для того чтобы использовать более мощный и гибкий механизм подпрограмм, без ручного переделывания шаблона, предназначена функция <**Преобразовать шаблон в подпрограмму**>. Преобразование производится с помощью кнопки 

, которая находится на панели, рядом с переключателями шаблона.

Для отмены преобразования необходимо перенести, используя буфер обмена, строку шаблона из редактора подпрограмм в редактор шаблонов и выставить соответствующие переключатели шаблона.

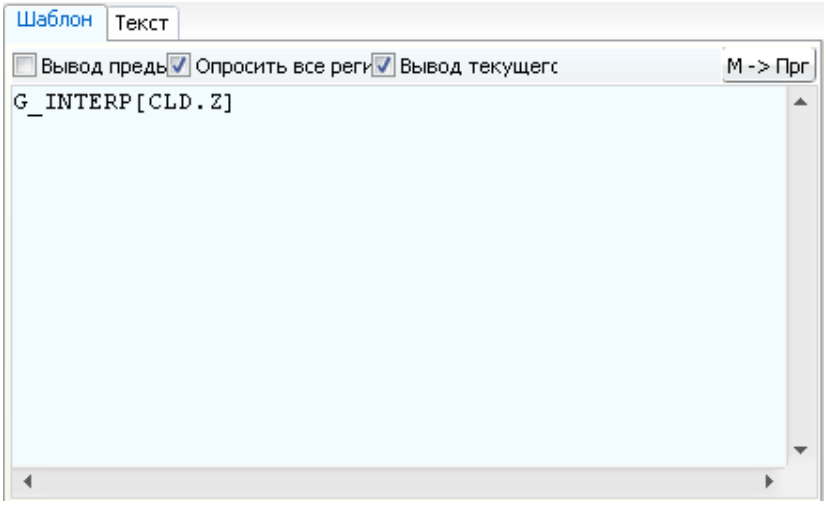
3.2.3 Визуальные средства формирования шаблона

Для удобного формирования шаблона предназначены три списка, находящихся под редактором шаблонов: перечень регистров, перечень текущих параметров CLData и перечень глобальных переменных, предопределённых переменных и предопределенных функций.



При двойном щелчке на позиции в перечне регистров в шаблон, в место положения курсора, подставляется имя регистра, а при отсутствии имени его идентификатор. Затем добавляются символы <**[**> и <**]**>, а курсор перемещается между ними.

При двойном щелчке на позиции, в перечне текущих параметров и перечне переменных и функций, в шаблон, в место положения курсора, подставляется имя текущего CLData параметра или переменной. К имени CLData параметра добавляется префикс < **CLD.**>. Если после имени CLData параметра или переменной есть символ <**]**> то курсор перемещается за квадратную скобку.



4 Описание языка

4.1 Основные понятия

4.1.1 Условные обозначения

При описании форматов операторов в данном документе применяются следующие правила и условные обозначения:

- Прописные буквы используются для обозначения зарезервированных слов;
- Строчные буквы используются для обозначения имен переменных, числовых значений или арифметических выражений;
- Языковые конструкции заключаются в пару символов **<**…**>**;
- Необязательные части операторов заключаются в фигурные скобки **{**… **}**;
- Возможные варианты частей операторов, которые являются альтернативными, разделяются знаком **|**.

4.1.2 Программы обработки технологических команд, комментарии в программе

Исходными данными для генерации управляющих программ являются загруженные в постпроцессор файлы, содержащие последовательность технологических команд. Упрощенно процесс генерации управляющей программы можно представить следующим образом:

1. Считывается очередная команда из файла технологических команд.
2. Анализируется код команды, в соответствии с параметрами команды заполняется предопределённый массив <**CLD**> и предопределённые переменные.
3. Если активизирована программа обработки технологической команды, то вызывается программа обработки этой команды. В ней обычно содержится программный код, который формирует кадры управляющей программы (УП), соответствующие данной команде CLData.
4. Если активизирован шаблон для этой команды, то формируется выходная строка УП по шаблону.

Программы обработки технологических команд пишутся на специальном проблемно-ориентированном языке и могут содержать математические выражения и функции, операторы ввода/вывода, условные операторы, циклы, операторы перехода, вызовы других программ, операторы формирования кадров управляющей программы.

Каждая программа обработки технологической команды начинается с заголовка, состоящего из слова <**PROGRAM**> и имени программы, а заканчивается словом <**END**>. Имя программы совпадает с именем команды, которую она обрабатывает. При вызове программы обработки параметры команды передаются через предопределенный массив <**CLD**> и

оператор <**Cmd**>. Значения параметров для различных команд CLData описаны в главе <Описание технологических команд>.

Текст программы может содержать операторы и комментарии. В одной строке допустимо написание нескольких операторов. При этом они должны быть разделены символом <**;**>. Перенос части оператора на следующую строку допустим в любом месте при условии, что оператор является логически незавершенным. Для введения в программу пояснительных записей служат комментарии. Комментарием является любой текст, расположенный после символа <**!**> до конца строки. Перенос комментариев недопустим.

Первой выполняемой программой является программа <**COMMON** >. Затем выполняется команда <**PARTNO**>, последней – программа обработки команды <**FINI**>.

Особенностью программы <**COMMON**> является определение в ней глобальных переменных, значения которых доступны во всех программах. Для собственных нужд любая программа может использовать переменные, не объявленные в <**COMMON**> и, поэтому, не являющиеся общедоступными. Такие переменные являются локальными.

4.1.3 Понятие подпрограмм

<**Подпрограмма**> (англ. subprogram) – поименованная часть программы, содержащая описание определённого набора действий. Подпрограмма может быть многократно вызвана из разных частей программы. Подпрограммы изначально появились как средство оптимизации программ по объёму занимаемой памяти – они позволили не повторять в программе идентичные блоки кода, а описывать их однократно и вызывать по мере необходимости. К настоящему времени данная функция подпрограмм стала вспомогательной, главное их назначение – структуризация программы с целью удобства её понимания и сопровождения.

Выделение набора действий в подпрограмму и вызов её по мере необходимости позволяет логически выделить целостную подзадачу, имеющую типовое решение. Такое действие имеет ещё одно (помимо экономии памяти) преимущество перед повторением однотипных действий: любое изменение (исправление ошибки, оптимизация, расширение функциональности), сделанное в подпрограмме, автоматически отражается на всех её вызовах, в то время как при дублировании каждое изменение необходимо вносить в каждое вхождение изменяемого кода.

Даже в тех случаях, когда в подпрограмму выделяется однократно производимый набор действий, это оправдано, так как позволяет сократить размеры целостных блоков кода, составляющих программу, то есть сделать программу более понятной и обозримой.

Подпрограммы пишутся на том же языке, что и программы обработки технологических команд, они вызываются из программ с помощью оператора <**CALL**>. Каждая подпрограмма начинается с заголовка, состоящего из слова <**SUB**> и имени программы, а заканчивается словом <**SUBEND**>. Имя подпрограммы может быть любым, не совпадающим с именами программ и уже имеющимися подпрограммами. Внутрь подпрограмм можно передавать

параметры. Список формальных параметров задается в строке определения подпрограммы сразу после ее имени.

Синтаксис определения подпрограмм:

```
SUB <ИмяПодпрограммы>{(<Параметр1, Параметр2,...>)}
{
< Код подпрограммы >
}
SUBEND
```

Подпрограммы так же, как и программы, могут содержать математические выражения и функции, операторы ввода/вывода, условные операторы, циклы, операторы перехода, вызовы подпрограмм, операторы формирования кадров управляющей программы.

Ниже приведен пример подпрограммы, которая выполняет пересчет линейных координат на плоскости X и Y в линейную и поворотную координаты на цилиндре X и C.

Пример:

```
sub ConvertXYtoXC(XIn, YIn, COut: Real)
  HalfX: Real

  HalfX = XIn * 0.5 ! Radial value of X
  if abs(YIn)>0.000001 then begin
    if YIn>0 then begin
      COut = atn(-HalfX/YIn) + 180
    end else begin
      if XIn>=0 then COut = atn(-HalfX/YIn)
                else COut = atn(-HalfX/YIn) + 360
    end
  end else begin
    if HalfX>=0 then COut = 90
                else COut = 270
  end
subend
```

4.1.4 Понятие оператора языка

Оператор является функциональной единицей языка. Форматы операторов однозначно определяются множеством допустимых конструкций языка и способов задания. Примеры операторов на языке генератора постпроцессоров:

```
PRINT “символ” ! печать литерной строки "символ"
JUMP 1         ! переход на метку номер 1
```

Операторы входного языка системы могут включать в себя идентификаторы, числа, литерные строки и служебные символы.

4.1.5 Набор символов

При написании операторов языка используются следующие символы:

- <**a..z**>, <**A..Z**> – прописные и строчные буквы латинского алфавита;
- <**а..я**>, <**А..Я**> – прописные и строчные буквы русского алфавита;
- <**0..9**> – арабские цифры от 0 до 9;
- <_> – знак подчеркивания;
- < > – пробел;
- <**,**> – запятая;
- <**=**> – равно;
- <**;**> – точка с запятой;
- <**:**> – двоеточие;
- <**[**> – левая квадратная скобка;
- <**]**> – правая квадратная скобка;
- <**“**> – двойные кавычки;
- <**(**> – левая круглая скобка;
- <**)**> – правая круглая скобка;
- <**+**> – плюс;
- <**-**> – минус;
- <*****> – звездочка;
- <**/**> – наклонная черта;
- <****> – обратная наклонная черта;
- <**^**> – символ-указатель;
- <**$**> – знак доллара;
- <**<**> – левая угловая скобка;
- <**>**> – правая угловая скобка;
- <#> – не равно;
- <**.**> – точка;
- <@> – А-коммерческое.

В комментариях и литерных строках допускаются любые символы.

- <**Идентификатор**> – непрерывный набор букв латинского и русского алфавитов, цифр и символов <_>, <@> и <**$**>, начинающийся с буквы. Большие и малые буквы в идентификаторах не различаются;
- <**Число**> – непрерывная последовательность цифр, содержащая не более одной десятичной точки. Перед числом допустимы знаки <**+**> или <**-**>. По умолчанию число принимается положительным;
- <**Литерные строки**> – произвольная последовательность

символов, заключенная в двойные кавычки;

4.1.6 Переменные

Переменная – проименованная область памяти, имя которой можно использовать для осуществления доступа к данным, находящимся в переменной. Иными словами переменная – это имя, с которым может быть связано некоторое значение.

Переменные принято делить на типы. Тип переменной определяет множество значений, которые могут быть ей присвоены и операции, которые могут быть с нею произведены. Используемые в языке переменные делятся на следующие типы:

- <**Integer**> (целые) – используется для представления целых чисел. Переменная целого типа имеет диапазон значений от -2147483648 до 2147483647.
- <**Real**> (вещественные) – предназначены для хранения чисел, имеющих дробную часть. Переменная вещественного типа имеет следующий диапазон значений: $-2.9*10^{39}..1.7*10^{38}$ и имеет 11-12 значащих разрядов.
- <**String**> (строковые) – представляют из себя последовательность символов длиной до ~231.
- <**Array**> (массивы) – в отличие от приведенных выше типов, массивы представляют собой сложный тип данных. Если переменные простого типа могут ссылаться только на одно значение, то массивы являются ссылками на последовательность (таблицу) значений одного из простых типов.

По области видимости принято различать локальные и глобальные переменные. Переменные, доступные только в конкретной программе обработки или подпрограмме называются локальными. Переменные, к которым можно получить доступ из всех обработчиков или подпрограмм называются глобальными. Специально для объявления глобальных переменных предназначена программа <**COMMON**>. Все переменные, объявленные в ней являются глобальными.

Хороший стиль программирования предполагает определение в самом начале используемых в программе переменных. Объявление переменной состоит из указания имени переменной, двоеточия и указания типа переменной:

<**Имя переменной**>: <**Тип переменной**>

Здесь:

- <**Имя переменной**> – уникальный в пределах области видимости идентификатор переменной – непрерывный набор букв латинского и русского алфавитов, цифр и символов <_>, <@> и <**$**>, начинающийся с буквы. Большие и малые буквы в идентификаторах не различаются. Имена переменных не могут совпадать с именами операторов, программ обработки технологических команд, подпрограмм и другими стандартными языковыми конструкциями.

- <**Тип переменной**> – одно из перечисленных выше названий типов: <**Integer**>, <**Real**>, <**String**>, <**Array**> (определение переменных типа <**Array**> требует указания дополнительной информации).

Ниже приведен пример объявления и использования переменных.

Пример:

```
sub Sub1
  nn: Integer  ! Объявление переменной целочисленного типа
  xx: Real     ! Объявление переменной вещественного типа
  yy: Real     ! Объявление переменной вещественного типа
  rr: Real     ! Объявление переменной вещественного типа
  ss: String   ! Объявление переменной строкового типа

  nn = 2       ! Присвоение переменной значения
  xx = 7.43    ! Присвоение переменной значения
  yy = 12.6    ! Присвоение переменной значения
  rr = nn * sqr(xx^2 + yy^2) ! Использование переменных в выражениях
  ss = "Результат = "        ! Присвоение переменной значения
  print ss, rr ! Вывод значений переменных в окно отладочной информации
subend
```

4.1.7 Массивы

Массив представляет собой структуру данных, позволяющую хранить под одним именем совокупность данных определенного типа. Массив характеризуется своим именем, типом хранимых элементов, размером (числом хранимых элементов) и нумерацией элементов. Можно сказать, что массив представляет собой таблицу.

Индекс элемента массива	Значение элемента массива
1	Значение1
2	Значение2
...	...
N	ЗначениеN

Здесь N – размер массива (количество элементов в массиве). Максимальное значение размера массива не ограничено, но не рекомендуется необоснованно использовать большие значения, т. к. при этом расходуется оперативная память. Минимальным значением индекса массива является единица. Максимальное значение индекса массива ограничено размером массива.

В настоящее время система поддерживает только одномерные массивы. Это значит, что одному индексу элемента может соответствовать только одно значение. В качестве значений элементов массива могут быть значения одного из поддерживаемых простых типов: целые числа (<**Integer**>), вещественные числа (<**Real**>), символьные строки (<**String**>).

Использование массивов допустимо везде, где допустимо использование переменных соответствующего типа. Для обращения к любому элементу массива указывается идентификатор массива и индекс соответствующего элемента. В качестве индекса может выступать число или числовая переменная.

```
<ИдентификаторМассива>[<ИндексЭлемента>]
```

Объявление переменной как массива имеет вид:

<**Имя массива**>**: array** <**Размер**> **of** <**Тип элемента**>

Здесь:

- <**Имя массива**> – любой допустимый идентификатор;
- <**Размер**> – целое положительное число;
- <**Тип элемента**> – имя простого типа данных (<**Integer**>, <**Real**> или <**String**>).

Пример:

```
! Объявление массива вещественных чисел размером в 10 элементов
V: array 10 of Real

! Присвоение значения пятому элементу массива
V[5] = 10.67

! Использование массива в выражениях
i = 7
V[10] = 15*sqr((sin(V[i]))^2 + (cos(V[i+1]))^2)
```

Массив с заведомо известным количеством элементов называется статическим. Массив, количество элементов которого заранее не известно, называется динамическим. При объявлении такого массива размер массива можно опустить. Его размер будет определяться максимальным индексом заполненного элемента. Индексы динамических массивов, так же как и в статических, начинаются с единицы. Максимальное значение индекса динамического массива не ограничено, но не рекомендуется необоснованно использовать большие значения индекса, т.к. при этом расходуется оперативная память.

Массивы строк могут быть только динамическими. Ниже приведен пример объявления и использования динамического массива на примере массива строк.

Пример:

```
ArrayS: array of string ! Объявление массива
```

```
! Заполнение массива
ArrayS[1] = "Hello," ! Нет необходимости явно задавать количество
ArrayS[2] = " World" ! элементов. Выделение и освобождение
ArrayS[3] = "!"      ! памяти происходит автоматически

! Использование массива
output ArrayS[1] + ArrayS[2] + ArrayS[3]
```

4.1.8 Математические выражения

Под математическим выражением понимается точное описание операций, результатом которых является числовое значение. Математические выражения аналогичны математическим формулам.

Синтаксически математическое выражение представляет собой комбинацию чисел, числовых переменных и числовых функций, разделенных знаками математических операций и круглыми скобками. Простейшими случаями математического выражения являются число и числовая переменная.

Реализованы следующие операции:

- <**+**> – сложение;
- <**-**> – вычитание;
- <*****> – умножение;
- <**/**> – деление;
- <**^**> – возведение в степень.

Необходимо помнить, что два знака арифметических действий нельзя располагать рядом. Для управления последовательностью выполнения операций следует использовать круглые скобки <**(**> и <**)**>.

Кроме этих операций, в выражениях допустимо использование предопределенных функций и переменных.

4.1.9 Предопределенные переменные и функции

В языке используются следующие предопределенные переменные и функции:

- стандартные математические функции;
- функции работы со строками;
- служебные функции и переменные;
- функции и операторы работы с CLDATA.

Стандартные математические функции

В выражениях допустимо использовать следующие стандартные математические функции.

- <**SIN(x)**> – синус угла x (град).

- <**COS(x)**> – косинус угла x (град).
- <**TAN(x)**> – тангенс угла x (град).
- <**ATN(x)**> – арктангенс x в град.
- <**ASIN(x)**> – арксинус x в град.
- <**ACOS(x)**> – арккосинус x в град.
- <**SQR(x)**> – квадратный корень из x.
- <**ABS(x)**> – вычисление абсолютной величины (модуля) x.
- <**SGN(x)**> – вычисление знака x. Возвращает "-1", если x меньше нуля, "1", если x больше нуля и "0", если x равно нулю.
- <**ROUND(x, y)**> – округление x до y знаков после запятой.
- <**LOG(x, y)**> – логарифм y по основанию x.
- <**LG(x)**> – десятичный логарифм x.
- <**LN(x)**> – натуральный логарифм x.

Здесь <**x**> и <**y**> – аргументы функции, представляющие собой число, переменную, математическую функцию или математическое выражение.

Функции работы со строками

В системе реализованы следующие функции работы с символьными строками.

- <**CHR(x)**> – возвращает символ, соответствующий номеру <**x**> (ASCII-коду).
- <**ORD(Str)**> – возвращает номер (ASCII-код) первого символа строки <**Str**>.
- <**LEN(Str)**> – длина литерной строки <**Str**>; определяет количество символов в строке.
- <**POS(Pat, Str)**> – позиция начала литерной строки <**Pat**> в строке <**Str**>. Результатом функции является позиция первого вхождения подстроки <**Pat**> в строке <**Str**>. Если указанная подстрока не входит в строку <**Str**>, то функция возвращает нулевой результат.
- <**NUM(Str)**> – превращение строки в число; осуществляет преобразование строки <**Str**> в вещественное значение; при этом, если в строке <**Str**> встречаются символы, недопустимые в числе, то при работе функции происходит ошибка и функция возвращает нулевой результат.
- <**STR(n)**> – преобразует число <**n**> в строку.
- <**COPY(Str, n, m)**> – выделение из строки <**Str**> <**m**> символов, начиная с <**n**>.
- <**UPCASE(Str)**> – функция переводит все символы строки в верхний регистр.
- <**REPLACE(<Строковая переменная>, <Искомая строка>,**

<Строка для замены>)> Оператор замены подстроки в строке <REPLACE>.

Здесь приняты следующие обозначения.

- <**Str**>, <**Pat**> – литерные строки, переменные или строковые выражения. Строковое выражение представляет собой произвольную комбинацию литерных строк, литерных переменных и литерных функций, разделенных знаком " + " и круглыми скобками.
- <**n**>, <**m**> – числовые переменные или числа.

Служебные функции и переменные

В системе имеется ряд вспомогательных предопределенных переменных и функций, использование которых позволяет облегчить процесс написания постпроцессоров.

- Функции и операторы работы с CLData.
- <**Регистры**>. Каждому регистру в постпроцессоре соответствует переменная, имя которой совпадает с именем регистра. В переменной хранится текущее значение регистра. Тип регистровой переменной – вещественный.
- <**Старые значения регистров**>. До вывода текущего кадра можно получить старые значения регистров из переменных, идентификаторы которых есть идентификаторы соответствующих регистровых переменных со знаком <@> в конце. Например, <**X**@> – старое значение регистра <**X**>.
- <**Массив CLD**>. Имеется предопределенный массив вещественных чисел <**CLD**>. Он предназначен для хранения числовых параметров текущей обрабатываемой технологической команды CLData.
- В переменной <**RecNum**> хранится количество параметров текущей технологической команды, которые содержатся в массиве <**CLD**>. То есть, при вызове программы обработки технологической команды в переменную заносится максимальный номер значащего элемента массива <**CLD**>.
- <**Массив GMA**> для работы с параметрами команды многокоординатного перемещения <**MULTIGOTO**>.
- Переменная <**CLData$**>. Если параметром технологической команды является не массив вещественных данных, а текстовая строка, то при вызове соответствующей программы обработки строка передается через переменную <**CLData$**>. Строка хранится до тех пор, пока не будет переопределена новой командой. Тип – строковый.
- Переменная <**OutStr$**>. В переменной содержится текст текущего кадра управляющей программы после команд <**FORMBLOCK**> или <**OUTBLOCK**>. Тип – строковый.
- Переменная <**BlockStep**> содержит значение приращения, используемого при автоматическом формировании номера

кадра. Автоматическая нумерация кадров настраивается в окне <**Общие параметры**>. При включенной автоматической нумерации перед формированием каждого следующего кадра к номеру кадра прибавляется величина <**BlockStep**>. Если присвоить данной переменной значение 0, то автоматическая нумерация кадра отключается.

- В строковой переменной <**NCName$**> содержится имя файла управляющей программы без расширения и без пути к файлу.
- Строковая переменная <**NCPath$**> содержит имя файла управляющей программы с расширением и полным путём к файлу.
- В строковой переменной <**SPPName$**> содержится имя файла постпроцессора без расширения и без пути к файлу.
- Строковая переменная <**SPPPath$**> содержит имя файла постпроцессора с расширением и полным путём к файлу.
- <**CurDate**> – возвращает строку, содержащую текущую дату.
- <**CurTime**> – возвращает строку, содержащую текущее время.
- Переменная <**Error**> – целочисленная переменная. Если при выполнении программы произошли какие-либо ошибки, то номер ошибки содержится в переменой <**Error**>.
- Переменные <**Xt**>, <**Yt**>, <**Zt**> – текущее положение инструмента. Тип – вещественный (переменные заполняются автоматически перед вызовом программ <**CIRCLE**>, <**FROM**>, <**ABSMOV**>, <**GOHOME**>).
- Переменные <**Xp**>, <**Yp**>, <**Zp**> – предыдущее положение инструмента. Тип – вещественный (заполняются автоматически перед вызовом программ <**CIRCLE**>, <**FROM**>, <**ABSMOV**>, <**GOHOME**>).
- Переменные <**Xc**>, <**Yc**>, <**Zc**> – координаты центра окружности (заполняются автоматически перед вызовом программы <**CIRCLE**>).
- Переменная <**Interp**> – текущая интерполяция (значение заносится автоматически перед вызовом программ <**CIRCLE**>, <**FEDRAT**>, <**ABSMOV**>, <**RAPID**>, <**GOHOME**>). Возможные значения:
  - 0 – быстрый ход,
  - 1 – линейная интерполяция,
  - 2 – круговая интерполяция по часовой стрелке,
  - 3 – круговая интерполяция против часовой стрелки.
- Переменная <**ToolRad**> – радиус инструмента (значение заносится автоматически перед вызовом программы <**LOADTL**>).
- Переменная <**ArcPlane**> – текущая плоскость круговой интерполяции (значение заносится автоматически перед вызовом программы <**PLANE**>), возможные значения:
  - 33 – XY,
  - 37 – YZ,

  - 41 – XZ,
  - 133 – YX,
  - 137 – ZY,
  - 141 – ZX.
- Переменная <**Feed**> – текущее значение подачи (значение заносится автоматически перед вызовом программы <**FEDRAT**>).
- Переменные <**TlComp**>, <**TrComp**> – номер корректора на длину и на радиус соответственно (заполняется перед вызовом программы <**CUTCOM**> и <**LOADTL**>).
- Переменные <**FromX**>, <**FromY**>, <**FromZ**> – координаты исходной точки (заполняется перед вызовом программы <**FROM**>).
- <**FlagIn**> возвращает:
  - 1 – если происходит внутренний обход следующего пересечения;
  - 0 – если внешний обход;
  - -1 – если тип обхода невозможно определить (например, совпадение направлений соседних кадров при линейной интерполяции).
- <**Cross**> – возвращает значение 0, если текущий элемент сопрягается с последующим, и 1 если пересекаются. В общем случае две прямые всегда пересекаются, прямая и окружность могут сопрягаться, либо пересекаются. Эта переменная может быть использована для реализации коррекции.
- <**NextToolNum**> – возвращает номер следующего инструмента.
- <**ToolChange**> – возвращает признак смены инструмента (последует ли далее в файлах CLData команда <**LOADTL**>):
  - 1 – смена инструмента ещё не была, но будет или в данный момент происходит;
  - 0 – смена инструмента была и ещё будет или происходит в очередной раз;
  - -1 – смена инструмента происходит в последний раз и далее смены инструмента не будет.

Работу функции можно продемонстрировать на следующем примере:

Последовательность команд CLData	Значение, возвращаемое функцией <**ToolChange**>
PARTNO	1
...	1
...	1
...	1
LOADTL	1
...	0
...	0
...	0

LOADTL	0
...	0
...	0
...	0
LOADTL	-1
...	-1
...	-1
...	-1
FINI	-1

Функции и операторы работы с CLData

В системе имеется ряд функций, переменных и операторов языка, позволяющих получить доступ к технологическим командам CLData и к их параметрам.

- Пердопределенный массив CLD
- Оператор Cmd
- Оператор CLDFile
- Предопределенный массив <GMA>
- Функция GetCLDStr
- Функция CurCode
- Функция NextCode
- Функция CLDCounter
- Функция CodeOfCmd
- Функция GetCLD
- Функция FindCld
- Функция GFindCld

Предопределенный массив CLD

Предопределенный массив <**CLD**> представляет собой массив вещественных чисел. Он предназначен для хранения числовых параметров текущей обрабатываемой технологической команды CLData. Перед вызовом обработчиков текущей команды ее параметры заносятся в массив <**CLD**>. Внутри обработчиков параметры команды могут быть проанализированы путем доступа к отдельным элементам массива и на основе анализа могут формироваться кадры управляющей программы.

Доступ к конкретному элементу массива <**CLD**> может быть осуществлен либо по индексу, либо по уникальному имени параметра.

Доступ по индексу ничем не отличается от стандартного обращения к элементам массива: <**CLD[i]**> – i-й элемент массива

– вещественное число, <**i**> – индекс элемента массива – целое положительное число. Индекс элемента массива <**i**> может сам являться переменной или выражением целого типа. Пример <**CLD [3]**>, <**CLD[n]**>, <**CLD[2*n+1]**> и т.д. Общее количество элементов в массиве CLD содержится в предопределенной переменной < **RecNum**>.

Доступ по имени производится следующим образом: <**CLD. Имя_параметра**>. Здесь <**Имя_параметра**> – уникальный идентификатор параметра в команде. Например, <**CLD.X**>, <**CLD. Mode**> и т.п. Имена параметров индивидуальны для каждой команды. Список параметров с указанием имен и подробным описанием приводится в приложении. Список имен параметров также доступен на панели <**Текущие параметры**> страницы < **Шаблон**> главного окна. Список параметров команды с именами также доступен на закладке <CLData> в нижней части главного окна.

Пример использования массива CLD.

```
program Circle
  if CLD.R > 0 then INTERP_ = 3
               else INTERP_ = 2 ! G3/G2
  X = cld[5]
  Y = cld[6]
  Z = cld[7]
  I = CLD.Xc
  J = CLD.Yc
  R = abs(cld.R)
  OutBlock
end
```

Приведенный выше обработчик команды CIRCLE, выводит в управляющую программу кадры перемещения по окружности G2 или G3 (по или против часовой стрелки) с указанием конечной точки (X, Y, Z), центра (I, J) и радиуса (R).

Доступ к параметрам команды CLData, в том числе и не числовым, может быть получен также при помощи оператора Cmd.

Оператор Cmd

Оператор <**Cmd**> предоставляет гибкий механизм для работы с текущей командой CLData. Он возвращает ссылку на текущую команду CLData, по которой можно запросить код, имя и параметры данной команды. Формат доступа к конкретным данным команды следующий.

<**Cmd.Code**> - возвращает уникальный числовой код команды CLData. Каждая команда однозначно идентифицируется таким кодом. Список кодов команд приведен в приложении.

<**Cmd.Name**> - возвращает текстовое имя команды либо параметра команды.

<**Cmd.Data**> - для отдельных команд CLData, которые имеют единственный строковый параметр (например **Comment**, **PartNo** и

др.), возвращает значение данного строкового параметра. Для остальных команд возвращается строковое представление команды, точно такое, какое отображается на закладке **CLData** главного окна.

<**Cmd.Data[Index]**> - представляет собой альтернативный способ доступа к массиву числовых параметров команды CLD. Возвращает вещественное значение элемента массива параметров с индексом **Index**.

Следующие ниже несколько конструкций позволяют получить доступ к параметрам команды CLData по имени параметра. В качестве имени параметра в них могут выступать не только строковые константы, как показано здесь, но и строковые переменные и выражения. Каждая команда CLData может характеризоваться собственным именованным набором параметров. Имена параметров с описанием для каждой команды приведены в приложении. Имена параметров можно увидеть на закладке **CLData** в нижней части главного окна.

<**Cmd.Str["ParameterName"]**> - возвращает значение параметра команды с именем **ParameterName** в виде строки.

<**Cmd.Int["ParameterName"]**> - возвращает значение параметра команды с именем **ParameterName** в виде целого числа (Integer).

<**Cmd.Flt["ParameterName"]**> - возвращает значение параметра команды с именем **ParameterName** в виде вещественного числа (Real, либо, как его еще называют, число с плавающей точкой - Float).

<**Cmd.Ptr["ParameterName"]**> - возвращает ссылку на параметр команды с именем **ParameterName**. Если параметра с таким именем в данной команде не существует, то оператор вернет значение 0, а иначе число отличное от нуля. По полученной ссылке через точку может быть получен доступ либо к значению данного параметра, либо к одному из дочерних параметров. Для этого нужно после точки снова указать одну из описаных здесь команд: Str, Int, Flt, Ptr, Item, ItemCount, Code, Name. Если при этом после инструкции не указывается имя параметра, то значит, что она относится к самому параметру. Например, инструкция Cmd.Ptr ["Parameter1"].Int означает взять значение самого параметра с именем <Parameter1> в виде целого числа. А запись Cmd.Ptr ["Parameter1"].Int["SubParameter1"] означает взять в виде целого числа значение дочернего параметра <SubParameter1> внутри параметра <Parameter1>.

<**Cmd.Item[Index]**> - возвращает ссылку на дочерний параметр по индексу. Индексация параметров начинается с 1.

<**Cmd.ItemCount**> - возвращает количество дочерних параметров в списке.

Параметры команд CLData могут представлять собой не только элементы простого типа (String, Integer, Real), но и более сложные структуры (комплексные параметры и массивы). Каждый параметр характеризуется своим уникальным именем и типом. Имеются следующие типы параметров.

- Строковые (String) - параметры данного типа имеют значение в виде текстовой строки и не могут иметь дочерних параметров.

- Целые числа (Integer) - параметры данного типа имеют значение в виде целого числа и не могут иметь дочерних параметров.
- Дробные числа (Double) - параметры данного типа имеют значение в виде дробного числа с плавающей точкой (Float) и не могут иметь дочерних параметров.
- Комплексные параметры (ComplexType) - параметры данного типа не имеют собственного значения, но они могут иметь дочерние свойства любого из описанных здесь типов. Данный тип необходим для объединения нескольких параметров в группу. Каждый из дочерних параметров обладает уникальным именем. Для доступа к дочерним параметрам необходимо последовательно через точку указать имена параметров от родительского до самого вложенного дочернего.
  Например, инструкция

  ```
  Cmd.Flt["MCS.OriginPoint.X"]
  ```

  означает взять у свойства <MCS> текущей команды дочерний параметр с именем <OriginPoint>, в котором найти дочернее свойство с именем <X>; результат вернуть в виде дробного числа.
- Массивы (Array). Параметры-массивы как и комплексные не обладают собственными значениями и могут иметь дочерние свойства. Однако дочерние свойства массивов, как правило, имеют общий тип и, следовательно, одинаковое имя. Поэтому доступ к дочерним свойствам массивов осуществляется не по имени, а по индексу. Для доступа к дочернему свойству массива с номером 4 нужно написать либо

  `Cmd.Ptr["ArrayName"].Item[4]`, либо
  `Cmd.Ptr["ArrayName(4)"]`.

  Здесь <ArrayName> - имя параметра-массива.
  Кроме индексного, доступ к дочернему параметру массива в некоторых случаях может быть осуществлен по ключу. Если у массива определен параметр Key, который задает имя поля-ключа в дочерних параметрах, то для получения доступа к конкретному элементу массива необходимо после имени массива в круглых скобках указать значение ключа. Например, инструкция

  ```
  Cmd.Ptr["ArrayName(KeyValue)"]
  ```

  означает взять у текущей команды параметр <ArrayName>, в котором найти дочернее свойство с ключевым полем равным <KeyValue>

Рассмотрим несколько примеров использования параметров команд CLData.

Пример 1. Внутри обработчика команды загрузки инструмента LoadTl в управляющую программу выводится строка с указанием

адреса T, номера инструмента и именем револьверной головки в скобках.

```
program LoadTl
  Output "T" + Cmd.Int["ToolID"] + " (" + Cmd.Str["RevolverID"] + ")"
end
```

Возможный результат вывода в УП: T3 (TopTurret)

Пример 2. Пример обработки команды MultiGoto с использованием различных способов доступа к элементам параметра-массива Axes.

```
program MultiGoto
  i: Integer
  AxisName: String

  for i = 1 to Cmd.Ptr["Axes"].ItemCount do begin
    AxisName = cmd.Ptr["Axes"].Item[i].Str["AxisID"]
    if AxisName="AxisXPos" then begin
      X = cmd.Ptr["Axes"].Item[i].Flt["Value"]
    end else
    if AxisName="AxisYPos" then begin
      Y = cmd.Ptr["Axes("+Str(i)+")"].Flt["Value"]
    end else
    if AxisName="AxisZPos" then begin
      Z = cmd.Flt["Axes("+Str(i)+").Value"]
    end else
    if cmd.Ptr["Axes(AxisCPos)"]>0 then begin
      C = cmd.Flt["Axes(AxisCPos).Value"]
    end
  end
  OutBlock
end
```

Пример 3. Пример обработки команды установки системы координат Origin. Здесь производится анализ значения свойства <OriginType> команды. Если оно равно нулю, то выводится команда выбора стандартной системы координат заготовки G54-G59. Номер системы координат берется из параметра <CSNumber>. В противном случае формируется команда переноса локальной системы координат G92 X Y Z. Величины смещений вдоль осей системы координат берутся из соответствующих дочерних свойств параметра <MCS.OriginPoint> различными способами.

```
program Origin
  if Cmd.Int["OriginType"]=0 then begin !G54-G59
    Output "G" + Cmd.Str["CSNumber"]
  end else begin                        !G92 X Y Z
    X = Cmd.Flt["MCS.OriginPoint.X"]
    Y = Cmd.Ptr["MCS.OriginPoint.Y"].Flt
    Z = Cmd.Ptr["MCS.OriginPoint"].Flt["Z"]
    Output "G92 X" + Str(X) + " Y" + Str(Y) + " Z" + Str(Z)
  end
```

```
end
```

Оператор CLDFile

Оператор <**CldFile**> предназначен для доступа к данным файлов CLData из постпроцессора. Он предоставляет информацию только для чтения, т.е. может находиться только справа от оператора присвоения. Синтаксис:

- <**CldFile.FileCount**> – возвращает количество загруженных файлов CLData;
- <**CldFile.CurrentFile**> – возвращает индекс текущего файла CLData (индексация начинается с 0);
- <**CldFile.CurrentCmd**> – возвращает индекс текущей команды в текущем файле CLData (индексация начинается с 1);
- <**CldFile[<FileIndex>].Enabled**> – возвращает 1, если файл с индексом <**FileIndex**> включен, 0 – файл выключен;
- <**CldFile[<FileIndex>].IsNCSub**> – возвращает 1, если файл является файлом подпрограммы, 0 – не является;
- <**CldFile[<FileIndex>].CmdCount**> – возвращает количество команд CLData в файле с индексом <**FileIndex**>;
- <**CldFile[<FileIndex>].Cmd[<CmdIndex>]**> – возвращает ссылку на команду CLData под номером <**CmdIndex**> в файле с индексом <**FileIndex**>. После этой команды через точку допустимо указывать любую из инструкций, описанных для оператора Cmd. Например:
  <**CldFile[<FileIndex>].Cmd[<CmdIndex>].Code**> – код команды CLData под номером <**CmdIndex**>;
  <**CldFile[<FileIndex>].Cmd[<CmdIndex>].Data**> – возвращает строковые данные команды CLData под номером <**CmdIndex**> (например, строку комментария). Если команда не содержит строковых данных, то выводится строковое представление команды.
  <**CldFile[<FileIndex>].Cmd[<CmdIndex>].Data [<CldIndex>]**> – значение элемента массива CLD с номером <**CldIndex**>;
  <**CldFile[<FileIndex>].Cmd[<CmdIndex>].Name**> - имя команды либо параметра команды CLData;
  <**CldFile[<FileIndex>].Cmd[<CmdIndex>].Str** | **.Int** | **.Flt** | **.Ptr**> - значение параметра команды соответствующего типа;
  <**CldFile[<FileIndex>].Cmd[<CmdIndex>].Item[Index]**>- дочерний параметр по индексу;
  <**CldFile[<FileIndex>].Cmd[<CmdIndex>].ItemCount**> - возвращает количество дочерних параметров.

Приведенный ниже пример выводит в окно отладочной информации номера и комментарии для всех используемых инструментов:

```
sub PrintAllTools
  i: Integer
  j: Integer
  for i = 0 to CLDFile.FileCount-1 do begin
    if (CldFile[i].Enabled>0) and (CldFile[i].IsNCSub=0) then begin
      for j = 1 to CldFile[i].CmdCount do begin
        ! Если это команда загрузки инструмента
        if CldFile[i].Cmd[j].Code = CodeOfCmd("LoadTl") then begin
          ! Выводим идентификатор револьверной головки
          Print "RevolverID = " + CldFile[i].Cmd[j].Str["RevolverID"]
          ! Выводим номер инструмента
          Print "Tool number = ", CldFile[i].Cmd[j].Data[1]
          ! Выводим комментарий инструмента
          if CldFile[i].Cmd[j+1].Code = CodeOfCmd("Comment") then
            Print "Tool comment: " + CldFile[i].Cmd[j+1].Data
        end
      end
    end
  end
subend
```

Предопределенный массив <GMA>

Предопределенный массив <**GMA**> предназначен для удобного обращения с параметрами команды многокоординатного перемещения <**MULTIGOTO**> внутри процедуры обработки, а также при использовании шаблонов обработки этой команды. Массив <**GMA**>, также как и массив <**CLD**>, заполняется каждый раз перед обработкой команды. Но если массив <**CLD**> является универсальным и может использоваться для любой команды, то <**GMA**> следует использовать только для команды <**MULTIGOTO**>.

Каждый из элементов в массиве <**GMA**> всегда соответствует управляемой координате в кинематической схеме соответствующего станка. Список возможных элементов массива <**GMA**> определяется для каждого станка однократно. Он может быть импортирован из кинематической схемы станка SprutCAM, либо заполнен вручную. Редактирование списка координат массива производится в окне <**Общие параметры**> на вкладке <**Параметры осей**>.

Добавление управляемых координат в список может производиться не только в режиме проектирования постпроцессора, но и в режиме исполнения путем доступа из кода процедур обработки, например команды <**PartNo**>. Синтаксис добавления управляемой координаты в список следующий:

```
GMA(<AxisName> {,<RegisterName>})
```

Здесь:

- <**AxisName**> – строковая переменная или константа, содержащая имя управляемой координаты.
- <**RegisterName**> – имя регистра, ассоциированного с управляемой координатой (необязательный параметр).

Пример добавления:

```
GMA(“AxisXPos”)
```

Обращение к элементам массива <**GMA**> следует производить по имени управляемой координаты.

```
GMA[<AxisName>]
```

Здесь <**AxisName**> – строковая переменная или константа, содержащая имя управляемой координаты. Например, для управляемой координаты с именем <**AxisXPos**> обращаться к массиву в процедуре обработки нужно так:

```
GMA[“AxisXPos”].Vn  ! Vn – текущее значение координаты
```

Примечание: *Доступ к элементам массива <**GMA**> может быть осуществлен не только из программы обработки технологической команды <**MultiGoto**>, но и из шаблона данной команды. Более подробно об использовании <**GMA**> в шаблонах написано в разделе <**Шалоны**>.*

Каждым элементом массива <**GMA**> является структура, имеющая следующие составляющие:

```
GMA[<AxisName>].OutFlag
GMA[<AxisName>].Vn
GMA[<AxisName>].Vp
GMA[<AxisName>].Axis
GMA[<AxisName>].Reg
GMA[<AxisName>].TurnCount
GMA[<AxisName>].Dir
```

- <**GMA[<AxisName>].OutFlag**> – поле имеет тип <**Integer**> (целое) и может содержать только два значения: 0 или 1. Данный признак идентифицирует наличие управляемой координаты с именем <**AxisName**> в списке текущей команды <**MULTIGOTO**>. Если в данной команде <**MULTIGOTO**> координата с именем <**AxisName**> присутствует, то <**OutFlag**> имеет значение 1, иначе <**OutFlag**> равен 0. Поле <**OutFlag**> может использоваться только для чтения.
- <**GMA[<AxisName>].Vn**> – поле имеет тип <**Real**> (вещественные) и содержит новое значение управляемой координаты с именем <**AxisName**>, т.е. то значение, которое содержится в текущей команде <**MULTIGOTO**>. Поле <**Vn**> может использоваться как для чтения значения, так и для записи в него другого значения (т.е. может находиться слева от оператора присвоения <**=**>).
- <**GMA[<AxisName>].Vp**> – элемент структуры имеет тип <**Real**

> (вещественные) и содержит значение управляемой координаты <**AxisName**>, присвоенное предыдущей командой <**MULTIGOTO**>. Элемент <**Vp**> может использоваться как для чтения значения, так и для записи в него другого значения (т.е. может находиться слева от оператора присвоения <**=**>).

- <**GMA[<AxisName>].Axis**> – поле имеет тип <**String**> (строка) и содержит имя соответствующей управляемой координаты. Может использоваться только для чтения.
- <**GMA[<AxisName>].Reg**> – поле имеет тип <**String**> (строка) и содержит имя регистра, связанного с соответствующей управляемой координатой. Может использоваться только для чтения.
- <**GMA[<AxisName>].TurnCount**> и <**GMA[<AxisName>].Dir**> – поля имеют смысл только для поворотных (периодических) осей станка. В системе SprutCAM величины углов поворотных осей могут не ограничиваться значениями от 0є до 360є, а, к примеру, могут принимать значения 764є, -135є, 1276є и т.п. Однако в некоторых стойках ЧПУ используются значения только из ограниченного диапазона: от 0є до 360є или от -180є до 180є. В настройках генератора постпроцессоров для любой из поворотных управляемых координат станка можно установить произвольный диапазон, к которому должны приводиться неограниченные значения, приходящие из системы SprutCAM. Поля <**TurnCount**> и <**Dir**> имеют тип <**Integer**> (целые) и используются для сохранения информации соответственно о количестве полных оборотов и о направлении изменения управляемой координаты после приведения ее к указанному ограничивающему диапазону. Поле <**Dir**> может принимать только два значения: +1 и -1. Значение +1 сигнализирует об изменении состояния координаты в положительном направлении (в направлении увеличения), а -1 – в отрицательном направлении (направлении уменьшения). <**TurnCount**> и <**Dir**> могут использоваться только для чтения.

Рассмотрим несколько примеров заполнения массива <**GMA**> параметрами команды <**MULTIGOTO**>. Предположим, что поворотная ось C (<**AxisCPos**>) ограничена диапазоном от 0є до 360є.

1. Команда:

```
MULTIGOTO COUNT 4, AxisXPos(18) 80, AxisYPos(20) 19.877,
          AxisZPos(22) -200, AxisCPos(24) 0
```

Массив <**GMA**> для команды <**MULTIGOTO**>

Имя управляемой координаты	**OutFlag**	**Vn**	**Vp**	**Axis**	**Reg**	**TurnCount**	**Dir**
AxisXPos	1	80	-//-//-	“AxisXPos”	“X”	-//-//-	-//-//-
AxisYPos	1	19.877	-//-//-	“AxisYPos”	”Y”	-//-//-	-//-//-
AxisZPos	1	-200	-//-//-	“AxisZPos”	”Z”	-//-//-	-//-//-

AxisCPos	1	0	-//-//-	“AxisCPos”	”C”	-//-//-	-//-//-

Неопределенность некоторых величин (помечены знаком «-//-//-») связана с отсутствием информации о предыдущем значении координаты.

2. Команда:

```
MULTIGOTO COUNT 1, AxisCPos(24) 270
```

Массив <**GMA**> для команды <**MULTIGOTO**>							
Имя управляемой координаты	**OutFlag**	**Vn**	**Vp**	**Axis**	**Reg**	**TurnCount**	**Dir**
AxisXPos	0	80	80	“AxisXPos”	”X”	-//-//-	-//-//-
AxisYPos	0	19.877	19.877	“AxisYPos”	”Y”	-//-//-	-//-//-
AxisZPos	0	-200	-200	“AxisZPos”	”Z”	-//-//-	-//-//-
AxisCPos	1	270	0	“AxisCPos”	”C”	0	+1

3. Команда:

```
MULTIGOTO COUNT 2, AxisZPos(22) 250, AxisCPos(24) -745
```

Массив <**GMA**> для команды <**MULTIGOTO**>							
Имя управляемой координаты	**OutFlag**	**Vn**	**Vp**	**Axis**	**Reg**	**TurnCount**	**Dir**
AxisXPos	0	80	80	“AxisXPos”	”X”	-//-//-	-//-//-
AxisYPos	0	19.877	19.877	“AxisYPos”	”Y”	-//-//-	-//-//-
AxisZPos	1	250	-200	“AxisZPos”	”Z”	-//-//-	-//-//-
AxisCPos	1	335	270	“AxisCPos”	”C”	2	-1

В последнем примере исходное значение оси <**AxisCPos(C)**> «-745 є» приводится к диапазону от 0є до 360є следующи образом:

```
TurnCount = |(Vn’ - Vp’) \ 360|
TurnCount = |(-745 -270) \ 360| = 2
Dir = Sgn(Vn’-Vp’)
Dir = Sgn(-745 -270) = -1
Vn = Vn’ – (Vn’ \ 360)*360, если Vn’ > 360
Vn = Vn’ – (Vn’ \ 360)*360 + 360, если Vn’ < 0
Vn = -745 – (-745\360)*360 + 360 = 335
```

Здесь <**Vn’**> – текущее неприведенное значение координаты, <**Vp’** > – предыдущее неприведенное значение координаты.

Рассмотрим простейший пример обработки технологической команды <**MULTIGOTO**> с использованием предопределенного массива <**GMA**>:

```
program MultiGoto
  ! Заносим в соответствующие регистры значения
  ! координат из текущей команды MULTIGOTO
  if GMA["AxisXPos"].OutFlag>0 then
    X = GMA["AxisXPos"].Vn
  if GMA["AxisYPos"].OutFlag>0 then
    Y = GMA["AxisYPos"].Vn
  if GMA["AxisZPos"].OutFlag>0 then
    Z = GMA["AxisZPos"].Vn
  if GMA["AxisCPos"].OutFlag>0 then
    C = GMA["AxisCPos"].Vn
  ! Выводим полученные значения в кадр УП
  OutBlock
end
```

Функция GetCLDStr

Функция <**GetCLDStr**> возвращает строку текстового представления текущей команды CLData с параметрами аналогичную той, которая отображается в окне текстового представления CLData.

Например, внутри обработчика команды AbsMov функция может вернуть строку, подобную следующей.

```
GOTO.abs   X 134.533, Y 99.684, Z 80
```

Функция CurCode

Функция <**CurCode**> возвращает код текущей команды CLData. Каждая команда CLData обладает уникальным числовым идентификатором - кодом. Коды всех существующих команд приведены в приложении. Пример использования функции внутри обработчика команды CIRCLE приведен ниже.

```
program Circle
  Print CurCode
end
```

В результате выполнения программы в окне отладочной информации выведется число 15000.

Код следующей команды CLData может быть получен функцией NextCode. Для получения кода произвольной команды по ее

текстовому наименованию следует использовать функцию CodeOfCmd.

Функция NextCode

Функция <**NextCode**> возвращает код технологической команды, которая расположена сразу вслед за текущей командой. Каждая команда CLData обладает уникальным числовым идентификатором - кодом. Коды всех существующих команд приведены в приложении. Для получения кода текущей команды можно использовать функцию CurCode. Для получения кода произвольной команды по ее текстовому наименованию следует использовать функцию CodeOfCmd.

Пример:

```
if NextCode=CodeOfCmd("Fini") then
  Output "M02"
```

Функция CLDCounter

Переменная <**CLDCounter**> содержит номер текущей команды в файле технологических команд. Для гибкого доступа к произвольной команде CLData по ее индексу в файле можно использовать оператор CLDFile.

Функция CodeOfCmd

Функция <**CodeOfCmd**> возвращает код команды CLData, имя которой передано в качестве параметра. Каждая команда CLData обладает уникальным числовым идентификатором - кодом. Коды всех существующих команд приведены в приложении. Функция имеет следующий формат.

```
CodeOfCmd(CommandName: String): Integer
```

CommandName - текстовое имя команды. Оно должно соответствовать одному из имен, которое отображается в списке команд слева в главном окне.

Возвращаемое значение можно, например, использовать с функциями <**CurCode**>, <**NextCode**>, <**GetCLD**>, <**CLDFile[i].Cmd[j].Code**>.

Пример:

```
if NextCode=CodeOfCmd("Fini") then
  Output "M02"
```

Функция GetCLD

Функция <**GetCLD**> предназначена для просмотра команд CLData в порядке их следования в файле. Просмотр возможен только в направлении от начала CLData к концу. Синтаксис:

```
{CldCode =} GetCLD(i: Integer; a: Array of Real)
```

Здесь:

- <**CldCode**> – возвращаемое функцией значение, числовой код команды CLData. Использование возвращаемого значения необязательно.
- <**i**> – относительный индекс запрашиваемой команды:
  - 0 – текущая команда,
  - 1 – следующая команда,
  - 2 – команда через одну от текущей и т.д.
- <**a**> – заранее объявленный динамический массив вещественных чисел (<**Array of Real**>, аналог массива <**CLD**>), в который будут занесены данные из команды CLData.

Например, пусть список команд CLDATA имеет вид, приведенный ниже, и допустим, что номер текущей команды равен 9:

```
 5: CUTCOM    ON(71),LENGTH(9) 2,X 0,Y 0,Z 0,N 0,K 0,M 0,LEFT(8)
 6: RAPID     N 10000
 7: GOTO.abs  X 134.533,Y 99.684,Z 80
 8: RAPID     N 10000
 9: GOTO.abs  X 134.533,Y 99.684,Z 74.400
10: FEDRAT    N 50,K 4,MMPM(315)
11: GOTO.abs  X 134.533,Y 99.684,Z 73.400
12: FEDRAT    N 200,K 0,MMPM(315)
13: PLANE     XY(33)
```

Обработчик технологической команды <**GOTO.abs**>:

```
program AbsMov
  a: Array of Real ! Массив должен быть заранее объявлен
  c: Integer       ! Объявление переменной целого типа
  c = GetCld(1, a) ! Читаем параметры следующей команды
  if c=CodeOfCmd(“FEDRAT”) then   ! Выведем значение первого элемента
    Print “FEDRAT.CLD[1]=”, Str(a[1]) ! массива CLD следующей команды
end
```

В результате выполнения примера в окно отладочной информации будет выведена следующая строка:

```
FEDRAT.CLD[1]=50.
```

Для доступа к параметрам произвольных технологических команд также можно воспользоваться оператором CLDFile.

Функция FindCld

Функция <**FindCld**> ищет команду с именем <**CmdName**> в текущем файле CLData, начиная с команды под номером <**StartIndex**>. <**StartIndex**> отсчитывается от начала текущего файла CLData. Если <**StartIndex**> не указан, то поиск начинается с команды, которая следует за текущей. Результат, возвращаемый функцией, равен индексу найденной команды в текущем файле CLData. Если команда не найдена, то результат равен -1. В качестве параметра <**Data**> должен передаваться заранее объявленный динамический массив вещественных чисел (<**Array of Real**>). В этот массив будут записаны параметры найденной команды CLData. Синтаксис:

```
N = FindCld({<StartIndex>, }<CmdName>, <Data>)
```

Например, пусть список команд CLDATA имеет вид, приведенный ниже, и допустим, что номер текущей команды равен 0.

```
0: PARTNO     "Bottle"
1: PPFUN      …..
2: COMMENT    "Roughing Waterline"
3: LOADTL     N 2,X 0,Y 0,Z 0,D 8….
4: SPINDL     ON(71),NO 397.887,K 0,MODE RPM(0)
5: CUTCOM     ON(71),LENGTH(9) 2,X 0,Y 0,Z 0,N 0,K 0,M 0,LEFT(8)
6: RAPID      N 10000
7: GOTO.abs   X 134.533,Y 99.684,Z 80
8: RAPID      N 10000
```

Обработчик технологической команды <**PartNo**>:

```
program PartNo
  V: Array of Real
  NCircle: Integer
  NGoto: Integer
  NCircle = FindCld("CIRCLE", V)
  NGoto = FindCld("GOTO.abs", V)
  if NGoto > NCircle then
    Print "Первое перемещение по дуге"
  else
```

```
    Print "Первое перемещение по прямой"
End
```

В результате выполнения примера в окно отладочной информации будет выведено сообщение "Первое перемещение по прямой".

Для доступа к параметрам произвольных технологических команд также можно воспользоваться оператором CLDFile.

Функция GFindCld

Функция <**GFindCld**> похожа на функцию FindCLD, но ищет команду с именем <**CmdName**> во всех файлах CLData, начиная с команды под номером <**StartIndex**>. <**StartIndex**> отсчитывается от начала первого файла CLData. Если <**StartIndex**> не указан, то поиск начинается с команды, которая следует за текущей. Результат, возвращаемый функцией, равен индексу найденной команды в сквозном списке команд CLData по всем файлам CLData. Если команда не найдена, то результат равен -1. В качестве параметра <**Data**> должен передаваться заранее объявленный динамический массив вещественных чисел (<**Array of Real**>). В этот массив будут записаны параметры найденной команды CLData. Синтаксис:

```
N = GFindCld({<StartIndex>, }<CmdName>, <Data>)
```

Для доступа к параметрам произвольных технологических команд также можно воспользоваться оператором CLDFile.

4.1.10 Предопределенная подпрограмма <Filter>

Во всех постпроцессорах присутствует предопределенная подпрограмма <**Filter**>, которая содержится в списке подпрограмм . Особенностью данной подпрограммы является то, что она вызывается сразу после формирования очередного кадра управляющей программы (УП) и непосредственно перед добавлением его в текст УП. Таким образом, внутри данной подпрограммы существует возможность корректировать каждый кадр УП в момент его окончательного вывода. Например, можно произвести замену символов, или добавить какие-либо метки в начало или конец строки и т.п.

Если текст подпрограммы <**Filter**> не был ранее задан то по умолчанию, система создает следующий прототип:

```
sub Filter(S: string)
subend
```

Здесь <**S**> – строка, содержащая сформированный кадр управляющей программы. Все изменения, произведенные внутри подпрограммы, в этой строке попадут в кадр УП.

Пример:

```
sub Filter(S: String)
  replace(S,"R1","L")    ! Заменяет "R1" на "L"
  replace(S,"R-1","R")   ! Заменяет "R-1" на "R"
  replace(S,"F0","FMAX") ! Заменяет "F0" на "FMAX"
subend
```

При выполнении приведенной подпрограммы для строки "17 L Z-10 R0 F0" на выходе строка примет вид "17 L Z-10 R0 FMAX".

4.2 Операторы

4.2.1 Оператор начала программы обработки технологической команды <PROGRAM>

Каждая программа обработки технологической команды начинается с заголовка, состоящего из слова <**PROGRAM**> и имени программы, а заканчивается словом <**END**>.

Формат:

```
PROGRAM ProgramName
  <оператор 1>
  ...
  <оператор N>
END
```

Описание:

Тело программы обработки технологической команды пишется на специальном проблемно-ориентированном языке и может содержать математические выражения и функции, операторы ввода/вывода, условные операторы, циклы, операторы перехода, вызовы подпрограмм, операторы формирования кадров управляющей программы.

Пример:

```
PROGRAM AbsMov
  FormBlock
  X = cld[1]
  Y = cld[2]
  if X <> LastX or y <> LastY then begin
    xs$ = Str(40*x)
```

```
    ys$ = str(40*y)
    if IsRapid> 0 then fs$ = "PU"
    else fs$ = "PD"
    output fs$ + xs$+","+ys$+";"
    LastX = X
    LAstY = Y
  end;
END
```

4.2.2 Оператор присваивания

Оператор предназначен для присваивания переменной значения выражения. В результате выполнения оператора присваивания переменные в ходе выполнения программы могут изменять свои значения.

Формат:

```
<числовая переменная> = <математическое выражение>
```

или

```
<строковая переменная> = <строковое выражение>
```

Описание:

Ключевым словом оператора присваивания является идентификатор переменной. Далее следует знак <**=**> (равно) и выражение, значение которого присваивается указанной переменной. Оператор присваивания указывает, что вновь вычисленное значение необходимо занести в ячейку памяти, идентифицируемую именем переменной. При этом необходимо обратить внимание на совпадение типов переменной и выражения в правой части оператора, т.е. числовой переменной должно присваиваться значение математического выражения, а строковой переменной – значение строкового выражения.

***Примечание**: При выполнении оператора сначала вычисляется значение правой части оператора, а затем полученное значение присваивается указанной переменной. Это правило позволяет использовать одну переменную в левой и правой частях оператора. При этом, в момент вычисления выражения будет участвовать старое значение переменной, а лишь после этого результат заносится в указанную переменную.*

Примеры:

```
! Использование оператора присваивания
Zet = 41
Ddel = Mn / Cos(Betta) * Zet
Number = Number + 5

! Формирование строк информации о станке
System$ = “ 2C42 “

! Занесение названия системы ЧПУ в System$
```

```
NAME$ = “ TEST PROGRAM “ + System$
```

4.2.3 Оператор вывода <PRINT>

Оператор предназначен для вывода результатов работы в окно отладочной информации в процессе тестовой генерации управляющей программы. Если управляющая программа формируется автономной исполняющей системой постпроцессора (<**InpD.dll**> или <**SprutPP.exe**>), то оператор игнорируется.

Формат:

```
PRINT <математическое выражение> | <строковое выражение>
   {, <математическое выражение> | <строковое выражение>} {, :}}
```

Описание:

Ключевым словом оператора вывода является слово <**PRINT**>. Вслед за ним следует список, состоящий из одного или более математических или строковых выражений. Число выражений в списке не ограничено; если их больше одного, то они разделяются запятыми.

В результате выполнения оператора последовательно будут вычисляться значения выражений и результаты расчетов будут выводиться на дисплей.

Примеры:

```
PRINT “Работает программа ABSMOVE”
PRINT CLD[1]-XT, “ приращение по X ”
PRINT AA, “ ”, CLD[1]
```

4.2.4 Оператор ввода <INPUT>

Оператор предназначен для ввода исходных данных с клавиатуры в процессе выполнения программы.

Формат:

```
INPUT <числовая переменная> | <строковая переменная> | <литерная строка>
   {, <числовая переменная> | <строковая переменная> | <литерная строка>
   {, :}}
```

Описание:

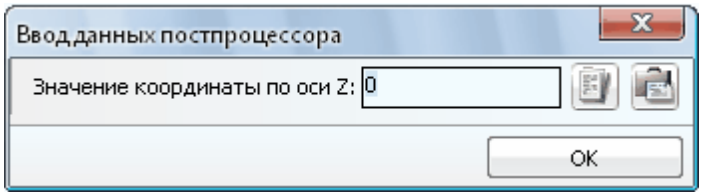
Ключевым словом оператора вывода является слово <**INPUT**>. Вслед за ним следует список, состоящий из одной или более числовых или строковых переменных и литерных строк. Число параметров в списке не ограничено; если их больше одного, то они разделяются запятыми.

В результате выполнения оператора будет сформировано окно ввода информации, состоящее из литерных строк и полей ввода. Если переменным перед вызовом оператора были присвоены какие-либо значения, то эти значения будут занесены в соответствующие поля окна в качестве значений по умолчанию. При вводе значения производится контроль на соответствие типа переменной вводимому значению. В числовую переменную могут вводиться только числовые значения, в строковую – любые символы. После ввода всех значений необходимо закрыть окно нажатием на кнопку <**Да**>. При этом переменным будут присвоены соответствующие им введенные значения.

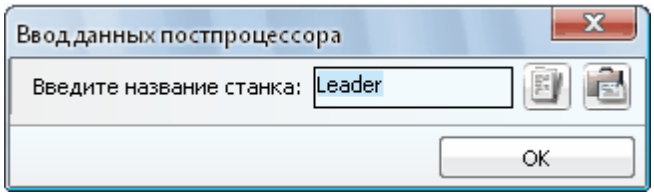
Результат действия оператора аналогичен оператору присваивания, но значения, заносимые в переменные, будут присваиваться не из программы, а при каждом выполнении программы запрашиваться для ввода. Использование этого оператора дает возможность, написав один раз любую программу, запускать ее и, вводя различные исходные данные, получать каждый раз новые результаты, зависящие от вводимых значений.

Примеры:

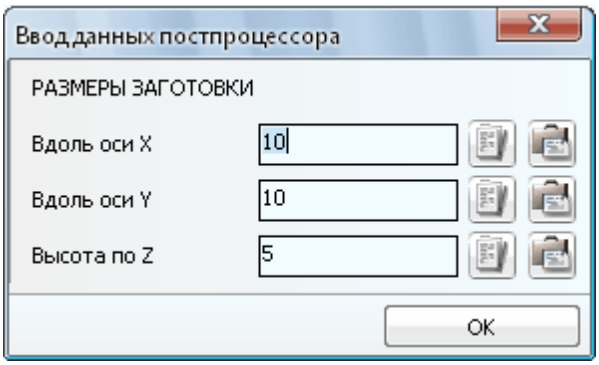
```
INPUT "Значение координаты по оси Z:", Z
```




```
MachineName$ = "Leader"
INPUT "Введите название станка: ", MachineName$
```




```
wx = 10; wy = 10; wz = 5
INPUT "РАЗМЕРЫ ЗАГОТОВКИ",
      "Вдоль оси X", wx,
      "Вдоль оси Y", wy,
      "Высота по Z", wz
```



Рядом с каждым текстовым полем в открывающемся окне расположены кнопки работы с буфером обмена Windows:

-  – копирует содержимое текстового поля в буфер обмена (аналогичное действие выполняется при нажатии сочетания клавиш [**Ctrl+C**]);
-  – вставляет в текстовое поле содержимое буфера обмена (аналогичное действие выполняется при нажатии сочетания клавиш [**Ctrl+V**]).

4.2.5 Оператор условного выполнения <IF>

Оператор предназначен для выполнения одного из входящих в него операторов, в зависимости от истинности какого-либо условия.

Формат:

```
IF <условное выражение> THEN <оператор 1>
{ELSE <оператор 2>}
```

Описание:

Этот оператор позволяет поставить дальнейшее выполнение программы в зависимость от некоторых условий.

Ключевым словом оператора является зарезервированное слово <**IF**>. Вслед за этим словом следует <**условное выражение**>, при истинности которого выполняется оператор, следующий за словом <**THEN**>. Если <**условное выражение**> ложно, то выполняется оператор, следующий за словом <**ELSE**>. <**ELSE**> – часть оператора, являющаяся необязательной. При ее отсутствии и ложности <**условного выражения**> действие оператора заканчивается и выполняется следующий оператор программы.

<**Условное выражение**> представляет собой одно <**Простое условное выражение**> или последовательность <**Простых условных выражений**>, связанных логическими операциями <**AND**> – логическое И и <**OR**> – логическое ИЛИ, то есть:

```
<простое условное выражение> {{AND <простое условное выражение>} |
```

```
{OR <простое условное выражение>}}
```

<**Условное выражение**> истинно, если оба <**простых условных выражений**>, связанные операцией <**AND**> истинны, или если хотя бы одно из условий, связанных операцией <**OR**>, истинно.

В свою очередь, <**простое условное выражение**> – это два математических или два строковых выражения, разделенные знаком операции сравнения, то есть:

```
<Математическое выражение> <Знак сравнения> <Математическое выражение>
```

или

```
<Строковое выражение> <Знак сравнения> <Строковое выражение>
```

Выражения, находящиеся по обе стороны <**знака сравнения**> должны быть обязательно одного типа, иначе при трансляции оператора вы получите сообщение об ошибке: "Несовпадение типов сравниваемых переменных".

<**Знаки сравнения**> обозначают одну из операций сравнения:

- **=** – значение левого выражения <**РАВНО**> значению правого;
- **>** – значение левого выражения <**БОЛЬШЕ**> значения правого;
- **<** – значение левого выражения <**МЕНЬШЕ**> значения правого;
- # или **<>** – значение левого выражения <**НЕ РАВНО**> значению правого;
- **>=** или **=>** – значение левого выражения <**БОЛЬШЕ ИЛИ РАВНО**> значению правого;
- **<=** или **=<** – значение левого выражения <**МЕНЬШЕ ИЛИ РАВНО**> значению правого.

Особо следует обратить внимание на сравнение строковых выражений. Сравнение двух строк выполняется посимвольно слева направо с учетом их кодов в ASCII. Для русских и латинских букв эта последовательность совпадает с алфавитной последовательностью, причем:

```
Прописные латинские буквы < строчные латинские <
прописные русские буквы < строчные русские
```

Если строки имеют различную длину, но все символы более короткой строки совпадают с соответствующими по расположению символами более длинной, то считается, что более короткая строка меньше длинной. Строки считаются равными только тогда, когда они содержат одинаковые символы и имеют одинаковую длину.

При обработке условного оператора сначала вычисляются значения математических и строковых выражений в <**простых условных выражениях**>, затем производятся операции сравнения, затем последовательно логические операции <**AND**> и <**OR**>. После выполнения всех этих действий, в зависимости от

истинности результата выполняется либо <**THEN**>-часть, либо <**ELSE**>-часть.

Примеры:

```
! Пример 1.
! Пример использования условного оператора.
IF kadr<224 THEN PRINT “Маленькая программа”
ELSE PRINT “Вы гений программирования”


! Пример 2.
! Пример использования вложенных условных операторов
IF POS(“ ”, строка$) = 2 THEN
  b$ = “Первой в строке идет буква”
ELSE
IF POS(“ ”, строка$) <= 4 THEN
  b$ = “Первым в строке идет слово длиной до 4 букв”
ELSE
  b$ = “Первым в строке идет слово длиной более 5 букв”
```

4.2.6 Оператор множественного условного выполнения <CASE>

Оператор позволяет провести анализ значения некоторого выражения и в зависимости от его значения выполнять те или иные действия.

Формат:

```
CASE <выражение> OF
  <список значений 1>: <оператор 1>
  ...
  <список значений N>: <оператор N>
  ELSE <оператор M>
END
```

Описание:

В этой конструкции <**выражение**> должно иметь числовой результат. Нельзя использовать выражения, возвращающие строки.

Списки значений могут содержать одно или несколько разделённых запятыми возможных значений константных выражений. После списка ставится двоеточие <**:**>, а затем пишется оператор, который должен выполняться, если выражение приняло одно из перечисленных в списке значений. После выполнения этого оператора работа структуры <**CASE**> завершается, и управление передаётся следующему за этой конструкцией оператору.

Если значение выражения не соответствует ни одному из перечисленных во всех списках, то выполняется оператор,

следующий после ключевого слова <**ELSE**>. Раздел <**ELSE**> не обязательно должен включаться в структуру <**CASE**>. В этом случае, если в списках не нашлось соответствующего значения выражения, то ни один оператор не будет выполнен.

Пример:

```
CASE i OF
  1, 2, 3, 4, 5: Str = "Меньше или равно 5"
  6, 7, 8, 9: Str = "Больше 5"
  ELSE Str = "Ошибочное значение"
END
```

4.2.7 Оператор перехода на метку <JUMP>

Оператор предназначен для выполнения операторов в порядке, отличном от порядка их написания.

Формат:

```
JUMP <номер метки>
...
<номер метки>:
```

Описание:

В операторе перехода на метку за ключевым словом <**JUMP**> указывается одна из меток, указанных в текущей программе. В отличие от объявления метки в операторе перехода на метку, символ <**:**> (двоеточие) после номера метки не указывается.

После выполнения оператора перехода выполняется оператор, помеченный соответствующей меткой. Для успешного выполнения оператора необходимо, чтобы соответствующая метка была в тексте программы. В противном случае, при трансляции программы Вы увидите сообщение об ошибке.

Использование оператора перехода на метку не может быть рекомендовано в качестве обычного средства написания программ.

Пример:

```
AFF = 20    ! оператор выполняется первым
JUMP 2      ! переход на метку номер 2
ABC = 1     ! оператор не выполняется
2: ATR = 4  ! оператор выполняется вторым
```

4.2.8 Оператор цикла <FOR>

Оператор предназначен для повторения выполнения оператора указанное число раз.

Формат:

```
FOR <числовая переменная цикла> = <математическое выражение>
    TO <математическое выражение>
    {STEP <число> | <числовая переменная>}
    DO <оператор>
```

Описание:

Оператор цикла предусматривает повторяющееся выполнение некоторого оператора с одновременным увеличением переменной цикла до тех пор, пока значение переменной цикла не превысит установленного предела.

Ключевым словом в операторе является слово <**FOR**>. Вслед за ним указывается конструкция, внешне очень близкая к оператору присваивания:

```
<числовая переменная цикла> = <математическое выражение>
```

И по действиям здесь много общего, т.к. при выполнении оператора цикла результат математического выражения присваивается указанной переменной. Кроме этого, в операторе цикла эта переменная объявляется переменной цикла, т.е. именно той переменной, по значению которой и будет контролироваться количество раз выполнения оператора.

За зарезервированным словом <**TO**> указывается математическое выражение, определяющее верхнюю границу, при достижении которой переменной цикла, выполнение оператора цикла прекращается и управление передается следующему оператору в программе. В качестве верхней границы может быть указано значение любого математического выражения.

Необязательная часть оператора цикла, состоящая из зарезервированного слова <**STEP**> и следующего за ним числа или числовой переменной, указывает на шаг изменения переменной цикла. При каждом последующем выполнении цикла, переменная цикла увеличивается на величину шага цикла. Если шаг равен 1, то эта часть оператора может быть опущена. Необходимо помнить, что шагом цикла не может быть выражение.

Затем, вслед за словом <**DO**>, указывается оператор, который и должен выполняться циклически определенное число раз. Таким образом, все предыдущие параметры оператора цикла служат лишь для организации циклического выполнения этого оператора.

Примеры:

```
! Пример 1.
! Простой пример использования оператора цикла.
FOR as = 3 TO 10 DO PRINT as:0," ",as^2:0
```

```
! Пример 2.
! Использование вложенных операторов цикла
FOR i = 0 TO 0.9 STEP 0.5 DO
  FOR j = -1 TO 0 STEP 0.2 DO
    PRINT " i = ",i:1," j = ",j:1
```

4.2.9 Оператор цикла <REPEAT>

Структура <**REPEAT...UNTIL**> используется для организации циклического выполнения совокупности операторов, называемой телом цикла, до тех пор, пока не выполнится некоторое условие.

Формат:

```
REPEAT
  <операторы тела цикла>
UNTIL <условное выражение>
```

Описание:

Структура работает следующим образом. Выполняются операторы тела цикла, затем вычисляется <**условное выражение**>. Если выражение условия возвращает ложь, то повторяется выполнение операторов тела цикла и после этого снова вычисляется условие. Такое циклическое повторение цикла продолжается до тех пор, пока проверяемое условие не вернет истину. После этого цикл завершается, и управление передается оператору, следующему за структурой <**REPEAT...UNTIL**>.

Поскольку проверка условия осуществляется после выполнения операторов тела цикла, то эти операторы заведомо будут выполнены хотя бы один раз, даже если условие сразу истинно. С другой стороны, условие рано или поздно должно вернуть истину. Если этого не произойдет, то программа "зациклится", т.е. цикл будет выполняться бесконечно.

4.2.10 Оператор цикла <WHILE>

Структура <**WHILE...DO**> используется для организации циклического выполнения оператора, называемого телом цикла, пока выполняется некоторое условие.

Формат:

```
WHILE <условное выражение> DO <оператор>
```

Структура работает следующим образом. Сначала вычисляется <**условное выражение**>, которое должно возвращать результат

булева типа. Если условие возвращает истину, то выполняется оператор тела цикла, после чего опять вычисляется выражение, определяющее условие. Такое циклическое повторение выполнения оператора и проверки условия продолжается до тех пор, пока условие не вернет ложь. После этого цикл завершается, и управление передается оператору, следующему за структурой <**WHILE...DO**>.

Поскольку проверка выражения осуществляется перед выполнением оператора тела цикла, то, если условие сразу ложно, оператор не будет выполнен ни одного раза. В этом основное отличие структуры <**WHILE...DO**> от структуры <**REPEAT...UNTIL**>, в котором тело цикла заведомо выполняется хотя бы один раз. Условие рано или поздно должно вернуть ложь. Если этого не произойдет, то программа "зациклится", т.е. цикл будет выполняться вечно.

4.2.11 Составной оператор <BEGIN...END>

Предназначен для объединения группы операторов в один оператор. Выполняет роль операторных скобок.

Формат:

```
BEGIN
  <оператор 1>
  ...
  <оператор N>
END
```

Описание:

Составной оператор предусматривает выполнение входящих в него операторов в порядке их написания. Служебные слова <**BEGIN**> и <**END**> выступают как операторные скобки. Глубина вложенности составных операторов не ограничена. Составной оператор может использоваться везде, где допустимо использование одиночного оператора.

Пример:

```
! Простой пример составного оператора
IF var < 3 THEN BEGIN
  ac=12; bb=16
END ELSE BEGIN
  ac=15; bb=60
END ! Слово END закрывает оператор
```

4.2.12 Оператор вызова программы <CALL>

Оператор предназначен для вызова подпрограммы из программы обработки технологических команд или другой подпрограммы.

Формат:

```
CALL <имя программы> {(<список фактических параметров>)}
```

Описание:

Словом, определяющим оператор вызова подпрограммы, является слово <**CALL**>, вслед за которым указывается <**имя подпрограммы**> – строковое выражение или литерная строка, не ограниченная двойными кавычками, значение которой соответствует имени вызываемой подпрограммы.

<**Список фактических параметров**> представляет собой последовательность числовых либо строковых переменных, констант, выражений или массивов. Если число параметров в списке больше одного, то они разделяются запятыми. Тип передаваемых внутрь подпрограммы фактических параметров должен соответствовать типу формальных параметров, которые объявлены в реализации подпрограммы. Если подпрограмма не содержит параметров, то скобок после имени подпрограммы не должны быть.

Оператор <**CALL**> передает управление указанной подпрограмме, после ее отработки делает возврат в программу, из которой осуществлялся вызов.

Пример:

```
X = 50; C = 0
CALL ConvertXYtoXC(X, 30, C)

CALL FillCycleParameters
```

4.2.13 Оператор начала подпрограммы <SUB>

Предназначен для объявления подпрограммы и списка параметров, передаваемых ей при запуске.

Формат:

```
SUB <имя подпрограммы> {(<список формальных параметров>)}
```

Описание:

Ключевое слово оператора – <**SUB**>, за которым следует обязательный параметр имя программы – литерная строка, не ограниченная двойными кавычками и далее необязательный <

список формальных параметров>, заключенный в круглые скобки.

<**Список формальных параметров**> представляет собой последовательность числовых, строковых переменных или массивов состоящую из одного и более элементов. Если число параметров в списке больше одного, то они разделяются запятыми.

Переменные, указываемые в списке формальных параметров, при вызове подпрограммы будут содержать значения, указанные в операторе вызова. Поэтому эти переменные уже определены в программе и могут быть использованы в любых операторах.

Пример:

```
sub GetProgramID(PrgID: Integer)
! Извлекаем из строкового имени NC-программы числовой идентификатор
  i: Integer
  j: Integer
  k: Integer
  s$: String

  i = 1
  j = 0
  k = 0
  while i<=Len(NCName$) do begin
    s$ = Copy(NCName$, i, 1)
    case Ord(s$) of
      48, 49, 50, 51, 52, 53, 54, 55, 56, 57: begin
        if j<1 then j = i
        if (k<1) or (k=(i-1)) then
          k = i
      end
    end
    i = i + 1
  end
  if (j>0) and (k>0) then begin
    s$ = Copy(NCName$, j, k-j+1)
    PrgID = Num(s$)
  end
subend
```

4.2.14 Оператор окончания подпрограммы <SUBEND>

Оператор предназначен для указания окончания исходного текста подпрограммы.

Формат:

```
SUBEND
```

Описание:

Оператор определяет конец подпрограммы. В результате выполнения оператора выполнение подпрограммы заканчивается, значения всех переменных, указанных в списке формальных параметров оператора объявления подпрограммы присваиваются переменным, указанным в операторе вызова подпрограммы и продолжается выполнение вызвавшей ее программы с оператора, следующего за оператором вызова.

4.2.15 Оператор начала процедуры <PROC>

Оператор предназначен для объявления начала и списка передаваемых параметров внутренней процедуры в программе обработки технологической команды или подпрограммы.

Формат:

```
PROC <имя процедуры> {(<список формальных параметров>)}
```

Описание:

За ключевым словом <**PROC**> указывается <**имя процедуры**> – идентификатор, не совпадающий с зарезервированными словами системы и именами ранее объявленных переменных. Это имя служит для идентификации процедуры при ее вызове, поэтому имена всех процедур в одной программе должны быть уникальны.

После указания имени процедуры, следует необязательный <**список формальных параметров**>, который представляет собой последовательность числовых, строковых переменных и массивов, состоящую из одного и более элементов. Если число параметров в списке больше одного, то они разделяются запятыми. В отличие от оператора объявления подпрограммы переменные, указываемые в <**списке формальных параметров**> процедуры не инициализируются, т.е. перед указанием этих переменных в списке параметров необходимо занести в них какие-либо значения.

Переменные, указываемые в <**списке формальных параметров** >, при вызове процедуры будут содержать значения, указанные в операторе вызова. Поэтому эти переменные уже определены в программе и могут быть использованы в любых операторах.

Оператор определения процедуры по формату и действию очень похож на оператор объявления подпрограммы. Самое существенное отличие состоит в том, что процедура имеет доступ ко всем переменным и массивам той программы или подпрограммы, в теле которой она описана.

Располагать объявленную процедуру можно в любом месте программы или подпрограммы.

4.2.16 Оператор возврата из процедуры <RETURN>

Оператор предназначен для указания окончания процедуры.

Формат:

```
RETURN
```

Описание:

Этот оператор является последним оператором процедуры. При выполнении оператора <**RETURN**> происходит обратное присвоение значений формальных параметров процедуры переменным, указанным при ее вызове, и возврат к выполнению программы с оператора, следующего за оператором вызова этой процедуры.

Пример:

```
! Пример использования процедуры
program TestProc
  NErr: Integer
  S: String

  proc PrintDebugInfo(NErr, S)
    case NErr of
      1: Print "Ошибка интерполяции: ", S
      2: Print "Ошибка аппроксимации: ", S
      else Print "Неизвестная ошибка: ", S
    end
  return

  PrintDebugInfo(CLD[1], "Параметр 1")
  PrintDebugInfo(CLD[2], "Параметр 2")
  PrintDebugInfo(CLD[3], "Параметр 3")
end
```

4.2.17 Оператор вывода кадра <OUTBLOCK>

Оператор формирует содержимое кадра в соответствии с заданным форматом, определенным порядком и форматом регистров, в переменной <**OutStr$**> и выводит в файл управляющей программы ЧПУ.

Формат:

```
OUTBLOCK {<FileName$>}
```

Описание:

Кадр управляющей программы формируется системой следующим

образом: перебираются регистры по порядку, начиная с первого; если старое и текущее значения регистра различаются, то регистр выводится в кадр, а его старое значение приравнивается текущему; а если старое и текущее значения регистра равны, то регистр в кадр не выводится.

При выводе регистра в кадр управляющей программы сначала записывается его идентификатор, а затем умноженное на масштаб текущее значение регистра. Число выводится в кадр в соответствии с описанным в регистре форматом (длина и точность значения регистра, наличие десятичной точки, знака, лидирующих и незначащих нулей).

Кадр формируется в переменной <**OutStr$**>, и, по окончании формирования, выводится в управляющую программу отдельной строкой.

Используйте необязательный параметр <**FileName$**>, чтобы указать файл в который будет осуществляться вывод кадра программы. Если параметр опущен, то кадр выводится в текущий файл, если указано значение "*", то кадр будет выводиться в главный файл, причем произойдет переключение текущего файла. Если <**FileName$**> – имя файла, то будет создана соответствующая закладка, и текущий файл вывода будет переключен на указанный.

4.2.18 Оператор формирования кадра <FORMBLOCK>

Оператор формирует содержимое кадра в соответствии с заданным форматом, определенным порядком и форматом регистров, в переменной <**OutStr$**> без вывода в файл управляющей программы.

Формат:

```
FORMBLOCK
```

Описание:

Кадр управляющей программы формируется точно так же, как и в операторе вывода кадра. Но, в отличие от него переменная <**OutStr$**> в кадр управляющей программы не выводится.

Далее с переменной <**OutStr$**> можно работать как с обычной строковой переменной, а затем вывести в кадр оператором <**OUTPUT**>.

4.2.19 Оператор прямого вывода в кадр <OUTPUT>

Оператор выводит содержимое строковой переменной отдельным кадром управляющей программы.

Формат:

```
OUTPUT {<FileName$>,} S$
```

Описание:

<**S$**> – идентификатор строковой переменной. Параметром оператора может служить также литерная константа или строковое выражение.

Используйте необязательный параметр <**FileName$**>, чтобы указать файл, в который будет осуществляться вывод кадра программы. Если параметр опущен, то кадр выводится в текущий файл, если указано значение "*", то кадр будет выводиться в главный файл, причем произойдет переключение текущего файла. Если <**FileName$**> – имя файла, то будет создана соответствующая закладка, и текущий файл вывода будет переключен на указанный.

Примеры:

```
! Вывод первого символа в УП
OutPut “%”
Name$ = “ Иван”
F$ = “Иванов”
! Вывод строки в УП
OutPut “Программист ”+Name$+” “+F$
! В кадр УП выведется строка 'Программист Иван Иванов'

! Пример использования операторов
! FormBlock и Output вместо OutBlock
FormBlock
Output OutStr$
! Действие двух операторов аналогично действию OutBlock
```

4.2.20 Оператор замены подстроки в строке <REPLACE>

Оператор ищет подстроку в строке и при нахождении заменяет её на требуемую.

Формат:

```
REPLACE (<строковая переменная>, <искомая подстрока>, <строка для замены>)
```

Описание:

Этот оператор ищет в <**строковой переменной**>, начиная с

начала строки, <**искомую подстроку**>. При нахождении <**искомая подстрока**> заменяется на <**строку для замены**> и поиск прекращается. При не нахождении <**искомой подстроки**> замены не происходит.

Пример:

```
SUB Filter(S: String)
  REPLACE(S, "G1", "L")    ! Поиск и замена "G1" на "L"
  REPLACE(S, "G2", "DR-")  ! Поиск и замена "G2" на "DR-"
  REPLACE(S, "G3", "DR+")  ! Поиск и замена "G3" на "DR+"
SUBEND
```

4.2.21 Оператор формирования кадра по шаблону <MASK>

Оператор формирует кадр управляющей программы по строке шаблона, в соответствии с правилами, и выводит в файл управляющей программы ЧПУ.

Формат:

```
MASK (<строка шаблона>) {, OutBlock {, Poll}}
```

Описание:

Оператор формирует кадр управляющей программы, в соответствии с правилами, описанными в строке шаблона. Опция <**OutBlock**> соответствует переключателю <**Вывод текущего кадра**>, а опция <**Poll**> соответственно переключателю <**Опросить все регистры**>.

Пример:

```
! Пример формирования кадра УП по шаблону
MASK(M[30]), OutBlock
MASK(% N[Off] ), OutBlock
MASK(G_INTERP[INTERP]X[CLD.X]Y[CLD.Y]Z[CLD.Z]), OutBlock, Poll
```

4.2.22 Оператор смены текущего файла УП

Оператор меняет текущий файл УП. Последующие операторы вывода текста УП в файл в которых не указан новый файл будут выводить кадры в указанный файл.

Формат:

```
ChangeNCFile NewNCFileName$
```

Описание:

<**NewNCFileName$**> – любое выражение, результатом вычисления которого является строка.

Пример:

```
! Пример изменения файла вывода
ChangeNCFile "File1.nc" ! сменили текущий файл вывода
Output "кадр УП" ! строка будет выведена в файл "File1.nc"
```

4.2.23 Оператор вызова внешней задачи

Используйте оператор <**ShellExec**> для вызова команды среды или внешней задачи.

Формат:

```
ShellExec CommandLine$
```

Описание:

<**CommandLine$**> – любое выражение, результатом вычисления которого является строка.

При выполнении данного оператора постпроцессор запустит интерпретатор команд операционной системы и передаст ему на исполнение команду <**CommandLine$**> (эквивалентно вызову "cmd /c CommandLine$").

Пример:

```
! Вызов команды системы
ShellExec "erase temp\*.*" ! очистим содержимое папки temp
```

4.2.24 Оператор обрашения к пользовательским данным SprutCam

Используйте оператор <**CustomData**> для обращения к пользовательским данным проекта SprutCam.

Формат:

```
S$ = CustomData(ItemName$)
```

Описание:

<**ItemName$**> – любое выражение, результатом вычисления которого является строка.

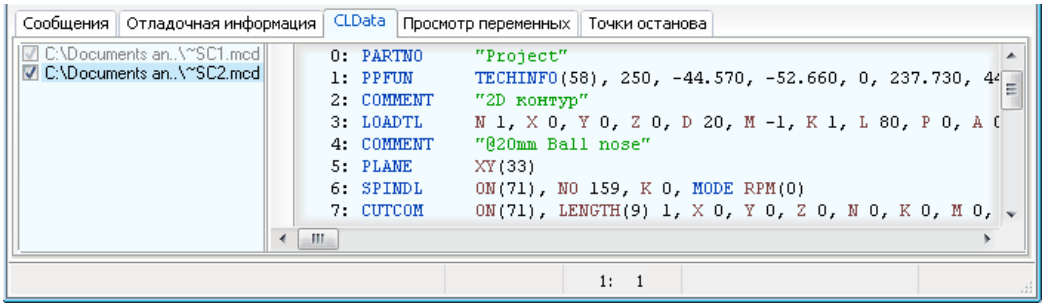
Оператор **CustomData** возвращает значение установленное пользователем для элемента с именем **ItemName$** при создании проекта.

Пример:

```
! Вывод пользовательских данных в УП:
Output CustomData("UserData")
```

4.2.25 Операторы работы с NC-подпрограммами

NC-подпрограмма представляет собой последовательность технологических команд, оформленную в виде отдельного файла технологических команд ***.mcd** (эти файлы формируются автоматически), обрамленную специальными технологическими командами начала NC-подпрограммы <**PPFUN STARTSUB(50)**> и конца NC-подпрограммы <**PPFUN ENDSUB(51)**>.



Как показано на рисунке в списке файлов технологических команд файлы с NC-подпрограммами помечаются серым цветом, и изменение состояния галочки рядом с именем файла становится невозможным.

Технологические команды внутри файлов подпрограмм в отличие от обычных файлов технологических команд не начинают обрабатываться автоматически. Технологические команды в NC-подпрограммах будут обрабатываться только при вызове специальных операторов, начинающихся с ключевого слова <**NCSUB**>. Например, в процессе обработки технологической команды <**PPFUN CALLSUB(52)**>.

Каждая NC-подпрограмма идентифицируется уникальным номером, который указывается в параметрах технологических команд <**PPFUN STARTSUB(50)**>, <**PPFUN ENDSUB(51)**> и <**PPFUN CALLSUB(52)**> под номером 2. Номер доступен в процедурах обработки через предопределенный массив <**CLD**>. Таким образом, внутри процедуры обработки выражение <**CLD[2]**> вернет номер соответствующей NC-подпрограммы. Также NC-подпрограмма может характеризоваться строковым именем. Начальное значение строкового имени указывается в технологической команде <**COMMENT**>, находящейся перед технологической командой <**PPFUN STARTSUB(50)**>. В процедурах обработки технологических команд строковое имя NC-подпрограммы может быть прочитано и модифицировано оператором <**NCSUB.NAME(n)**>, где <**n**> – уникальный номер NC-

подпрограммы.

В процессе трансляции технологических команд в управляющую программу может потребоваться идентифицировать строки начала и конца конкретной NC-подпрограммы (определить метки или номера кадров). Для этих целей предназначены операторы определения начальной и конечной меток подпрограммы <NCSUB.STARTLABEL(n)> и <NCSUB.ENDLABEL(n)>, где <**n**> – уникальный номер NC-подпрограммы.

Оператор вывода NC-подпрограммы <NCSUB.OUTPUT>

Оператор передает управление в файл технологических команд ***.mcd**, соответствующий определенной NC-подпрограмме.

Формат:

```
NCSUB.OUTPUT {(<Номер NC-подпрограммы>{, <Режим>}{, <Имя файла>})}
```

Оператор является функцией, т.е. возвращает результат. Возможные значения возвращаемого результата: 0 или 1.

Описание:

В результате выполнения оператора последовательно начинают вызываться программы обработки технологических команд NC-подпрограммы, указанной идентификатором <**Номер NC-подпрограммы**>. Номер NC-подпрограммы обычно передается через предопределенный массив параметров технологических команд <CLD> с индексом 2. Номер NC-подпрограммы является необязательным параметром. В случае отсутствия номера будет произведена последовательная трансляция всех имеющихся NC-подпрограмм.

Необязательный параметр <**Режим**> определяет, в каких случаях результат трансляции NC-подпрограммы должен выводиться в текст управляющей программы, и может принимать три значения: 0, 1 и 2. При неуказанном параметре <**Режим**> считается, что он равен 1.

В режиме 0 выполняется трансляция технологических команд, в результате чего производится изменение значений регистров и глобальных переменных. Однако результат трансляции не попадает в текст управляющей программы. Таким образом, осуществляется своеобразный "холостой прогон", позволяющий имитировать вызов программы в том случае, когда сам текст подпрограммы должен появиться в другом месте управляющей программы.

В режиме 1 (по умолчанию) трансляция технологических команд NC-подпрограммы производится также как и в режиме 0, однако результат трансляции попадает в текст управляющей программы, если NC-подпрограмма еще ни разу не выводилась. При последующих вызовах оператора в этом режиме осуществляется "холостой прогон", как и в режиме 0. Таким образом, результат

трансляции NC-подпрограммы попадает в управляющую программу только один раз.

При вызове оператора в режиме 2 результат трансляции NC-подпрограммы выводится в управляющую программу всегда, т.е. столько раз, сколько вызывается оператор. Такой режим позволяет "развернуть" NC-подпрограммы для систем ЧПУ не поддерживающих выполнение подпрограмм.

Возвращаемый результат показывает, производился ли вывод NC-подпрограммы в управляющую программу (УП). Результат равен 0, если вывода не было (был осуществлен "холостой прогон"). Результат равен 1, если от транслированная NC-подпрограмма выводилась в УП. Соответственно в режиме 0 возвращаемый результат всегда равен 0, в режиме 2 – всегда равен 1. В режиме 1 результат равен 1 только при первом вызове оператора для каждой NC-подпрограммы.

Необязательный строковый параметр <**Имя файла**> определяет имя файла, в который будет сохранен текст NC-подпрограмм, получаемых в результате трансляции. Если параметр не указан, то подпрограмма выводится в файл основной управляющей программы. Если в указанном имени отсутствует путь, то файл будет создан в той же папке, что и файл основной программы.

Пример.

Пусть, основной файл технологических команд содержит следующие команды:

```
0: PARTNO "Lathe Roughing"
1: PPFUN TECHINFO(58), 250.000, 114.890, 78.750, 0.000, 140.690,
2: COMMENT "Lathe Roughing"
3: LOADTL N 0, X 0.000, Y 0.000, Z 0.000, D 0.000, M 0, K 1, L 0.000
4: SPINDL ON(71), NO 1000.000, K 0, MODE CSS(2), SPEED 600.000
5: FEDRAT M 0.050, K 3, MMPR(316)
6: GOTO.abs X 140.690, Y 104.550, Z 0.000
7: GOTO.abs X 140.690, Y 78.750, Z 0.000
8: GOTO.abs X 114.890, Y 78.750, Z 0.000
9: FEDRAT M 0.050, K 0, MMPR(316)
10: PPFUN CALLSUB(52), 1.000, 0.000, 0.000, 0.000, 0.000, 0.000
11: FEDRAT M 0.050, K 8, MMPR(316)
12: GOTO.abs X 114.890, Y 104.550, Z 0.000
13: GOTO.abs X 140.690, Y 104.550, Z 0.000
14: SPINDL OFF(72), NO 0.000, K 0
15: FINI
```

В то время как файл NC-подпрограммы номер 1 имеет следующий список технологических команд:

```
1: COMMENT "Geometry_1"
2: PPFUN STARTSUB(50), 1.000, 0.000, 0.000, 0.000, 0.000, 0.000
3: FEDRAT M 0.000, K 0, MMPR(316)
4: GOTO.abs X 114.890, Y 33.381, Z 0.000
5: CIRCLE XC 114.690, YC 33.380, ZC 0.000, R 0.200, XE 114.831, YE
6: GOTO.abs X 110.151, Y 38.201, Z 0.000
7: CIRCLE XC 110.010, YC 38.060, ZC 0.000, R 0.200, XE 110.010, YE
```

```
8: GOTO.abs X 62.230, Y 38.260, Z 0.000
9: GOTO.abs X 41.460, Y 38.257, Z 0.000
10: PPFUN ENDSUB(51), 1.000, 0.000, 0.000, 0.000, 0.000, 0.000
```

Пусть программа обработки десятой команды (<**PPFUN CALLSUB (52)**>) основного файла выглядит так:

```
program PPFun
  if cld[1]=52 then begin ! CallSub
    NCSub.Output(CLD[2])
  end
end
```

В таком случае с самого начала будет производиться последовательная обработка технологических команд основного файла со строки 0 до строки 9. Достигнув строки 10, при обработке оператора <**NCSub.Output(CLD[2])**>, управление перейдет на файл с NC-подпрограммой номер 1, т.к. второй параметр технологической команды равен 1.000. Таким образом, начнут последовательно обрабатываться технологические команды NC-подпрограммы со строки 1 до строки 10. После обработки последней команды (<**PPFUN ENDSUB(51)**>) управление вернется в основной файл, и будут обработаны все оставшиеся команды (строки 11 – 15).

Примеры вызова оператора приведены ниже:

```
! Осуществляет трансляцию всех имеющихся NC-подпрограмм
NCSub.Output

! Осуществляет трансляцию NC-подпрограммы с номером 3 в режиме 0
NCSub.Output(3, 0)

! Осуществляет трансляцию NC-подпрограммы с номером,
! извлекаемым из второго параметра команды в режиме 2
NCSub.Output(CLD[2], 2)

if NCSub.Output(CLD[2]) = 0 then ! Если уже был вывод в УП
begin                            ! Осуществляет трансляцию
  Output "CALL " +               ! NC-подпрограммы в режиме 1
  NCSub.StartLabel(CLD[2])       ! Выводит строку "CALL "
end

! Транслирует NC-подпрограмму с номером 7 в режиме 1
! и выводит результат в файл Sub7.txt
NCSub.Output(7, "Sub7.txt")

! Транслирует NC-подпрограмму с номером 7 в режиме 2
! и выводит результат в файл Sub7.txt
NCSub.Output(7, 2, "Sub7.txt")
```

Оператор вывода всех подпрограмм <NCSUB.OUTPUTALL>

Оператор <**NCSUB.OUTPUTALL**>, как и оператор <**NCSUB.OUTPUT**> при вызове без параметров, осуществляет трансляцию всех загруженных NC-подпрограмм. Отличие в том, что <**NCSUB.OUTPUTALL**> позволяет вывести все подпрограммы в отдельные файлы.

Синтаксис:

```
NCSUB.OUTPUTALL(<Режим>)
```

<**Режим**> – режим выполнения:

- 0 – работает также, как и <**NCSUB.OUTPUT**> без параметров;
- 1 – сохранит тексты всех подпрограмм в файлах с именами соответствующими <**NCSUB.NAME**>.

Файлы будут иметь расширение указанное в настройках постпроцессора, и будут созданы в той же папке, что и файл главной программы

Оператор определения имени NC-подпрограммы <NCSUB.NAME>

Возвращает строковое имя NC-подпрограммы, а также позволяет присвоить строковому имени NC-подпрограммы произвольное текстовое значение.

Формат:

```
S$ = NCSUB.NAME(<Номер NC-подпрограммы>)
```

или

```
NCSUB.NAME(<Номер NC-подпрограммы>) = S$
```

Описание:

- <**Номер NC-подпрограммы**> – уникальный идентификатор NC-подпрограммы.
- <**S$**> – произвольная строковая переменная. Во втором случае также может задаваться строковая константа или произвольное выражение, возвращающее строку.

Оператор может использоваться аналогично строковой переменной в выражениях и функциях.

Для вывода в УП текстового идентификатора подпрограммы в нужном формате следует использовать оператор определения имени NC-подпрограммы.

***Примечание**: Перед первым присвоением имени NC-подпрограммы оператором*

*<**NCSUB.NAME**>, имя уже имеет значение по умолчанию, которое извлекается из строки комментария (технологической команды <**COMMENT**>), расположенной перед технологической командой начала NC-подпрограммы <**PPFUN STARTSUB(50)**> соответствующей NC-подпрограммы.*

Пример:

```
St$: String
n1: Integer
St$ = "Example"
n1 = 3

! Присвоение имени значения "Example3":
NCSub.Name(n1)=St$+Str(n1)

! Вывод в управляющую программу строки "oExample3":
Output "o"+NCSub.Name(n1)
```

Оператор определения начальной метки NC-подпрограммы <NCSUB.STARTLABEL>

Возвращает строковое значение начальной метки NC-подпрограммы, а также позволяет присвоить строковому значению начальной метки NC-подпрограммы произвольное текстовое значение.

Формат:

```
S$ = NCSUB.STARTLABEL(<Номер NC-подпрограммы>)
```

или

```
NCSUB.STARTLABEL(<Номер NC-подпрограммы>) = S$
```

Описание:

- <**Номер NC-подпрограммы**> – уникальный идентификатор NC-подпрограммы.
- <**S$**> – произвольная строковая переменная. Во втором случае также может задаваться строковая константа или произвольное выражение, возвращающее строку.

Оператор может использоваться аналогично строковой переменной в выражениях и функциях обрабатывающих строки.

Оператор можно использовать для определения начальной метки NC-подпрограммы для вывода в УП строк, идентифицирующих начало подпрограммы. В качестве метки можно задать и номер кадра.

Обычно на момент начала трансляции технологических команд в УП место расположения конкретной подпрограммы в тексте УП неизвестно. В этот момент начальная метка NC-подпрограммы установлена в значение по умолчанию <**SLabelNxxx**>, где <**xxx**>

– номер NC-подпрограммы. В процессе реализации NC-подпрограммы, т.е. вывода ее в УП, становится возможным определить начальную метку. Поэтому, в процедуре обработки команды начала подпрограммы <**PPFUN STARTSUB(50)**> следует использовать оператор установки начальной метки подпрограммы:

```
NCSub.StartLabel(CLD[2]) = S$ ! Присвоение начальной метки
```

- <**S$**> – строковая переменная или произвольное выражение, возвращающее строку, соответствующую метке или номеру кадра подпрограммы.
- <**CLD[2]**> – номер NC-подпрограммы переданный через предопределенный массив параметров технологической команды <**CLD**>.

***Примечание**: При включенной автоматической нумерации кадров управляющей программы, перед началом выполнения процедуры обработки команды начала подпрограммы <**PPFUN STARTSUB(50)**> начальной метке автоматически присваивается текущее значение регистра номера кадра. Если это значение устраивает, то можно не использовать оператор для присвоения значения начальной метки.*

*При вызове подпрограммы обычно требуется определить, где расположена данная подпрограмма. Поэтому в процедуре обработки команды вызова подпрограммы <**PPFUN CALLSUB(52)**> следует использовать оператор определения начальной метки подпрограммы.*

```
S$ = NCSub.StartLabel(CLD[2]) ! Получение начальной метки
```

Затем следует вывести данную метку в текст управляющей программы в нужном формате.

***Примечание**: Если выполнение процедуры обработки команды вызова подпрограммы <**PPFUN CALLSUB(52)**> производится раньше, чем произошло присвоение значения начальной метки (до того как подпрограмма выведена в УП), то оператор определения начальной метки вернет значение по умолчанию вида <**SLabelNxxx**>". После окончания трансляции всей управляющей программы, если подпрограмма была выведена в УП, и метке было присвоено новое значение, отличающееся от значения по умолчанию, производится замена всех значений по умолчанию в тексте УП на вновь присвоенную метку.*

*В этом случае преобразования, осуществляемые над строкой, содержащей значение метки (значение по умолчанию вида <**SLabelNxxx**>), не должны нарушать целостность выражения по умолчанию, иначе результат трансляции УП может оказаться неверным. В таких случаях преобразования над строкой метки следует производить не при получении значения метки, а перед присвоением значения метки, в процедуре обработки команды начала подпрограммы <**PPFUN STARTSUB(50)**>.*

Оператор определения конечной метки NC-подпрограммы <NCSUB.ENDLABEL>

Возвращает строковое значение конечной метки NC-подпрограммы, а также позволяет присвоить конечной метки NC-подпрограммы произвольную строку.

Формат:

```
S$ = NCSUB.ENDLABEL(<Номер NC-подпрограммы>)
```

или

```
NCSUB.ENDLABEL(<Номер NC-подпрограммы>) = S$
```

Описание:

- <**Номер NC-подпрограммы**> – уникальный идентификатор NC-подпрограммы.
- <**S$**> – произвольная строковая переменная. Во втором случае также может задаваться строковая константа или произвольное выражение, возвращающее строку.

Оператор может использоваться аналогично строковой переменной в выражениях и функциях обрабатывающих строки.

Оператор следует использовать для определения конечной метки NC-подпрограммы при выводе в управляющую программу строк, идентифицирующих конец подпрограммы. В качестве метки можно задать и номер кадра.

Обычно на начало трансляции технологических команд в УП место расположения конкретной подпрограммы в УП неизвестно. В это время конечная метка NC-подпрограммы установлена в значение по умолчанию <**ELabelNxxx**>, где <**xxx**> – номер NC-подпрограммы. В процессе реализации NC-подпрограммы, т.е. вывода ее в УП, становится возможным определить конечную метку. Поэтому в процедуре обработки команды конца подпрограммы <**PPFUN ENDSUB(51)**> следует использовать оператор определения конечной метки для установки конечной метки подпрограммы:

```
NCSub.EndLabel(CLD[2]) = S$ ! Присвоение конечной метки
```

Здесь:

- <**S$**> – строковая переменная или произвольное выражение, возвращающее строку, соответствующую метке или номеру кадра подпрограммы.
- <**CLD[2]**> – номер NC-подпрограммы переданный через предопределенный массив параметров технологической команды <**CLD**>.

***Примечание**: При включенной автоматической нумерации кадров УП перед*

*началом выполнения процедуры обработки команды конца подпрограммы <**PPFUN ENDSUB(51)**> конечной метке автоматически присваивается текущее значение регистра номера кадра. В таких случаях можно не использовать оператор для задания конечной метки.*

*При вызове подпрограммы обычно требуется определить, где расположена данная подпрограмма. Поэтому в процедуре обработки команды вызова подпрограммы <**PPFUN CALLSUB(52)**> следует использовать оператор определения конечной метки подпрограммы:*

```
S$ = NCSub.EndLabel(CLD[2]) ! Получение конечной метки
```

Затем следует вывести данную метку в текст управляющей программы в нужном формате.

***Примечание**: Если выполнение процедуры обработки технологической команды вызова подпрограммы <**PPFUN CALLSUB(52)**> производится раньше, чем произошло присвоение значения конечной метки (до того как подпрограмма выведена в УП), то оператор определения конечной метки вернет значение по умолчанию вида <**ELabelNxxx**>. После окончания трансляции всей УП, если подпрограмма была выведена в УП, и метке было присвоено значение отличающееся от значения по умолчанию, производится замена всех значений по умолчанию в тексте УП на вновь присвоенную метку.*

*В этом случае преобразования, осуществляемые над строкой, содержащей значение метки (значение по умолчанию вида <**ELabelNxxx**>), не должны нарушать целостность выражения по умолчанию, иначе результат трансляции УП может оказаться неверным. В таких случаях преобразования над строкой метки следует производить не при получении значения метки, а перед присвоением значения метки в процедуре обработки технологической команды конца подпрограммы <**PPFUN ENDSUB (51)**>.*

Методика работы с NC-подпрограммами

В существующих системах ЧПУ приняты различные способы оформления подпрограмм в управляющей программе (УП), однако по критерию расположения текста подпрограмм относительно строк основной программы можно выделить несколько типичных случаев:

- Подпрограммы выводятся в УП до вывода основной программы;
- Подпрограммы выводятся в УП после вывода основной программы;
- Подпрограммы выводятся в УП в месте первого вызова;
- Подпрограммы не поддерживаются, и текст подпрограммы

выводится при каждом вызове.

Рассмотрим, каким образом можно реализовать каждый из указанных случаев.

Подпрограммы выводятся в УП до вывода основной программы

Данный формат вывода предполагает, что в управляющую программу в самом начале выводятся все имеющиеся подпрограммы, затем вставляется основная программа, содержащая строки вызова подпрограмм.

Для вывода подпрограмм в самом начале, следует вставить соответствующий код в процедуру обработки технологической команды <**PARTNO**>, которая выполняется первой. Вид процедуры показан ниже.

```
PROGRAM PARTNO
  ! Инициализация переменных
  ! Вывод строк, идентифицирующих начало секции подпрограмм
  NCSub.Output ! Вывод всех имеющихся подпрограмм
  ! Вывод строк, идентифицирующих конец секции подпрограмм и
  ! начало секции основной программы
  ! Инициализация переменных
END
```

С целью формирования строк управляющей программы, идентифицирующих начало, конец и вызов NC-подпрограммы, следует написать процедуру – обработчик технологических команд <**PPFUN STARTSUB(50)**>, <**PPFUN ENDSUB(51)**> и <**PPFUN CALLSUB(52)**> как показано ниже.

```
program PPFun
  if CLD[1]=50 then begin ! Команда StartSub
    ! Вывод строки, идентифицирующей начало
    ! подпрограммы (здесь "оИмяПодпрограммы"):
    Output "o"+NCSub.Name(CLD[2])
    ! Присвоение начальной метке текущего
    ! значения из регистра номера кадра:
    NCSub.StartLabel(CLD[2]) = Str(BlockN)
  end
  else if CLD[1]=51 then begin ! Команда EndSub
    ! Вывод строки, идентифицирующей конец подпрограммы (здесь
    ! "Nxx M99", где Nxx - номер кадра):
    FormBlock
    Output OutStr$ + " M99"
    ! Присвоение конечной метке значения из регистра номера кадра:
    NCSub.EndLabel(CLD[2]) = Str(BlockN)
  end
  else if CLD[1]=52 then begin ! Команда CallSub
    ! Вывод строки вызова подпрограммы вида "Nxx M98 Nbb,Nee", где
    ! Nxx - текущий кадр,
    ! Nbb - кадр начала подпр.,
    ! Nee - кадр конца подпрограммы:
    FormBlock
    Output OutStr$+" M98 "+
```

```
    NCSub.StartLabel(CLD[2])+","+
    NCSub.EndLabel(CLD[2])
    NCSub.Output(CLD[2],0)  ! Холостой прогон подпрограммы
  end
end
```

В данном примере в качестве начальной и конечной меток присваиваются текущие значения из регистра номера кадра, однако при включенной автоматической нумерации подобные действия производятся автоматически и указанный код необязателен. Он приведен только для примера.

В операторе вывода NC-подпрограммы <**NCSUB.OUTPUT**> указан режим 0, однако, т.к. все подпрограммы уже выведены в управляющую программу ранее, можно не указывать данный параметр, что будет означать вывод в режиме 1 ("только в первый раз").

Подпрограммы выводятся в УП после вывода основной программы

Предполагается, что текст основной УП вместе со строками вызова NC-подпрограмм располагается перед кодом самих подпрограмм. В этом случае оператор вывода NC-подпрограмм помещается в процедуру обработки технологической команды, которая обрабатывается последней – <**FINI**>. Общий вид процедуры представлен ниже:

```
program Fini
  ! Вывод строк, идентифицирующих конец секции основной программы и
  ! начало секции подпрограмм
  ! Инициализация переменных
  NCSub.Output ! Вывод всех имеющихся подпрограмм
  ! Вывод строк, идентифицирующих конец секции подпрограмм
end
```

Пример обработчика для технологических команд начала, конца и вызова подпрограмм (<**PPFUN STARTSUB(50)**>, <**PPFUN ENDSUB (51)**> и <**PPFUN CALLSUB(52)**>) в сравнении с предыдущим случаем не изменяется. Величины, возвращаемые функциями работы с метками (<**NCSUB.STARTLABEL**> и <**NCSUB.ENDLABEL**>), до момента окончания трансляции имеют значения по умолчанию вида <**SLabelNxxx**> и <**ELabelNxxx**>, где <**xxx**> – номер NC-подпрограммы, т.к. реальные значения меток на момент запроса еще не определены. Реальные значения меток подставляются лишь после окончания трансляции управляющей программы в целом.

Подпрограммы выводятся в УП в месте первого вызова

В данном случае подпрограмма является частью основной программы, обращение к которой может быть произведено повторно, указанием, например, начального и конечного кадров. Таким образом, подпрограмма выводится при первом вызове как часть основного кода, а при последующих – как вызов существующего кода. Весь код, отвечающий за NC-

подпрограммы, помещается в процедуру обработки технологических команд начала, конца и вызова подпрограмм (<**PPFUN STARTSUB(50)**>, <**PPFUN ENDSUB(51)**> и <**PPFUN CALLSUB(52)**>).

```
program PPFun
  if CLD[1]=50 then begin ! Команда StartSub
    ! Присвоение начальной метке текущего значения из регистра
    ! номера кадра:
    NCSub.StartLabel(CLD[2]) = Str(BlockN)
  end
  else if CLD[1]=51 then begin ! Команда EndSub
    ! Присвоение конечной метке значения из регистра номера кадра
    NCSub.EndLabel(CLD[2]) = Str(BlockN)
  end
  else if CLD[1]=52 then begin ! Команда CallSub
    if NCSub.Output(CLD[2])=0 then begin ! Если вывод подпрограммы был
      ! Вывод строки вызова подпрограммы вида "Nxx M98 Nbb,Nee", где
      ! Nxx - текущий кадр,
      ! Nbb - кадр начала подпрограммы,
      ! Nee - кадр конца подпрограммы
      FormBlock
      Output OutStr$+" M98 "+
      NCSub.StartLabel(CLD[2])+","+
      NCSub.EndLabel(CLD[2])
    end
  end
end
```

В приведенном примере первый вызов оператора <**NCSUB. OUTPUT(CLD[2])**> для каждой подпрограммы осуществит вывод NC-подпрограммы в управляющую программу и вернет значение 1. При каждом последующем вызове, будет осуществлен "холостой прогон" и будет возвращено значение 0, а в управляющую программу будет добавлена строка вызова соответствующей подпрограммы.

В данном примере в качестве начальной и конечной меток присваиваются текущие значения из регистра номера кадра. При включенной автоматической нумерации эти действия производятся автоматически, и в случае если нумерация начальных и конечный кадров подпрограммы будет совпадать с автоматически присваиваемыми значениями, указанный код необязателен.

Подпрограммы не поддерживаются

В тех случаях, когда система ЧПУ не поддерживает использование подпрограмм, а последовательность технологических команд содержит NC-подпрограммы и их вызов, то можно осуществить своеобразное "развертывание" NC-подпрограмм, т.е. во всех местах вызова подпрограммы следует вставить вместо строки вызова саму подпрограмму. Для этого в процедуру обработки команды вызова подпрограммы <**PPFUN CALLSUB(52)**> следует вставить код следующего вида:

```
program PPFun
```

```
  if CLD[1]=52 then begin   ! Команда CallSub
    NCSub.Output(CLD[2], 2) ! Вывод подпрограммы в любом случае
  end ! (режим 2)
end
```

Обработка команд начала и конца подпрограмм в данном случае не требуется.

5 Приложения

5.1 Описание технологических команд

Список технологических команд CLData:

N	Слово	Код	Значение
1	**AXESBRAKE**	2012	Управление тормозами осей станка
2	**CIRCLE**	15000	Перемещение по дуге окружности
3	**COMMENT**	1065	Комментарий
4	**COOLNT**	1030	Охлаждение
5	**CUTCOM**	1007	Компенсация инструмента
6	**CYCLE**	1054	Циклы обработки отверстий
7	**DELAY**	1010	Выстой
8	**EDMMOVE**	16001	Электроэрозионное перемещение
9	**EXTCYCLE**	1053	Расширенный цикл
10	**FEDRAT**	1009	Подача
11	**FINI**	14000	Конечная запись
12	**FROM**	5003	Начальная точка
13	**GOHOME**	17	Возврат в нулевую точку станка
14	**GOTO.abs**	5005	Линейные перемещения
15	**HEAD**	1002	Номер шпиндельной головки
16	**INSERT**	1046	Непосредственный вывод в кадр
17	**INTERPOLATION**	1047	Режим интерполяции
18	**LOADTL**	1055	Загрузка инструмента
19	**MULTIGOTO**	9010	Многокоординатные перемещения
20	**OPSKIP**	1012	Условный пропуск
21	**OPSTOP**	2003	Дополнительный останов
22	**ORIGIN**	1027	Определение системы координат
23	**PALETA**	1001	Смена палеты
24	**PARTNO**	1045	Номер детали
25	**PLANE**	99	Рабочая плоскость
26	**PPFUN**	1079	Постпроцессорная функция
27	**PPRINT**	1044	Печать постпроцессора
28	**RAPID**	2005	Быстрый ход
29	**ROTABL**	1026	Поворот стола
30	**SAFPOS**	1094	Точка смены инструмента
31	**SELWORKPIECE**	2011	Выбор активной державки заготовки
32	**SINGLETHREAD**	1037	Нарезание резьбы за один проход
33	**SPINDL**	1031	Шпиндель
34	**STOP**	2002	Останов
35	**TAKEOVER**	2010	Перехват заготовки
36	**WAIT**	2014	Ожидание точки синхронизации

Вспомогательные слова:

N	Слово	Код	Значение

1	**BOTH**	83	Оба
2	**BRKCHP**	288	Ломка стружки
3	**CCLW**	59	Против часовой стрелки
4	**CLW**	60	По часовой стрелке
5	**CUTS**	511	Проход
6	**DEEP**	153	Глубокое сверление
7	**DEEPTH**	510	Глубина
8	**DRILL**	163	Сверление
9	**DWELL**	279	Выстой
10	**FACE**	81	Сверление G82
11	**FINCUT**	512	Окончательный проход
12	**INCR**	66	Приращение
13	**LENGTH**	9	Длина
14	**MMPM**	315	Миллиметры в минуту
15	**MMPR**	316	Миллиметры на оборот
16	**MULTRD**	119	Многозаходная резьба
17	**OFF**	72	Выключить
18	**ON**	71	Включить
19	**ORIENT**	246	Ориентация
20	**TAP**	168	Нарезание резьбы
21	**XYPLAN**	33	Плоскость XY
22	**YZPLAN**	37	Плоскость YZ
23	**ZXPLAN**	41	Плоскость ZX
24	**R**	23	Радиус
25	**RGT**	24	Положение инструмента справа
26	**LEFT**	8	Положение инструмента слева

5.1.1 Управление тормозами осей станка <AXESBRAKE>

Команда <**AXESBRAKE**> предназначена для включения и выключения тормозов осей, если таковые имеются в станке. Чаще всего тормоз используется на токарно-фрезерных станках для включения/выключения тормоза шпинделя (оси C) при переходе от токарной обработки к фрезерной и обратно. При токарной обработке шпиндель осуществляет основное движение на высоких оборотах, а при фрезерной индексной обработке угловое положение шпинделя фиксируется в определенном положении и зажимается при помощи тормоза. Это позволяет увеличить усилия, которые может выдержать шпиндель и уменьшить износ механизмов точного позиционирования шпинделя. Тормоза также часто применяются на поворотных осях фрезерных станков при индексной обработке.

Команда:

```
AXESBRAKE COUNT N, Axis1Pos(n1) State1, Axis2Pos(n2) State2, ...,
                   AxisNPos(nN) StateN
```

Параметры:

Параметр	CLD массив	Описание
N	CLD[1]	Количество управляемых координат станка, присутствующих в данной команде и состояние тормоза для которых переключается.
n1	CLD[2]	Номер оси с именем Axis1Pos в списке

Параметр	CLD массив	Описание
		управляемых координат.
State1	CLD[3]	Новое состояние тормоза оси с именем Axis1Pos: ON(71) - включен, OFF(72) - выключен.
n2	CLD[4]	Номер оси с именем Axis2Pos в списке управляемых координат.
State2	CLD[5]	Новое состояние тормоза оси с именем Axis2Pos: ON(71) - включен, OFF(72) - выключен.
…	…	…
nN	CLD[2*N]	Номер оси с именем AxisNPos в списке управляемых координат.
StateN	CLD[2*N+1]	Новое состояние тормоза оси с именем AxisNPos: ON(71) - включен, OFF(72) - выключен.

Параметры, доступные через оператор Cmd

TCLDAxesBrake: ComplexType	Команда переключения состояния тормозов осей.			
	Axes: Array, Key="AxisID"	Cmd.Ptr["Axes"] - Массив структур типа AxisBrake. Таким образом, одна команда может переключать тормоза сразу нескольких осей.		
		AxisBrake: ComplexType	Cmd.Ptr["Axes"].Item[Index] или Cmd.Ptr["Axes(<AxisName>)"] - Отдельный элемент массива Axes. Содержит состояние тормоза одной оси станка. Доступ к элементам массива возможен либо по индексу, либо по ключевому полю. Здесь <AxisName> - значение ключевого поля, которое должно совпадать со значением поля AxisID.	
			AxisID: String	Cmd.Str["Axes(<AxisName>).AxisID"] - Идентификатор управляемой координаты станка, для которой задается новое состояние тормоза. Определяется схемой станка.
			BrakeState: Integer	Cmd.Int["Axes(<AxisName>).BrakeState"] - Новое состояние тормоза оси: 0 (Off) - выключен, 1 (On) - включен.

Как видно из приведенного формата, в одной команде может происходить переключение состояний сразу для нескольких осей. Список управляемых координат, имена которых появляются в данной технологической команде, определяется кинематической схемой станка SprutCAM.

Ниже приведены два простых примера программы обработки для

данной команды.

```
program AxesBrake
  Index: Integer      ! Счётчик цикла
  AxisNumber: Integer ! Номер оси в списке осей станка
  BrakeState: Integer ! Новое состояние тормоза осей
  Index = 1
  while Index<=CLD[1] do begin
    AxisNumber = CLD[2*Index]
    BrakeState = CLD[2*Index+1]
    case AxisNumber of
      4: begin ! Номер оси AxisAPos(A) в списке осей станка
        if BrakeState=71 then Output "M680" ! Включение тормоза на оси A
                         else Output "M690" ! Выключение тормоза на оси A
      end
      6: begin ! Номер оси AxisCPos(C) в списке осей станка
        if BrakeState=71 then Output "M68" ! Включение тормоза на оси C
                         else Output "M69" ! Выключение тормоза на оси C
      end
    end
    Index = Index + 1
  end
end
```

Еще один пример с использованием оператора Cmd.

```
program AxesBrake
  if Cmd.Ptr["Axes(AxisAPos)"]>0 then begin ! Ось A присутствует в данной команд
    if Cmd.Int["Axes(AxisAPos).BrakeState"]=1 then Output "M680" ! Включение тор
                                              else Output "M690" ! Выключение то
  end
  if Cmd.Ptr["Axes(AxisCPos)"]>0 then begin ! Ось C присутствует в данной команд
    if Cmd.Int["Axes(AxisCPos).BrakeState"]=1 then Output "M68" ! Включение торм
                                              else Output "M69" ! Выключение тор
  end
end
```

5.1.2 Перемещение по дуге окружности <CIRCLE>

Команда <**CIRCLE**> осуществляет перемещение инструмента по дуге окружности. В параметрах задаются координаты точки центра окружности, конечной точки дуги, а также радиус окружности. Знак радиуса окружности задает направление дуги. Если радиус положительный, то дуга направлена против часовой стрелки (круговая интерполяция G03). Если радиус отрицательный, то дуга направлена по часовой стрелке (круговая интерполяция G02). Плоскость (G17/G18/G19), в которой расположена окружность, определяется текущей плоскостью обработки (команда <**PLANE**>).

Команда:

```
CIRCLE XC xc, YC yc, ZTOOL z, R r, XK xk, YK yk, ZTOOL zk
```

Параметры:

Параметр	CLD массив		Описание
xc, yc, z	CLD[1] CLD[2] CLD[3]	CLD.Xc CLD.Yc CLD.Zc	Координаты X, Y и Z центра окружности
r	CLD[4]	CLD.R	Радиус окружности. Радиус больше нуля, если вращение против часовой стрелки (G03), и радиус меньше нуля, если вращение по часовой стрелке (G02).
xk, yk, zk	CLD[5] CLD[6] CLD[7]	CLD.Xe CLD.Ye CLD.Ze	Координаты X, Y и Z конечной точки дуги

5.1.3 Комментарий <COMMENT>

Команда <**COMMENT**> выводит строку комментария в управляющую программу. Строка комментария записывается в предопределенную переменную <**CLDATA$**>. К комментариям может быть применена транслитерация.

Команда:

```
COMMENT "........"
```

5.1.4 Охлаждение <COOLNT>

Команда <**COOLNT**> осуществляет включение и выключение систем охлаждения станка. Первым параметром передается тип команды: включение или выключение. Второй параметр содержит номер трубопровода охлаждения, к которому применяется данная команда.

Команда:

```
COOLNT ON(71)|OFF(72), N n
```

Параметры:

Параметр	CLD массив		Описание
ON(71), OFF(72)	CLD[1]	CLD.OnOff	Тип команды: ON(71) – включение охлаждения, OFF(72) – выключение охлаждения.
n	CLD[2]	CLD.N	Номер включаемого трубопровода охлаждения:

			1 – жидкость, 2 – туман, 3 – инструмент.

Для создания аналогичных ISO постпроцессоров в масках возможно использование предопределённых значений ISO.M, которые соответствуют параметрам команды:

Значение параметра CLD	Значение ISO
CLD[1] = 71	ISO.M = 8
CLD[1] = 72	ISO.M = 9

5.1.5 Компенсация инструмента <CUTCOM>

Команда <**CUTCOM**> осуществляет переключение режимов коррекции на длину и радиус инструмента.

Команда:

```
CUTCOM ON(71)|OFF(72), LENGTH(9)|R(23) lr, X x, Y y, Z z,
       XY n, YZ m, XZ k, RIGHT(24)|LEFT(8)
```

Параметры:

Параметр	CLD массив		Описание
ON(71), OFF(72)	CLD[1]	CLD.OnOff	Тип команды: 71 – включение компенсации, 72 – выключение компенсации.
LENGTH(9), R(23)	CLD[2]	CLD.Length	Вид коррекции: 9 – коррекция на длину инструмента, 23 – коррекция на радиус инструмента.
LR	CLD[3]	LCD.LR	Номер корректора на длину или радиус инструмента.
x, y, z	CLD[4], CLD[5], CLD[6]	CLD.X CLD.Y CLD.Z	Номера корректоров по осям.
n, m, k	CLD[7], CLD[8], CLD[9]	CLD.N CLD.M CLD.K	Номера корректоров в плоскостях.
RIGHT(24), LEFT(8)	CLD[10]	CLD.RGT	Направление коррекции на радиус: 24 – коррекция направо, 9 – коррекция налево.

Для создания аналогичных ISO постпроцессоров в масках возможно использование предопределённых значений ISO.G, которые соответствуют параметрам команды:

Значение параметра CLD	Значение ISO

CLD[1] = 72, CLD[2] = 9	ISO.G = 49
CLD[1] = 71, CLD[2] = 9	ISO.G = 43
CLD[1] = 72, CLD[2] = 23	ISO.G = 40
CLD[1] = 71, CLD[2] = 23, CLD[10] = 24	ISO.G = 42
CLD[1] = 71, CLD[2] = 23, CLD[10] = 8	ISO.G = 41

5.1.6 Циклы обработки отверстий <CYCLE>

Через команду <**CYCLE**> передаются стандартные циклы обработки отверстий G81 – G89. В программе обработки технологической команды <**CYCLE**> необходимо определить тип цикла и обработать команду соответствующим образом. Набор параметров команды зависит от типа цикла. Тип цикла определяется первым параметром команды.

Команда:

```
CYCLE ON(71) | OFF(72) | DRILL(163) | FACE(81) | DEEP(153) |
      BRKCHP(288) | TAP(168) | BORE5(209) | BORE6(210) | BORE7(211) |
      BORE8(212) | BORE9(213)
```

Параметры:

Параметр	CLD массив		Описание
Type	CLD[1]	CLD.Type	Тип цикла: 71(ON) – команда включения цикла, 72(OFF) – команда отмены цикла, 163(DRILL) – цикл сверления типа G81, 81(FACE) – цикл сверления типа G82, 153(DEEP) – цикл сверления с удалением стружки, 288(BRKCHP) – цикл сверления с ломкой стружки, 168(TAP) – цикл нарезания резьбы метчиком типа G84, 209(BORE5) – цикл сверления типа G85, 210(BORE6) – цикл сверления типа G86, 211(BORE7) – цикл сверления типа G87, 212(BORE8) – цикл сверления типа G88, 213(BORE9) – цикл сверления типа G89.

Последовательность технологических команд при обработке отверстий обычно следующая:

```
...
GOTO.abs X, Y, Z  ! Перемещение к первому отверстию.
CYCLE DRILL(163)  ! Вызов цикла обработки отверстия конкретного типа.
GOTO.abs X, Y, Z  ! Перемещение к следующему отверстию.
```

```
CYCLE DRILL(163)  ! Вызов цикла обработки отверстия конкретного типа.
...
GOTO.abs X, Y, Z  ! Перемещение к последнему отверстию.
CYCLE DRILL(163)  ! Вызов цикла обработки отверстия конкретного типа.
CYCLE OFF(72)     ! Отмена цикла.
...
```

По команде вызова цикла для первого отверстия следует сформировать начальный кадр задания параметров цикла, например:

```
G81 X_Y_Z_R_F_
```

Последующие команды вызова цикла обычно только переопределяют изменяющиеся параметры, а остальные параметры наследуются из предыдущей команды вызова цикла. Таким образом, формируется приблизительно такая последовательность кадров:

```
X_Y_Z_
X_Y_Z_
...
X_Y_Z_
```

По команде отмены цикла следует сформировать кадр отмены использования цикла, например:

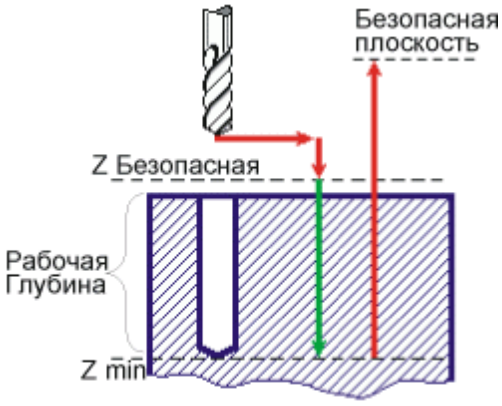
```
G80
```

Для создания аналогичных ISO постпроцессоров в масках возможно использование предопределённых значений ISO.G, которые соответствуют параметрам команды:

Значение CLD параметра	**Значение ISO**
CLD[1] = 163	ISO.G = 81
CLD[1] = 81	ISO.G = 82
CLD[1] = 168	ISO.G = 84
CLD[1] = 209	ISO.G = 85
CLD[1] = 210	ISO.G = 86
CLD[1] = 211	ISO.G = 87
CLD[1] = 212	ISO.G = 88
CLD[1] = 213	ISO.G = 89
CLD[1] = 153	ISO.G = 83
CLD[1] = 288	ISO.G = 73
CLD[1] = 72	ISO.G = 80

Цикл сверления типа G81 <CYCLE DRILL(163)>

Цикл простого сверления типа G81 производит сверление отверстия с ускоренным подходом на безопасное расстояние и последующим выходом на безопасную плоскость.



Цикл включает в себя:

- Ускоренный подвод инструмента до <**Z Безопасной**>.
- Рабочий ход инструмента на расстояние <**Z min**>.
- Ускоренный возврат инструмента до <**Безопасной плоскости** >.

Команда:

```
CYCLE DRILL(163), A a, MMPM(315), N nm, F f, P p, T t
```

или

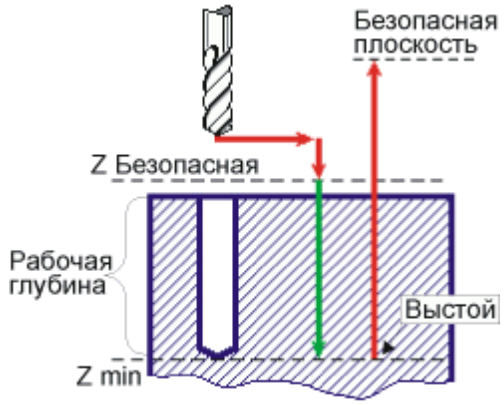
```
CYCLE DRILL(163), A a, MMPR(316), M nm, F f, P p, T t
```

Параметры:

Параметр	CLD массив		Описание
DRILL(163)	CLD[1]	CLD.Type	Тип цикла 163 (DRILL)
a	CLD[2]	CLD.A	Глубина отверстия в текущих единицах измерения (мм или дюймах)
MMPM(315), MMPR(316)	CLD[3]	CLD.MMPM	Единицы измерения подачи: 315 (MMPM) – мм/мин (дюйм/мин), 316 (MMPR) – мм/об (дюйм/об).
nm	CLD[4]	CLD.NM	Величина подачи
f	CLD[5]	CLD.F	Безопасный уровень
	CLD[6], CLD[7]	Зарезервировано	
p	CLD[8]	CLD.P	Уровень обратного хода
	CLD[9], CLD[10]	Зарезервировано	
t	CLD[11]	CLD.Top	Верхний уровень отверстия

Цикл сверления типа G82 <CYCLE FACE(81)>

Цикл сверления с задержкой типа G82 производит сверление отверстия с подходом на безопасное расстояние, выдержкой паузы при достижении уровня <**Z min**> при включенном шпинделе с последующим выходом на безопасную плоскость. Длительность паузы задается в секундах соответствующим параметром.



Цикл сверления типа G82 включает в себя:

- Ускоренный подвод инструмента до <**Z Безопасной**>.
- Рабочий ход инструмента на расстояние <**Z min**>.
- <**Выстой**> инструмента.
- Ускоренный возврат инструмента до <**Безопасной плоскости** >.

Команда:

```
CYCLE FACE(81), A a, MMPM(315), N nm, F f, P p, DWELL(279) h, T t
```

или

```
CYCLE FACE(81), A a, MMPR(316), M nm, F f, P p, DWELL(279) h, T t
```

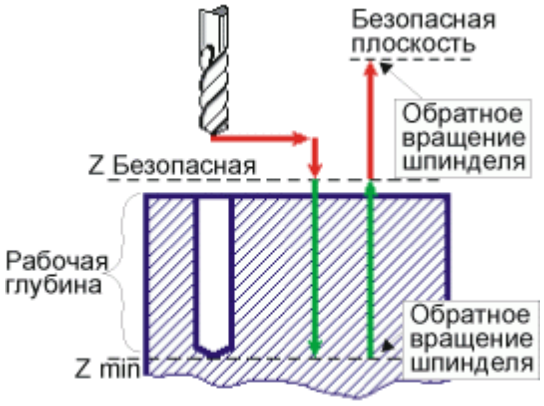
Параметры:

Параметр	CLD массив		Описание
FACE(81)	CLD[1]	CLD.Type	Тип цикла 81 (FACE)
a	CLD[2]	CLD.A	Глубина отверстия в текущих единицах измерения (мм или дюймах)
MMPM(315), MMPR(316)	CLD[3]	CLD.MMPM	Единицы измерения подачи: 315 (MMPM) – мм/мин (дюйм/мин), 316 (MMPR) – мм/об (дюйм/об).
nm	CLD[4]	CLD.NM	Величина подачи
f	CLD[5]	CLD.F	Безопасный уровень
	CLD[6], CLD[7]		Зарезервировано
p	CLD[8]	CLD.P	Уровень обратного хода
DWELL(279)	CLD[9]	CLD.DWell	Ключевое слово для времени выстоя
h	CLD[10]	CLD.H	Выстой инструмента в сек.

t	CLD[11]	CLD.Top	Верхний уровень отверстия

Цикл сверления типа G84 <CYCLE TAP(168)>

Цикл нарезание резьбы метчиком типа G84 производит подход инструмента на безопасное расстояние, нарезание резьбы с последующим подъемом на рабочем ходу при обратном вращении шпинделя.



Цикл включает в себя:

- Ускоренный подвод инструмента до <**Z Безопасной**>.
- Рабочий ход инструмента на расстояние <**Z min**>.
- <**Реверс шпинделя**> и возврат на рабочем ходу до < **Безопасной плоскости**>.
- Восстановление первоначального направления и частоты вращения шпинделя.

Команда:

```
CYCLE TAP(168), A a, MMPM(315), N nm, F f, P p, T t, S s, PS ps, SC sc,
      FR fr
```

или

```
CYCLE TAP(168), A a, MMPR(316), M nm, F f, P p, T t, S s, PS ps, SC sc,
      FR fr
```

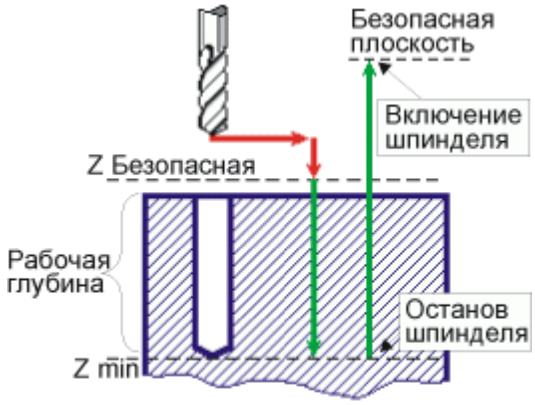
Параметры:

Параметр	CLD массив		Описание
TAP(168)	CLD[1]	CLD.Type	Тип цикла 168 (TAP)
a	CLD[2]	CLD.A	Глубина отверстия в текущих единицах измерения (мм или дюймах)
MMPM(315), MMPR(316)	CLD[3]	CLD.MMPM	Единицы измерения подачи: 315 (MMPM) – мм/мин (дюйм/мин), 316 (MMPR) – мм/об (дюйм/об).

nm	CLD[4]	CLD.NM	Величина подачи
f	CLD[5]	CLD.F	Безопасный уровень
	CLD[6], CLD[7]		Зарезервировано
p	CLD[8]	CLD.P	Уровень обратного хода
	CLD[9], CLD[10]		Зарезервировано
t	CLD[11]	CLD.Top	Верхний уровень отверстия
s	CLD[12]	CLD.Step	Шаг резьбы
ps	CLD[13]	CLD.Pos	Начальное угловое положение шпинделя
sc	CLD[14]	CLD.Socket	Тип патрона: 0 – плавающий, 1 – фиксированный
fr	CLD[15]	CLD.FR	Подача возврата

Цикл сверления типа G85 <CYCLE BORE5(209)>

Цикл расточки отверстия типа G85 производит подход инструмента на безопасное расстояние, расточку на рабочем ходу с остановкой шпинделя на минимальном уровне и выход на рабочем ходу на безопасный уровень.



Цикл включает в себя:

- Ускоренный подвод инструмента до <**Z Безопасной**>.
- Рабочий ход инструмента на расстояние <**Z min**>.
- <**Останов шпинделя**>.
- Возврат на рабочем ходу до <**Безопасной плоскости**>.
- Восстановление первоначального направления и частоты вращения шпинделя.

Команда:

```
CYCLE BORE5(209), A a, MMPM(315), N nm, F f, P p, T t
```

или

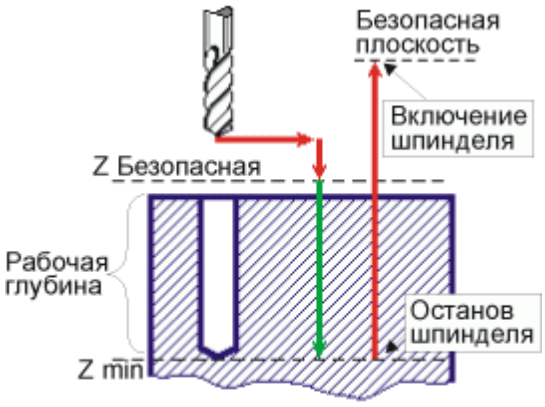
```
CYCLE BORE5(209), A a, MMPR(316), M nm, F f, P p, T t
```

Параметры:

Параметр	CLD массив		Описание
BORE5(209)	CLD[1]	CLD.Type	Тип цикла 209 (BORE5)
a	CLD[2]	CLD.A	Глубина отверстия в текущих единицах измерения (мм или дюймах)
MMPM(315), MMPR(316)	CLD[3]	CLD.MMPM	Единицы измерения подачи: 315 (MMPM) – мм/мин (дюйм/мин), 316 (MMPR) – мм/об (дюйм/об).
nm	CLD[4]	CLD.NM	Величина подачи
f	CLD[5]	CLD.F	Безопасный уровень
	CLD[6], CLD[7]		Зарезервировано
p	CLD[8]	CLD.P	Уровень обратного хода
	CLD[9], CLD[10]		Зарезервировано
t	CLD[11]	CLD.Top	Верхний уровень отверстия

Цикл сверления типа G86 <CYCLE BORE6(210)>

Цикл расточки отверстия типа G86 производит подход инструмента на безопасное расстояние, расточку на рабочем ходу с остановкой шпинделя на минимальном уровне и выход на ускоренном ходу на безопасный уровень.



Цикл включает в себя:

- Ускоренный подвод инструмента до <**Z Безопасной**>.
- Рабочий ход инструмента на расстояние <**Z min**>.
- <**Останов шпинделя**>. Если включен ориентированный останов шпинделя (параметр CLD[14]), то производится позиционирование шпинделя в заданную угловую позицию и отвод на заданные небольшие величины по координатам X, Y и

Z.

- Ускоренный возврат инструмента до <**Безопасной плоскости** >.
- Восстановление первоначального направления и частоты вращения шпинделя.

Команда:

```
CYCLE BORE6(210), A a, MMPM(315), N nm, F f, P p, T t
```

или

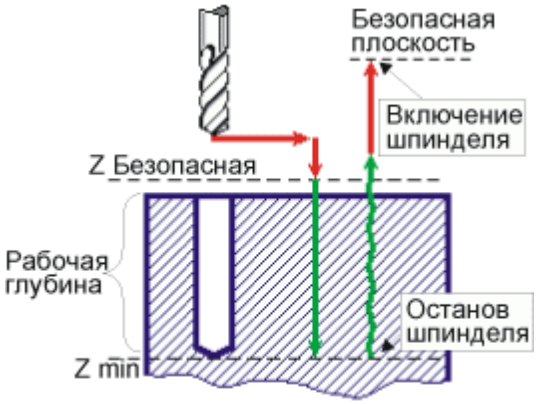
```
CYCLE BORE6(210), A a, MMPR(316), M nm, F f, P p, T t
```

Параметры:

Параметр	CLD массив		Описание
BORE6(210)	CLD[1]	CLD.Type	Тип цикла 210 (BORE6)
a	CLD[2]	CLD.A	Глубина отверстия в текущих единицах измерения (мм или дюймах)
MMPM(315), MMPR(316)	CLD[3]	CLD.MMPM	Единицы измерения подачи: 315 (MMPM) – мм/мин (дюйм/мин), 316 (MMPR) – мм/об (дюйм/об).
nm	CLD[4]	CLD.NM	Величина подачи
f	CLD[5]	CLD.F	Безопасный уровень
	CLD[6]		Величина отвода инструмента по координате X после ориентированного останова шпинделя на дне отверстия
	CLD[7]		Величина отвода инструмента по координате Y после ориентированного останова шпинделя на дне отверстия
p	CLD[8]	CLD.P	Уровень обратного хода
	CLD[9], CLD[10]		Зарезервировано
t	CLD[11]	CLD.Top	Верхний уровень отверстия
	CLD[12]		Величина отвода инструмента по координате Z после ориентированного останова шпинделя на дне отверстия
	CLD[13]		Угловое положение шпинделя при ориентированном останове на дне отверстия (в градусах)
	CLD[14]		Включен ли ориентированный останов шпинделя на дне отверстия: 0 - выключен, 1 - включен.

Цикл сверления типа G87 <CYCLE BORE7(211)>

Цикл расточки отверстия типа G87 производит подход инструмента на безопасное расстояние, расточку на рабочем ходу с остановкой шпинделя на минимальном уровне с отводом вручную до безопасной плоскости.



Цикл включает в себя:

- Ускоренный подвод инструмента до <**Z Безопасной**>.
- Рабочий ход инструмента на расстояние <**Z min**>.
- <**Останов шпинделя**>. Если включен ориентированный останов шпинделя (параметр CLD[14]), то производится позиционирование шпинделя в заданную угловую позицию и отвод на заданные небольшие величины по координатам X, Y и Z.
- Отвод вручную до <**Безопасной плоскости**>.
- Восстановление первоначального направления и частоты вращения шпинделя.

Команда:

```
CYCLE BORE7(211), A a, MMPM(315), N nm, F f, P p, T t
```

или

```
CYCLE BORE7(211), A a, MMPR(316), M nm, F f, P p, T t
```

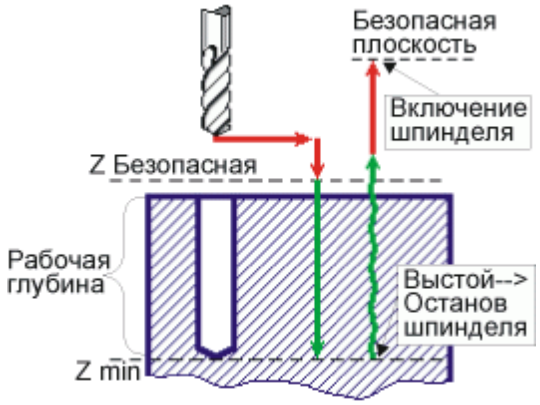
Параметры:

Параметр	CLD массив		Описание
BORE7(211)	CLD[1]	CLD.Type	Тип цикла 211 (BORE7)
a	CLD[2]	CLD.A	Глубина отверстия в текущих единицах измерения (мм или дюймах)
MMPM(315), MMPR(316)	CLD[3]	CLD.MMPM	Единицы измерения подачи: 315 (MMPM) – мм/мин (дюйм/мин), 316 (MMPR) – мм/об (дюйм/об).
nm	CLD[4]	CLD.NM	Величина подачи
f	CLD[5]	CLD.F	Безопасный уровень
	CLD[6]		Величина отвода инструмента по координате X после ориентированного останова шпинделя на дне отверстия
	CLD[7]		Величина отвода инструмента по координате Y после ориентированного останова шпинделя на дне отверстия
p	CLD[8]	CLD.P	Уровень обратного хода
	CLD[9], CLD[10]		Зарезервировано

t	CLD[11]	CLD.Top	Верхний уровень отверстия
	CLD[12]		Величина отвода инструмента по координате Z после ориентированного останова шпинделя на дне отверстия
	CLD[13]		Угловое положение шпинделя при ориентированном останове на дне отверстия (в градусах)
	CLD[14]		Включен ли ориентированный останов шпинделя на дне отверстия: 0 - выключен, 1 - включен.

Цикл сверления типа G88 <CYCLE BORE8(212)>

Цикл расточки отверстия типа G88 производит подход инструмента на безопасное расстояние, расточку на рабочем ходу с выстоем и последующей остановкой шпинделя на минимальном уровне с отводом вручную до безопасной плоскости.



Цикл включает в себя:

- Ускоренный подвод инструмента до <**Z Безопасной**>.
- Рабочий ход инструмента на расстояние <**Z min**>.
- <**Выстой**>.
- <**Останов шпинделя**>.
- Отвод вручную до <**Безопасной плоскости**>.
- Восстановление первоначального направления и частоты вращения шпинделя.

Команда:

```
CYCLE BORE8(212), A a, MMPM(315), N nm, F f, P p, DWELL(279) h, T t
```

или

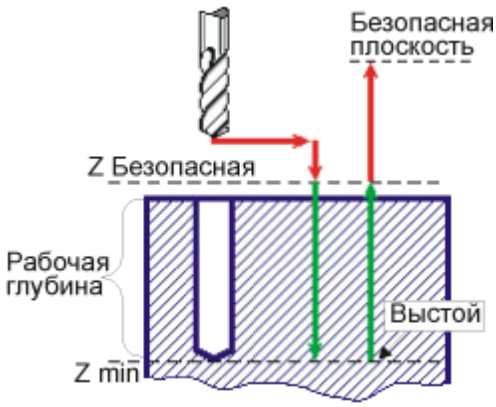
```
CYCLE BORE8(212), A a, MMPR(316), M nm, F f, P p, DWELL(279) h, T t
```

Параметры:

Параметр	CLD массив		Описание
BORE8(212)	CLD[1]	CLD.Type	Тип цикла 212 (BORE8)
a	CLD[2]	CLD.A	Глубина отверстия в текущих единицах измерения (мм или дюймах)
MMPM(315), MMPR(316)	CLD[3]	CLD.MMPM	Единицы измерения подачи: 315 (MMPM) – мм/мин (дюйм/мин), 316 (MMPR) – мм/об (дюйм/об).
nm	CLD[4]	CLD.NM	Величина подачи
f	CLD[5]	CLD.F	Безопасный уровень
	CLD[6], CLD[7]		Зарезервировано
p	CLD[8]	CLD.P	Уровень обратного хода
DWELL(279)	CLD[9]	CLD.DWELL	Ключевое слово для времени выстоя
h	CLD[10]	CLD.H	Выстой инструмента в сек
t	CLD[11]	CLD.Top	Верхний уровень отверстия

Цикл сверления типа G89 <CYCLE BORE9(213)>

Цикл расточки отверстия типа G89 производит подход инструмента на безопасное расстояние, расточку на рабочем ходу с выстоем и последующим отводом на рабочем ходу до безопасной плоскости.



Цикл включает в себя:

- Ускоренный подвод инструмента до <**Z Безопасной**>.
- Рабочий ход инструмента на расстояние <**Z min**>.
- <**Выстой**>.
- Возврат на рабочем ходу до <**Безопасной плоскости**>.

Команда:

```
CYCLE BORE9(213), A a, MMPM(315), N nm, F f, P p, DWELL(279) h, T t
```

или

```
CYCLE BORE9(213), A a, MMPR(316), M nm, F f, P p, DWELL(279) h, T t
```

Параметры:

Параметр	CLD массив		Описание
BORE9(213)	CLD[1]	CLD.Type	Тип цикла 213 (BORE9)
a	CLD[2]	CLD.A	Глубина отверстия в текущих единицах измерения (мм или дюймах)
MMPM(315), MMPR(316)	CLD[3]	CLD.MMPM	Единицы измерения подачи: 315 (MMPM) – мм/мин (дюйм/мин), 316 (MMPR) – мм/об (дюйм/об).
nm	CLD[4]	CLD.NM	Величина подачи
f	CLD[5]	CLD.F	Безопасный уровень
	CLD[6], CLD[7]		Зарезервировано
p	CLD[8]	CLD.P	Уровень обратного хода
DWELL(279)	CLD[9]	CLD.DWELL	Ключевое слово для времени выстоя
h	CLD[10]	CLD.H	Выстой инструмента в сек.
t	CLD[11]	CLD.Top	Верхний уровень отверстия

Глубокое сверление с полным выводом инструмента для удаления стружки <CYCLE DEEP(153)>

Цикл глубокого сверления отверстия с удалением стружки производит подход инструмента на безопасное расстояние и последующее циклическое сверление с учетом заданных численных значений глубины захода <**Zl**> и величины отхода <**Zi**>, которые задаются соответствующими параметрами.

Цикл включает в себя:

- Ускоренный подвод инструмента до <**Z Безопасной**>.
- Рабочий ход на глубину захода <**Zl**>.
- Ускоренный возврат инструмента до <**Z Безопасной**>.
- Ускоренный ход на расстояние <**Zl**> - <**Zi**>.
- Рабочий ход на расстояние <**Zl**> + <**Zi**>.

- Ускоренный возврат инструмента до <**Z Безопасной**>.
- Ускоренный ход на расстояние 2*<**Zl**> - <**Zi**>.
- Рабочий ход на расстояние <**Zl**> + <**Zi**>.
- Ускоренный возврат инструмента до <**Z Безопасной**>.
- Повторение предыдущих трех шагов до достижения полной глубины отверстия.
- Ускоренный возврат инструмента до <**Безопасной плоскости** >.

Команда:

```
CYCLE DEEP(153), A a, MMPM(315), N nm, F f, L l, I i, P p, T t
```

или

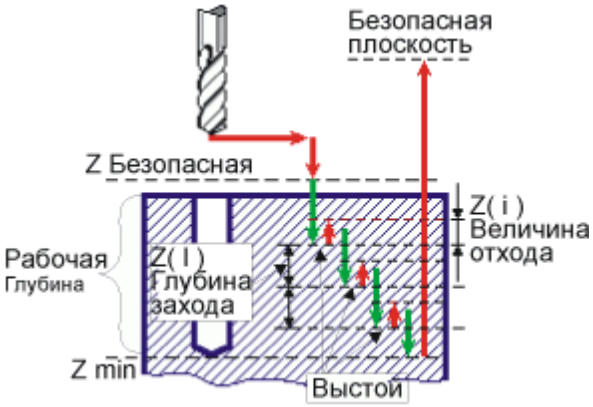
```
CYCLE DEEP(153), A a, MMPR(316), M nm, F f, L l, I i, P p, T t
```

Параметры:

Параметр	CLD массив		Описание
DEEP(153)	CLD[1]	CLD.Type	Тип цикла 153 (DEEP)
a	CLD[2]	CLD.A	Глубина отверстия в текущих единицах измерения (мм или дюймах)
MMPM(315), MMPR(316)	CLD[3]	CLD.MMPM	Единицы измерения подачи: 315 (MMPM) – мм/мин (дюйм/мин), 316 (MMPR) – мм/об (дюйм/об).
nm	CLD[4]	CLD.NM	Величина подачи
f	CLD[5]	CLD.F	Безопасный уровень
L	CLD[6]	CLD.L	Глубина захода Zl
i	CLD[7]	CLD.I	Величина отхода Zi
p	CLD[8]	CLD.P	Уровень обратного хода
	CLD[9], CLD[10]		Зарезервировано
t	CLD[11]	CLD.Top	Верхний уровень отверстия

Глубокое сверление с отводами сверла для ломки стружки <CYCLE BRKCHP(288)>

Цикл глубокого сверления отверстия с ломкой стружки производит подход инструмента на безопасное расстояние и последующее циклическое сверление с учетом заданных численных значений глубины захода <**Zl**>, величины отхода <**Zi**> и задержки, которые задаются соответствующими параметрами.



Цикл включает в себя:

- Ускоренный подвод инструмента до <**Z Безопасной**>.
- Рабочий ход на глубину захода <**Zl**>.
- Отвод на величину отхода <**Zi**> и <**Выстой**>.
- Рабочий ход на расстояние <**Zl**> + <**Zi**>.
- Отвод на величину <**Zi**> и <**Выстой**>.
- Рабочий ход на расстояние <**Zl**> + <**Zi**>.
- Отвод на величину <**Zi**> и <**Выстой**>.
- Повторение предыдущих двух шагов до достижения полной глубины отверстия.
- Ускоренный возврат инструмента до <**Безопасной плоскости** >.

Команда:

```
CYCLE BRKCHP(288), A a, MMPM(315), N nm, F f, L l, I i, P p,
      DWELL(279) h, T t
```

или

```
CYCLE BRKCHP(288), A a, MMPR(316), M nm, F f, L l, I i, P p,
      DWELL(279) h, T t
```

Параметры:

Параметр	CLD массив		Описание
BRKCHP(288)	CLD[1]	CLD.Type	Тип цикла 288 (BRKCHP)
a	CLD[2]	CLD.A	Глубина отверстия в текущих единицах измерения (мм или дюймах)
MMPM(315), MMPR(316)	CLD[3]	CLD.MMPM	Единицы измерения подачи: 315 (MMPM) – мм/мин (дюйм/мин), 316 (MMPR) – мм/об (дюйм/об).
nm	CLD[4]	CLD.NM	Величина подачи
f	CLD[5]	CLD.F	Безопасный уровень
L	CLD[6]	CLD.L	Глубина захода Zl
i	CLD[7]	CLD.I	Величина отхода Zi
p	CLD[8]	CLD.P	Уровень обратного хода
DWELL(279)	CLD[9]	CLD.DWELL	Ключевое слово для времени выстоя
h	CLD[10]	CLD.H	Выстой инструмента в сек.

t	CLD[11]	CLD.Top	Верхний уровень отверстия

5.1.7 Выстой <DELAY>

Команда <**DELAY**> осуществляет приостановку выполнения программы на заданное время. Чаще всего соответствует G-коду G04.

Команда:

```
DELAY A a
```

Параметры:

Параметр	CLD массив		Описание
a	CLD[1]	CLD.A	Время выстоя в секундах

5.1.8 Электроэрозионное перемещение <EDMMOVE>

Команда электроэрозионного перемещения <**EDMMOVE**> генерируется операциями электроэрозионной обработки SprutCAM. Она задает координаты перемещения электрода-проволоки, а также определяет режимы интерполяции перемещений.

Команда:

```
EDMMOVE MODE: RAPID(0) | 2AXIS(1) | TAPER(2) | MULTIPROF(3) | 4AXIS(4),
        SPAN1: CUT(0) | ARC(1), SPAN2: CUT(0) | ARC(1),
        EP1: (X x; Y y; Z z), EP2: (X x; Y y; Z z),
        PC1: (X x; Y y), PC2: (X x; Y y), R1 r1, R2 r2, A a,
        ROLLMODE: OFF(0) | SHARP(1) | CONICAL(2) | CYLINDRICAL(3) |
                  FIXED(4),
        ROLLR1 rollr1, ROLLR2 rollr2
```

Параметры:

Параметр	CLD массив		Описание
Режим перемещения	CLD[1]	CLD.Mode	RAPID(0) – Ускоренное перемещение, 2AXIS(1) – 2-х координатное перемещение, TAPER(2) – 2-х координатное перемещение с конусностью, MULTIPROF(3) – Многоконтурное перемещение (дуги и отрезки), 4AXIS(4) – 4-х осевое перемещение (только отрезки)
Тип элемента	CLD[2]	CLD.Span1	CUT(0) – отрезок,

Параметр	CLD массив		Описание
нижнего контура			ARC(1) – дуга окружности
Тип элемента верхнего контура	CLD[3]	CLD.Span2	CUT(0) – отрезок, ARC(1) – дуга окружности
Конечная точка элемента нижнего контура, EP1	CLD[4], CLD[5], CLD[6]	CLD.Ep1X, CLD.Ep1Y, CLD.Ep1Z	Координаты X, Y и Z, конечной точки элемента нижнего контура
Конечная точка элемента верхнего контура, EP2	CLD[7], CLD[8], CLD[9]	CLD.Ep2X, CLD.Ep2Y, CLD.Ep2Z	Координаты X, Y и Z, конечной точки элемента верхнего контура
Точка центра дуги нижнего контура, PC1	CLD[10], CLD[11]	CLD.Pc1X, CLD.Pc1Y	Координаты X и Y точки центра дуги для нижнего контура, если этот элемент является дугой.
Точка центра дуги верхнего контура, PC2	CLD[12], CLD[13]	CLD.Pc2X, CLD.Pc2Y	Координаты X и Y точки центра дуги для верхнего контура, если этот элемент является дугой.
Радиус дуги нижнего контура, R1	CLD[14]	CLD.R1	Если элементом нижнего контура является дуга окружности, то параметр содержит знаковый радиус дуги: R > 0 – дуга направлена против ч.с., R < 0 – дуга направлена по ч.с.
Радиус дуги верхнего контура, R2	CLD[15]	CLD.R2	Если элементом верхнего контура является дуга окружности, то параметр содержит знаковый радиус дуги: R > 0 – дуга направлена против ч.с., R < 0 – дуга направлена по ч.с.
Угол конусности, A	CLD[16]	CLD.A	Для режима перемещения TAPER(2) параметр содержит знаковый угол конусности: A > 0 – конусность направо, A < 0 – конусность налево
Режим обката острых углов, ROLLMODE	CLD[17]	CLD.RollMode	Определяет режим обката острых углов для 2-х координатной и конусной обработки (функция CornerR): OFF(0) – обкат углов выключен (R1 = 0, R2 = 0); SHARP(1) – острый, сглаживаются углы только нижнего контура (R1 = R1, R2 = 0); CONICAL(2) – конический, сглаживаются углы нижнего и верхнего контуров, радиусы скруглений зависят от угла конусности (R1 = R1, R2 = R1 + <Припуск конусности>); CYLINDRICAL(3) – цилиндрический, сглаживаются углы нижнего и верхнего контуров, радиусы скруглений одинаковы (R1 = R1, R2 = R1); FIXED(4) – фиксированный, сглаживаются углы нижнего и верхнего контуров, радиусы скруглений независимы (R1 = R1, R2 = R2)
Радиус обката для элемента нижнего контура, ROLLR1	CLD[18]	CLD.RollR1	Определяет радиус обката острого угла для элемента нижнего контура

Параметр	CLD массив		Описание
Радиус обката для элемента верхнего контура, ROLLR2	CLD[19]	CLD.RollR2	Определяет радиус обката острого угла для элемента верхнего контура

5.1.9 Расширенный цикл <EXTCYCLE>

Команда расширенный цикл <**EXTCYCLE**> представляет собой обобщенную технологическую команду для работы со стандартными циклами. В настоящий момент команда объединяет следующие типы стандартных циклов:

- Цикл чистового точения G70, G73 – <**LATHEFINISH(400)**>;
- Цикл чернового точения G71, G72 – <**LATHEROUGH(401)**>;
- Цикл точения канавок G74, G75 – <**LATHEGROOVE(402)**>;
- Цикл точения резьбы в несколько проходов G76 – <**LATHETHREAD(403)**>;
- Цикл точения резьбы за один проход G92 – <**LATHETHREADG92(404)**>;
- Цикл токарного сверления G81 – <**DRILL(163)**>;
- Цикл токарного сверления G82 – <**FACE(81)**>;
- Цикл токарного нарезания резьбы осевым инструментом G84 – <**TAP(168)**>;
- Цикл токарного сверления с ломкой стружки – <**BRKCHP(288)**>;
- Цикл токарного сверления с удалением стружки – <**DEEP(153)**>;
- Цикл сверления G81 – <**W5DDrill(481)**>;
- Цикл сверления G82 – <**W5DFace(482)**>;
- Цикл сверления с удалением стружки G83 – <**W5DChipRemoving(483)**>;
- Цикл сверления с ломкой стружки G73 – <**W5DChipBreaking (473)**>;
- Цикл нарезания резьбы метчиком G84 – <**W5DTap(484)**>;
- Цикл сверления G85 – <**W5DBore5(485)**>;
- Цикл сверления G86 – <**W5DBore6(486)**>;
- Цикл сверления G87 – <**W5DBore7(487)**>;
- Цикл сверления G88 – <**W5DBore8(488)**>;
- Цикл сверления G89 – <**W5DBore9(489)**>;
- Цикл фрезерования резьбы – <**W5DThreadMill(490)**>;
- Цикл выборки отверстия – <**W5DHolePocketing(491)**>.

Команда:

```
EXTCYCLE ON(71) | CALL(52) | OFF(72)
         LATHEFINISH(400) | LATHEROUGH(401) | LATHEGROOVE(402) |
         LATHETHREAD(403) | DRILL(163) | FACE(81) | TAP(168) |
         BRKCHP(288) | DEEP(153) | W5DDrill(481) | W5DFace(482) |
         W5DChipRemoving(483) | W5DChipBreaking(473) | W5DTap(484) |
         W5DBore5(485) | W5DBore6(486) | W5DBore7(487) | W5DBore8(488) |
         W5DBore9(489) | W5DThreadMill(490) | W5DHolePocketing(491)
         {, a} {, b} {, c} {, d} ...
```

Параметры:

Параметр	CLD массив		Описание
ON(71), CALL(52), OFF(72)	CLD[1]	CLD.SubCmd	Модификатор команды: ON(71) – включение стандартного цикла, CALL(52) – вызов стандартного цикла, OFF(72) – отмена стандартного цикла.
LATHEFINISH(400), LATHEROUGH(401), LATHEGROOVE(402), LATHETHREAD(403), DRILL(163), FACE(81), TAP(168), BRKCHP(288), DEEP(153), W5DDrill(481), W5DFace(482), W5DChipRemoving(483), W5DChipBreaking(473), W5DTap(484), W5DBore5(485), W5DBore6(486), W5DBore7(487), W5DBore8(488), W5DBore9(489), W5DThreadMill(490), W5DHolePocketing(491)	CLD[2]	CLD.SubType	Идентификатор конкретного типа цикла.
a, b, c, d …	CLD[3] – CLD[258]	CLD.CLParams(1) – CLD.CLParams(256)	Набор параметров, индивидуальный для каждого конкретного типа цикла.

Первый параметр команды предназначен для идентификации начала, вызова и отмены стандартного цикла. Типовая последовательность технологических команд с участием расширенного цикла следующая:

```
EXTCYCLE ON(71) ...
FEDRAT ...
GOTO ...
GOTO ...
EXTCYCLE CALL(52) ...
```

```
FEDRAT ...
GOTO ...
GOTO ...
EXTCYCLE CALL(52) ...
FEDRAT ...
GOTO ...
GOTO ...
EXTCYCLE CALL(52) ...
EXTCYCLE OFF(72) ...
```

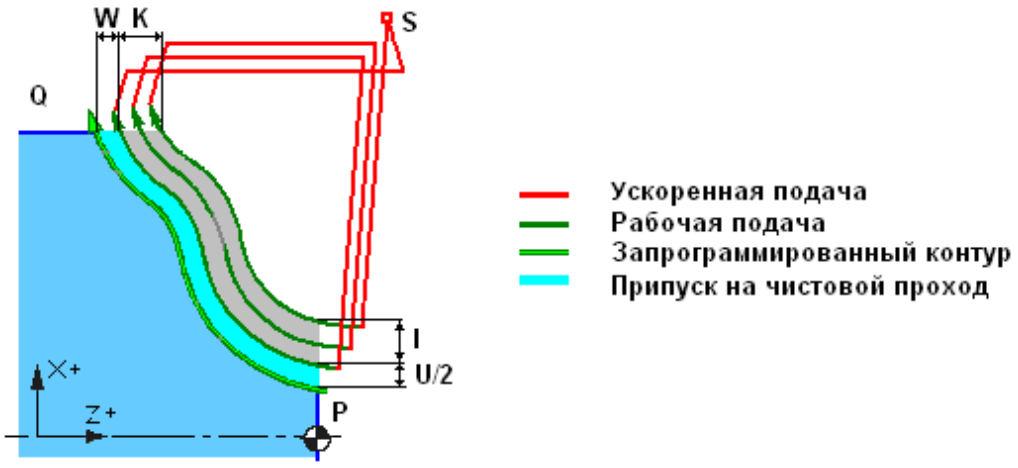
В зависимости от конкретного типа цикла набор последующих параметров меняется.

Цикл чистового точения G70, G73 <LATHEFINISH>

Стандартный цикл <**LATHEFINISH**> (G73) используется для черновой обработки предварительно сформованных (например, литых) заготовок. В данном стандартном цикле предполагается, что материал снят или отсутствует на каком-то известном расстоянии от программной траектории инструмента, блок PQ.

Действуют только F, S и T, заданные до кадра с G73 или в самом этом кадре. Все коды подачи (F), скорости шпинделя (S) или смены инструмента (T) в строках от P до Q игнорируются.

Смещение первого чернового резания определяется как (U/2 + I) для оси X и как (W + K) для оси Z. Каждый последовательный черновой проход перемещается приращениями ближе к последнему черновому проходу на (I/(D-1)) по оси X, и на (K/(D-1)) по оси Z. Последнее черновое резание всегда оставляет материал на допуск, указанный U/2 для оси X и W для оси Z.



Стандартный цикл <**LATHEFINISH**> также можно использовать для генерации цикла чистового точения вдоль контура G70. Он может быть вызван, если в параметрах все припуски равны нулю, а количество проходов равно единице.

Циклы точения предполагают наличие программного контура детали PQ. В постпроцессор программный контур передается

посредством NC-подпрограммы. Номер NC-подпрограммы передается через параметр <**CLD[3]**>.

Параметры:

CLD массив		Описание
CLD[1]	CLD.SubCmd	Модификатор команды: ON(71) – включение стандартного цикла, CALL(52) – вызов стандартного цикла, OFF(72) – отмена стандартного цикла.
CLD[2]	CLD.SubType	Идентификатор конкретного типа цикла: LATHEFINISH(400)
CLD[3]	CLD.CLParams(1)	Номер NC-подпрограммы, содержащей обрабатываемый контур PQ.
CLD[4]	CLD.CLParams(2)	Количество проходов (D). Если обработка производится за один проход D=1, имеет место цикл G70. Если D>1, то имеет место цикл G73.
CLD[5]	CLD.CLParams(3)	Знаковая толщина снимаемого слоя по оси Z (K). Для цикла G71 равна нулю.
CLD[6]	CLD.CLParams(4)	Знаковая толщина снимаемого слоя по оси X (I). Для цикла G71 равна нулю.
CLD[7]	CLD.CLParams(5)	Знаковый припуск по оси Z (W). Для цикла G71 равен нулю.
CLD[8]	CLD.CLParams(6)	Знаковый припуск по оси X (U/2). Для цикла G71 равен нулю.
CLD[11]	CLD.CLParams(9)	Тип обработки: 1 - наружная, 2 - внутренняя
CLD[12]	CLD.CLParams (10)	Наличие чистового прохода (G70): 0 - без чистового прохода, 1 - с чистовым проходом.
CLD[13]	CLD.CLParams (11)	Эквидистантный припуск для контура.

Приставка "знаковая" для величины означает, что если величина больше нуля, то она откладывается в положительном направлении соответствующей оси, а если меньше нуля – в отрицательном направлении оси.

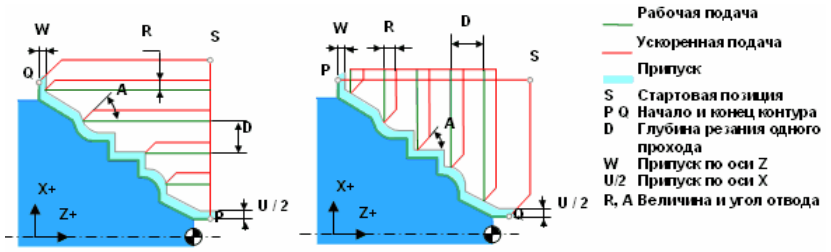
Цикл чернового точения G71, G72 <LATHEROUGH>

Стандартный цикл чернового точения G71 производит черновую обработку заготовки по заданной форме готовой детали. Команды F, S и T в строке G71 или уже установленные на момент G71, используются в цикле черновой обработки G71. Команда G71 обращается с двумя типами траекторий обработки. Если в программной траектории ось X не меняет направление, это первый тип траектории (тип I). Второй тип траектории (тип II) позволяет смену направления оси X. Смена направления оси Z недопустима для обоих типов траектории, как типа I, так и типа II.

Стандартный цикл G72 снимает материал с детали по заданной

форме готовой детали. Он аналогичен G71, но снимает материал по торцу детали.

Последним перемещением для обоих типов циклов является возврат в начальное положение S. На рисунке начальное положение S – это положение инструмента в момент вызова цикла.



Циклы точения предполагают наличие программного контура детали PQ. В постпроцессор программный контур передается посредством NC-подпрограммы. Номер NC-подпрограммы передается через параметр <**CLD[3]**>.

Если параметр CLD[12]=1, то после чернового цикла G71 или G72 может быть вызван цикл чистового прохода по контуру G70.

Параметры:

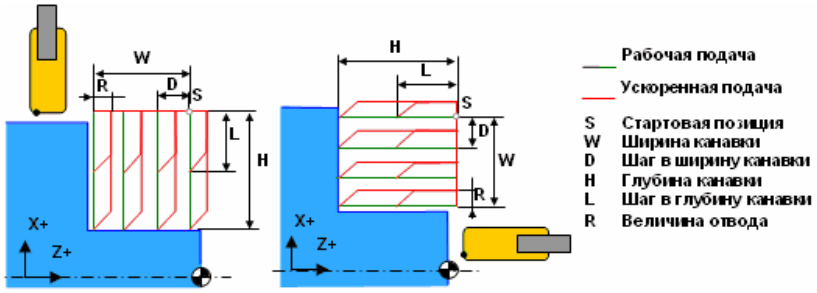
CLD массив		Описание
CLD[1]	CLD.SubCmd	Модификатор команды: ON(71) – включение стандартного цикла, CALL(52) – вызов стандартного цикла, OFF(72) – отмена стандартного цикла.
CLD[2]	CLD.SubType	Идентификатор конкретного типа цикла: LATHEROUGH(401)
CLD[3]	CLD.CLParams(1)	Номер NC-подпрограммы, содержащей обрабатываемый контур PQ.
CLD[4]	CLD.CLParams(2)	Шаг обработки (D).
CLD[5]	CLD.CLParams(3)	Направление обработки: 0 – вдоль оси вращения (G71), 1 – перпендикулярно оси вращения (G72)
CLD[6]	CLD.CLParams(4)	Тип обработки: 0 – без использования перебега, 1 – с использованием перебега (дополнительные проходы вдоль контура, для снятия гребешка)
CLD[7]	CLD.CLParams(5)	Знаковый припуск по оси Z (W)
CLD[8]	CLD.CLParams(6)	Знаковый припуск по оси X (U/2)
CLD[9]	CLD.CLParams(7)	Длина отхода (R)
CLD[10]	CLD.CLParams(8)	Угол отхода (A)
CLD[11]	CLD.CLParams(9)	Тип обработки: 1 - наружная, 2 - внутренняя

CLD массив		Описание
CLD[12]	CLD.CLParams (10)	Наличие чистового прохода (G70): 0 - без чистового прохода, 1 - с чистовым проходом.
CLD[13]	CLD.CLParams (11)	Эквидистантный припуск для контура.

Приставка "знаковая" для величины означает, что если величина больше нуля, то она откладывается в положительном направлении соответствующей оси, а если меньше нуля – в отрицательном направлении оси.

Цикл точения канавок G74, G75 <LATHEGROOVE>

Стандартный цикл <**LATHEGROOVE(402)**> реализует цикл точения цилиндрических канавок G74, и цикл точения торцевых канавок G75. Форма канавки только прямоугольная и задается своими размерами: шириной и высотой относительно начальной позиции S. Выборка материала производится последовательными рабочими ходами инструмента с шагом как по оси Z, так и по оси X.



Параметры:

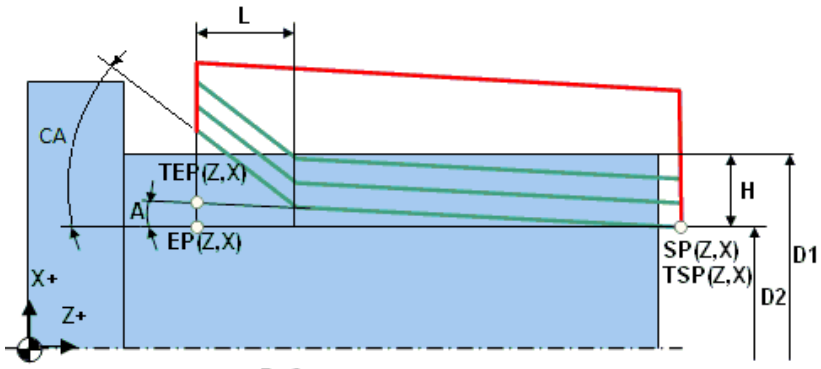
CLD массив		Описание
CLD[1]	CLD.SubCmd	Модификатор команды: ON(71) – включение стандартного цикла, CALL(52) – вызов стандартного цикла, OFF(72) – отмена стандартного цикла.
CLD[2]	CLD.SubType	Идентификатор конкретного типа цикла: LATHEGROOVE(402)
CLD[3]	CLD.CLParams(1)	Направление канавки: 0 – цилиндрическая (G75), 1 – торцевая (G74)
CLD[4]	CLD.CLParams(2)	Знаковая ширина канавки относительно стартовой позиции (W)
CLD[5]	CLD.CLParams(3)	Знаковая глубина канавки относительно стартовой позиции (H)
CLD[6]	CLD.CLParams(4)	Абсолютное значение шага в ширину

CLD массив		Описание
		(D)
CLD[7]	CLD.CLParams(5)	Абсолютное значение шага в глубину (глубина для ломки стружки, L)
CLD[8]	CLD.CLParams(6)	Абсолютное значение отскока в бок перед выводом инструмента (R)
CLD[9]	CLD.CLParams(7)	Абсолютное значение отскока вверх для ломки стружки.

Приставка "знаковая" для величины означает, что если величина больше нуля, то она откладывается в положительном направлении соответствующей оси, а если меньше нуля – в отрицательном направлении оси.

Цикл точения резьбы в несколько проходов G76 <LATHETHREAD>

Цикл нарезания резьбы в несколько проходов <**LATHETHREAD (403)**> (G76) позволяет одним кадром управляющей программы задать все параметры, необходимые станку для нарезания резьбы. При этом требуемая глубина резьбы достигается автоматически выполнением нескольких проходов. Среди параметров цикла имеются такие как координаты начальной и конечной точек резьбы, угол конуса (для конических резьб), размер фаски под выход инструмента, углы профиля, глубина резьбы, количество проходов, стратегия врезания и др.



Рабочая подача
Ускоренная подача
SP Начальная точка
EP Конечная точка
TSP Начальная точка с конусностью
TEP Конечная точка с конусностью
A Угол конуса
H Глубина резьбы
D1 Наружный диаметр
D2 Внутренний диаметр
L Выход резьбы (сбег)

Параметры:

CLD массив		Описание
CLD[1]	CLD.SubCmd	Модификатор команды: ON(71) – включение стандартного цикла, CALL(52) – вызов стандартного цикла, OFF(72) – отмена стандартного цикла.
CLD[2]	CLD.SubType	Идентификатор конкретного типа цикла: LATHETHREAD(403)
CLD[3]	CLD.CLParams(1)	Направление резьбы: 0 – левая, 1 – правая.
CLD[4]	CLD.CLParams(2)	Ориентация резьбы: 0 – наружная, 1 – внутренняя, 2 – торцевая.
CLD[5]	CLD.CLParams(3)	Угол конуса в градусах (для конической резьбы) (A).
CLD[6]	CLD.CLParams(4)	Координата Z начальной точки резьбы (SP.Z).
CLD[7]	CLD.CLParams(5)	Координата X начальной точки резьбы (SP.X).
CLD[8]	CLD.CLParams(6)	Координата Z конечной точки резьбы (EP.Z).
CLD[9]	CLD.CLParams(7)	Координата X конечной точки резьбы (EP.X).
CLD[10]	CLD.CLParams(8)	Координата Z начальной точки резьбы (с учетом конусности) (TSP.Z).
CLD[11]	CLD.CLParams(9)	Координата X начальной точки резьбы (с учетом конусности) (TSP.X).
CLD[12]	CLD.CLParams(10)	Координата Z конечной точки резьбы (с учетом конусности) (TEP.Z).
CLD[13]	CLD.CLParams(11)	Координата X конечной точки резьбы (с учетом конусности) (TEP.X).
CLD[14]	CLD.CLParams(12)	Длина выхода (подъема в конце резьбы) (L).
CLD[15]	CLD.CLParams(13)	Тип резьбы: 0 – специальная, 1 – метрическая, 2 – трубная цилиндрическая, 3 – трапецеидальная, 4 – упорная, 5 – круглая, 6 – трубная коническая, 7 – внутренняя коническая, 8 – внутренняя цилиндрическая, 9 – дюймовая коническая.
CLD[16]	CLD.CLParams(14)	Наружный диаметр (D1).
CLD[17]	CLD.CLParams(15)	Внутренний диаметр (D2).
CLD[18]	CLD.CLParams(16)	Глубина резьбы (H).
CLD[19]	CLD.CLParams(17)	Угол профиля (между кромками впадины) 
CLD[20]	CLD.CLParams(18)	Угол между вертикалью и одной из кромок впадины 

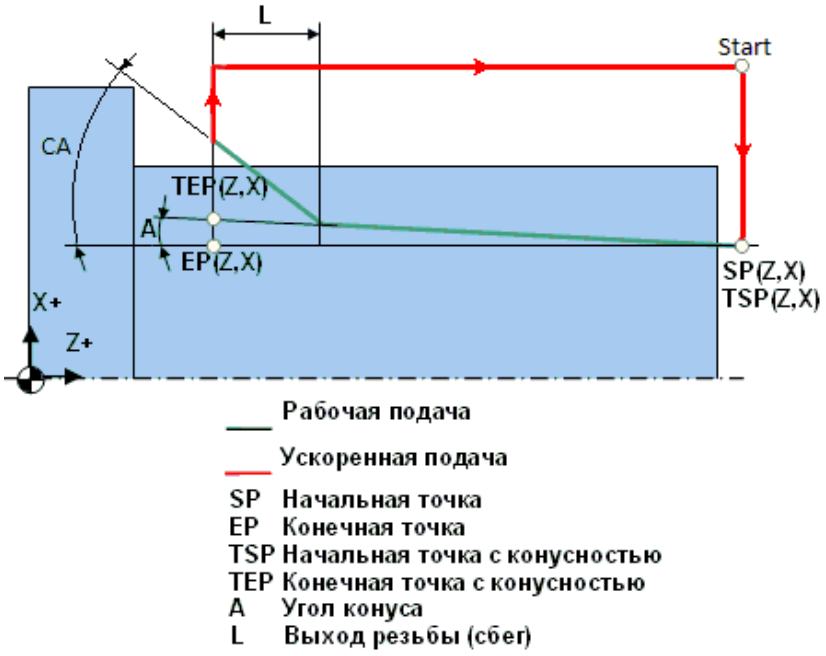
CLD массив		Описание
CLD[21]	CLD.CLParams(19)	Количество заходов резьбы.
CLD[22]	CLD.CLParams(20)	Тип задания шага резьбы: 0 – расстоянием, 1 – количеством витков на единицу длины.
CLD[23]	CLD.CLParams(21)	Значение шага резьбы.
CLD[24]	CLD.CLParams(22)	Способ врезания: 0 – радиально  1 – вдоль одной грани  2 – попеременно вдоль двух граней 
CLD[25]	CLD.CLParams(23)	Способ определения глубины резания: 0 – из условия равной площади  1 – из условия равной глубины резания 
CLD[26]	CLD.CLParams(24)	Метод определения глубины резания: 0 – по глубине первого прохода, 1 – по количеству проходов.
CLD[27]	CLD.CLParams(25)	Глубина первого прохода (использовать при методе определения глубины «0»).
CLD[28]	CLD.CLParams(26)	Количество проходов (использовать при методе определения глубины «1»).
CLD[29]	CLD.CLParams(27)	Глубина чистового реза.
CLD[30]	CLD.CLParams(28)	Количество выглаживаний.
CLD[31]	CLD.CLParams(29)	Минимальная глубина чернового прохода.
CLD[32]	CLD.CLParams(30)	Угол участка выхода из резьбы (CA) в градусах.
CLD[33]	CLD.CLParams(31)	Начальный угол шпинделя в градусах (для многозаходных резьб)

Цикл точения резьбы за один проход G92 – <LATHETHREADG92(404)>

Цикл точения резьбы за один проход <**LATHETHREADG92(404)**> (в разных стойках может иметь обозначения G92, G78, G21 и др.) генерирует замкнутую последовательность ходов для одного прохода нарезания резьбы. Схема выполнения показана на

рисунке ниже. Перед вызовом цикла инструмент находится в точке Start. Цикл вызывается одним кадром управляющей программы, в котором указывается конечная точка резьбы, шаг, размер конусности, фаски и др. В результате выполнения этого кадра инструмент из точки Start опустится в точку TSP, произведет нарезание резьбы до точки TEP и вернется в точку Start. Т.к. нарезание резьб обычно выполняют за несколько проходов, управляющая программа часто состоит из нескольких вызовов цикла с разным значением диаметра резьбы:

```
...
X60.0 Z20.0 M08
G01 Z10.0 F1.0          (Подход к точке Start)
G92 X29.4 Z-52.0 F2.0 (Вызов цикла)
X28.9                   (Модальный вызов цикла G92 с другим значением диаметра)
X28.5                   (Модальный вызов цикла G92 с другим значением диаметра)
X28.1                   (Модальный вызов цикла G92 с другим значением диаметра)
X27.8                   (Модальный вызов цикла G92 с другим значением диаметра)
X27.56                  (Модальный вызов цикла G92 с другим значением диаметра)
X27.36                  (Модальный вызов цикла G92 с другим значением диаметра)
X27.26                  (Модальный вызов цикла G92 с другим значением диаметра)
G00 X200.0Z150.0M09     (Отвод инструмента)
...
```



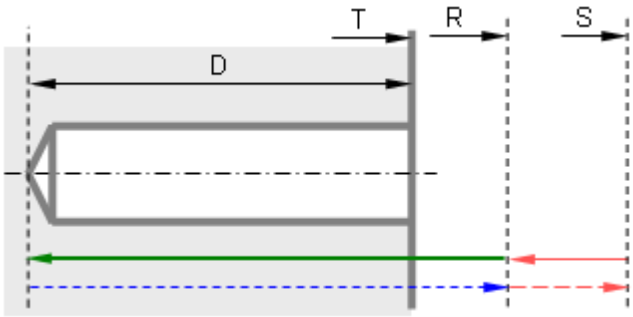
Параметры:

CLD массив		Описание
CLD[1]	CLD.SubCmd	Модификатор команды: ON(71) – включение стандартного цикла, CALL(52) – вызов стандартного цикла, OFF(72) – отмена стандартного цикла.
CLD[2]	CLD.SubType	Идентификатор конкретного типа цикла: LATHETHREADG92(404)

CLD массив		Описание
CLD[4]	CLD.CLParams(2)	Ориентация резьбы: 0 – наружная, 1 – внутренняя, 2 – торцевая.
CLD[5]	CLD.CLParams(3)	Угол конуса в градусах (для конической резьбы) (A).
CLD[6]	CLD.CLParams(4)	Координата Z начальной точки резьбы (SP.Z).
CLD[7]	CLD.CLParams(5)	Координата X начальной точки резьбы (SP.X).
CLD[8]	CLD.CLParams(6)	Координата Z конечной точки резьбы (EP.Z).
CLD[9]	CLD.CLParams(7)	Координата X конечной точки резьбы (EP.X).
CLD[10]	CLD.CLParams(8)	Координата Z начальной точки резьбы (с учетом конусности) (TSP.Z).
CLD[11]	CLD.CLParams(9)	Координата X начальной точки резьбы (с учетом конусности) (TSP.X).
CLD[12]	CLD.CLParams(10)	Координата Z конечной точки резьбы (с учетом конусности) (TEP.Z).
CLD[13]	CLD.CLParams(11)	Координата X конечной точки резьбы (с учетом конусности) (TEP.X).
CLD[14]	CLD.CLParams(12)	Длина выхода (подъема в конце резьбы) (L).
CLD[22]	CLD.CLParams(20)	Тип задания шага резьбы: 0 – расстоянием, 1 – количеством витков на единицу длины.
CLD[23]	CLD.CLParams(21)	Значение шага резьбы.
CLD[32]	CLD.CLParams(30)	Угол участка выхода из резьбы (CA) в градусах.
CLD[33]	CLD.CLParams(31)	Начальный угол шпинделя в градусах (для многозаходных резьб)

Цикл токарного сверления G81 <DRILL>

Цикл токарного сверления типа <**DRILL(163)**> (G81) осуществляет сверление осевых отверстий неподвижным инструментом во вращающейся в шпинделе заготовке. Инструмент совершает подход к отверстию на уровне безопасной плоскости, опускается на ускоренной подаче до уровня возврата, обрабатывает отверстие на рабочей подаче на нужную глубину, и возвращается на ускоренной подаче на уровень безопасной плоскости.

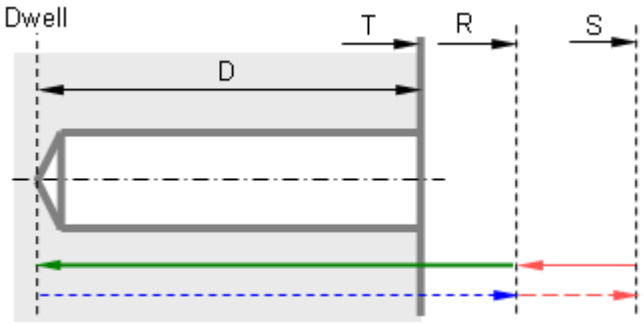


Параметры:

CLD массив		Описание
CLD[1]	CLD.SubCmd	Модификатор команды: ON(71) – включение стандартного цикла, CALL(52) – вызов стандартного цикла, OFF(72) – отмена стандартного цикла.
CLD[2]	CLD.SubType	Идентификатор конкретного типа цикла: DRILL(163)
CLD[3]	CLD.CLParams(1)	Верхний уровень отверстия (T).
CLD[4]	CLD.CLParams(2)	Глубина отверстия относительно верхнего уровня (D).
CLD[5]	CLD.CLParams(3)	Уровень возврата (R).
CLD[6]	CLD.CLParams(4)	Уровень безопасной плоскости (S).
CLD[7]	CLD.CLParams(5)	Зарезервировано.
CLD[8]	CLD.CLParams(6)	Способ задания подачи: 0 – мм/об (дюйм/об), 1 – мм/мин (дюйм/мин).
CLD[9]	CLD.CLParams(7)	Рабочая подача.
CLD[10]	CLD.CLParams(8)	Подача возврата.

Цикл токарного сверления G82 <FACE>

Цикл токарного сверления типа <**FACE(81)**> (G82) осуществляет сверление осевых отверстий неподвижным инструментом во вращающейся в шпинделе заготовке. Инструмент совершает подход к отверстию на уровне безопасной плоскости, опускается на ускоренной подаче до уровня возврата, обрабатывает отверстие на рабочей подаче на нужную глубину, производит выстой на дне отверстия, и затем возвращается на ускоренной подаче на уровень безопасной плоскости.



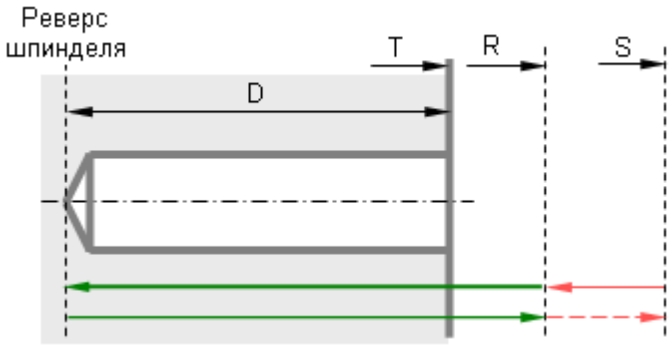
Параметры:

CLD массив		Описание
CLD[1]	CLD.SubCmd	Модификатор команды:

CLD массив		Описание
		ON(71) – включение стандартного цикла, CALL(52) – вызов стандартного цикла, OFF(72) – отмена стандартного цикла.
CLD[2]	CLD.SubType	Идентификатор конкретного типа цикла: FACE(81)
CLD[3]	CLD.CLParams(1)	Верхний уровень отверстия (T).
CLD[4]	CLD.CLParams(2)	Глубина отверстия относительно верхнего уровня (D).
CLD[5]	CLD.CLParams(3)	Уровень возврата (R).
CLD[6]	CLD.CLParams(4)	Уровень безопасной плоскости (S).
CLD[7]	CLD.CLParams(5)	Выдержка на нижнем уровне в секундах (Dwell).
CLD[8]	CLD.CLParams(6)	Способ задания подачи: 0 – мм/об (дюйм/об), 1 – мм/мин (дюйм/мин).
CLD[9]	CLD.CLParams(7)	Рабочая подача.
CLD[10]	CLD.CLParams(8)	Подача возврата.

Цикл токарного нарезания резьбы осевым инструментом G84 <TAP>

Цикл токарного нарезания резьбы осевым инструментом типа <**TAP (168)**> (G84) предназначен для нарезания резьбы неподвижным метчиком в осевом отверстии заготовки, которая вращается в шпинделе. Инструмент подходит к оси отверстия на уровне безопасной плоскости, перемещается ускоренно на уровень возврата, затем с рабочей подачей равной шагу резьбы производит обработку на заданную глубину. На дне направление вращения шпинделя инвертируется и на рабочей подаче инструмент поднимается до уровня возврата, где восстанавливаются исходные направление и частота вращения шпинделя.



Параметры:

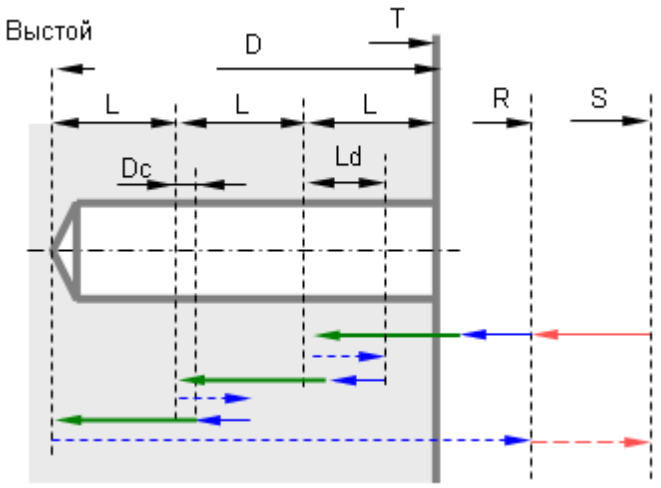
CLD массив		Описание
CLD[1]	CLD.SubCmd	Модификатор команды: ON(71) – включение стандартного цикла, CALL(52) – вызов стандартного цикла, OFF(72) – отмена стандартного цикла.
CLD[2]	CLD.SubType	Идентификатор конкретного типа цикла: TAP(168)
CLD[3]	CLD.CLParams(1)	Верхний уровень отверстия (T).
CLD[4]	CLD.CLParams(2)	Глубина отверстия относительно верхнего уровня (D).
CLD[5]	CLD.CLParams(3)	Уровень возврата (R).
CLD[6]	CLD.CLParams(4)	Уровень безопасной плоскости (S).
CLD[7]	CLD.CLParams(5)	Зарезервировано.
CLD[8]	CLD.CLParams(6)	Зарезервировано.
CLD[9]	CLD.CLParams(7)	Рабочая подача (Шаг резьбы).
CLD[10]	CLD.CLParams(8)	Подача возврата.

Цикл токарного сверления с ломкой стружки <BRKCHP>

Цикл глубокого сверления с отводами сверла для ломки стружки типа <**BRKCHP(288)**> производит обработку осевых отверстий во вращающейся в шпинделе заготовке неподвижным инструментом. Он включает в себя:

1. Ускоренный подвод инструмента до уровня осевой плоскости безопасности (S).
2. Подвод инструмента до уровня возврата (R).
3. Подход на глубину, достигнутую на предыдущей итерации (для нулевой итерации совпадает с верхним уровнем отверстия) плюс Торможение (Dc).
4. Рабочий ход на расстояние равное шагу L.
5. Выстой в течение заданного времени.
6. Отвод инструмента на заданную величину отхода Ld
7. Повторение шагов 3 – 6 до достижения полной глубины отверстия.
8. Отход на уровень возврата (R).
9. Отвод до уровня осевой плоскости безопасности (S).

Величина шага может уменьшаться на каждой итерации, если уменьшение не равно нулю.



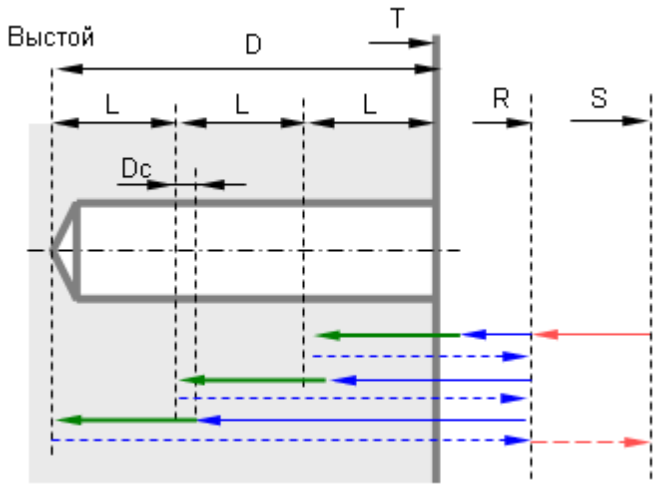
Параметры:

CLD массив		Описание
CLD[1]	CLD.SubCmd	Модификатор команды: ON(71) – включение стандартного цикла, CALL(52) – вызов стандартного цикла, OFF(72) – отмена стандартного цикла.
CLD[2]	CLD.SubType	Идентификатор конкретного типа цикла: BRKCHP(288)
CLD[3]	CLD.CLParams(1)	Верхний уровень отверстия (T).
CLD[4]	CLD.CLParams(2)	Глубина отверстия относительно верхнего уровня (D).
CLD[5]	CLD.CLParams(3)	Уровень возврата (R).
CLD[6]	CLD.CLParams(4)	Уровень безопасной плоскости (S).
CLD[7]	CLD.CLParams(5)	Выдержка на нижнем уровне в секундах (Dwell).
CLD[8]	CLD.CLParams(6)	Способ задания подачи: 0 – мм/об (дюйм/об), 1 – мм/мин (дюйм/мин).
CLD[9]	CLD.CLParams(7)	Рабочая подача.
CLD[10]	CLD.CLParams(8)	Подача возврата.
CLD[11]	CLD.CLParams(9)	Шаг для ломки стружки (L)
CLD[12]	CLD.CLParams(10)	Торможение (недоход) (Dc)
CLD[13]	CLD.CLParams(11)	Отход для ломки стружки (Ld)
CLD[14]	CLD.CLParams(12)	Уменьшение шага L на каждой итерации

Цикл токарного сверления с удалением стружки <DEEP>

Цикл глубокого сверления с полным выводом инструмента для удаления стружки типа <**DEEP(153)**> производит обработку осевых отверстий во вращающейся в шпинделе заготовке неподвижным инструментом. Он включает в себя:

1. Ускоренный подвод инструмента до уровня осевой плоскости безопасности (S).
2. Подвод инструмента до уровня возврата (R).
3. Подход на глубину, достигнутую на предыдущей итерации (для нулевой итерации совпадает с верхним уровнем отверстия) плюс Торможение (Dc).
4. Рабочий ход на расстояние равное шагу L.
5. Выстой в течение заданного времени.
6. Отход на уровень возврата (R).
7. Повторение шагов 3 – 6 до достижения полной глубины отверстия.
8. Отвод до уровня осевой плоскости безопасности (S).



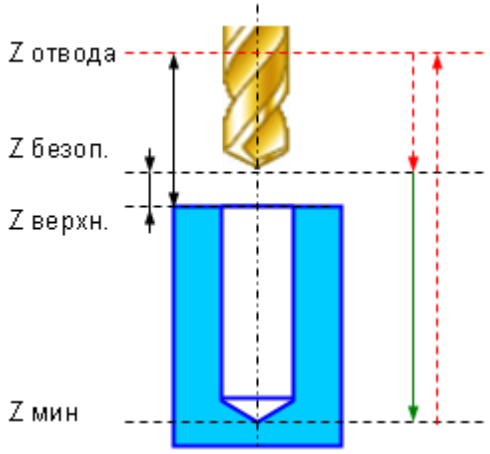
Параметры:

CLD массив		Описание
CLD[1]	CLD.SubCmd	Модификатор команды: ON(71) – включение стандартного цикла, CALL(52) – вызов стандартного цикла, OFF(72) – отмена стандартного цикла.
CLD[2]	CLD.SubType	Идентификатор конкретного типа цикла: DEEP(153)
CLD[3]	CLD.CLParams(1)	Верхний уровень отверстия (T).
CLD[4]	CLD.CLParams(2)	Глубина отверстия относительно верхнего уровня (D).
CLD[5]	CLD.CLParams(3)	Уровень возврата (R).
CLD[6]	CLD.CLParams(4)	Уровень безопасной плоскости (S).

CLD массив		Описание
CLD[7]	CLD.CLParams(5)	Выдержка на нижнем уровне в секундах (Dwell).
CLD[8]	CLD.CLParams(6)	Способ задания подачи: 0 – мм/об (дюйм/об), 1 – мм/мин (дюйм/мин).
CLD[9]	CLD.CLParams(7)	Рабочая подача.
CLD[10]	CLD.CLParams(8)	Подача возврата.
CLD[11]	CLD.CLParams(9)	Шаг для удаления стружки (L)
CLD[12]	CLD.CLParams(10)	Торможение (недоход) (Dc)
CLD[13]	CLD.CLParams(11)	Зарезервировано
CLD[14]	CLD.CLParams(12)	Уменьшение шага L на каждой итерации

Цикл сверления G81 <W5DDrill>

Цикл сверления типа <**W5DDrill(481)**> (G81) производит сверление отверстия с ускоренным подходом на <**безопасное расстояние**> и последующим выходом на <**плоскость отвода**>.



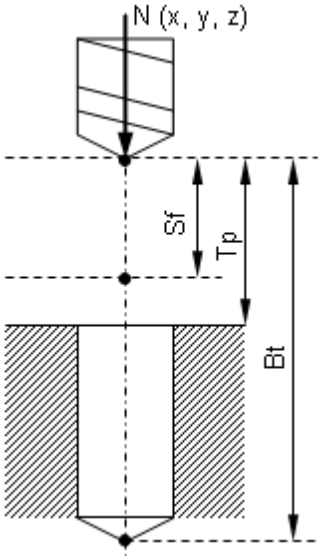
Цикл сверления типа G81 включает в себя следующие шаги:

- Ускоренный подвод инструмента к центру отверстия на уровне <**Z отвода**>.
- Опускание на ускоренном ходу до <**Z безопасной**>.
- Рабочий ход инструмента на расстояние <**Z мин**>.
- Ускоренный возврат инструмента до уровня <**Z отвода**>.

Параметры:

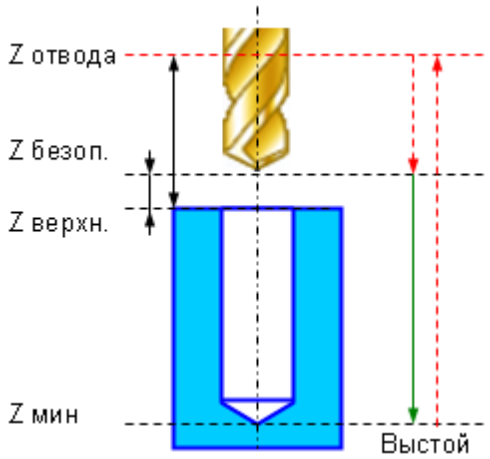
CLD массив		Описание
CLD[1]	CLD.SubCmd	Модификатор команды:

CLD массив		Описание
		ON(71) – включение стандартного цикла, CALL(52) – вызов стандартного цикла, OFF(72) – отмена стандартного цикла.
CLD[2]	CLD.SubType	Идентификатор конкретного типа цикла: W5DDrill(481)
CLD[3]	CLD.CLParams(1)	Nx, координата X вектора нормали инструмента
CLD[4]	CLD.CLParams(2)	Ny, координата Y вектора нормали инструмента
CLD[5]	CLD.CLParams(3)	Nz, координата Z вектора нормали инструмента
CLD[6]	CLD.CLParams(4)	Sf, Расстояние по нормали от текущего положения инструмента до уровня безопасной плоскости
CLD[7]	CLD.CLParams(5)	Tp, Расстояние по нормали от текущего положения инструмента до верхнего уровня отверстия
CLD[8]	CLD.CLParams(6)	Bt, Расстояние по нормали от текущего положения инструмента до нижнего уровня отверстия
CLD[9]	CLD.CLParams(7)	Единицы измерения рабочей подачи: 0 – мм/об 1 – мм/мин
CLD[10]	CLD.CLParams(8)	Величина рабочей подачи
CLD[11]	CLD.CLParams(9)	Единицы измерения подачи подвода: 0 – мм/об 1 – мм/мин
CLD[12]	CLD.CLParams(10)	Величина подачи подвода
CLD[13]	CLD.CLParams(11)	Единицы измерения подачи отвода: 0 – мм/об 1 – мм/мин
CLD[14]	CLD.CLParams(12)	Величина подачи отвода
CLD[50]	CLD.CLParams(48)	Какой шпиндель приводится в движение: 1 - приводной инструмент, 2 - шпиндель заготовки (токарный).



Цикл сверления G82 <W5DFace>

Цикл сверления типа <**W5DFace(482)**> (G82) производит сверление отверстия с подходом на <**безопасное расстояние**>, выдержкой паузы на дне при включенном шпинделе с последующим ускоренным выходом на <**плоскость отвода**>.

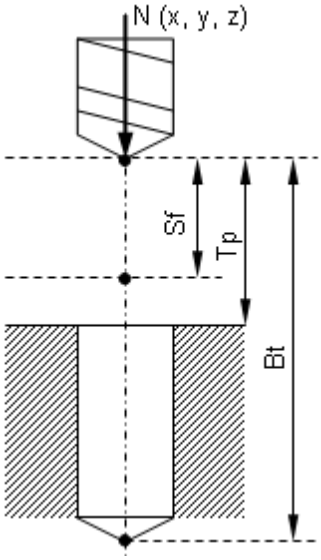


Цикл сверления типа G82 включает в себя следующие шаги:

- Ускоренный подвод инструмента к центру отверстия на уровне <**Z отвода**>.
- Опускание на ускоренном ходу до <**Z безопасной**>.
- Рабочий ход инструмента на расстояние <**Z мин**>.
- Выстой на дне отверстия.
- Ускоренный возврат инструмента до уровня <**Z отвода**>.

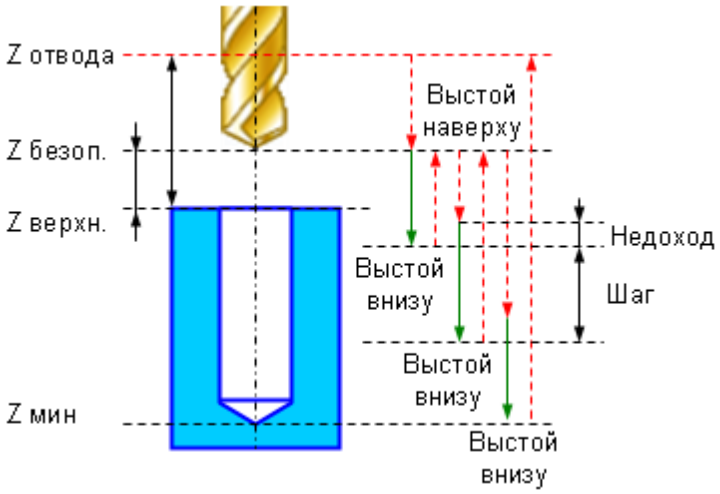
Параметры:

CLD массив		Описание
CLD[1]	CLD.SubCmd	Модификатор команды: ON(71) – включение стандартного цикла, CALL(52) – вызов стандартного цикла, OFF(72) – отмена стандартного цикла.
CLD[2]	CLD.SubType	Идентификатор конкретного типа цикла: W5DFace(482)
CLD[3]	CLD.CLParams(1)	Nx, координата X вектора нормали инструмента
CLD[4]	CLD.CLParams(2)	Ny, координата Y вектора нормали инструмента
CLD[5]	CLD.CLParams(3)	Nz, координата Z вектора нормали инструмента
CLD[6]	CLD.CLParams(4)	Sf, Расстояние по нормали от текущего положения инструмента до уровня безопасной плоскости
CLD[7]	CLD.CLParams(5)	Tp, Расстояние по нормали от текущего положения инструмента до верхнего уровня отверстия
CLD[8]	CLD.CLParams(6)	Bt, Расстояние по нормали от текущего положения инструмента до нижнего уровня отверстия
CLD[9]	CLD.CLParams(7)	Единицы измерения рабочей подачи: 0 – мм/об 1 – мм/мин
CLD[10]	CLD.CLParams(8)	Величина рабочей подачи
CLD[11]	CLD.CLParams(9)	Единицы измерения подачи подвода: 0 – мм/об 1 – мм/мин
CLD[12]	CLD.CLParams(10)	Величина подачи подвода
CLD[13]	CLD.CLParams(11)	Единицы измерения подачи отвода: 0 – мм/об 1 – мм/мин
CLD[14]	CLD.CLParams(12)	Величина подачи отвода
CLD[15]	CLD.CLParams(13)	Величина задержки на нижнем уровне отверстия в секундах
CLD[50]	CLD.CLParams(48)	Какой шпиндель приводится в движение: 1 - приводной инструмент, 2 - шпиндель заготовки (токарный).



Цикл сверления с удалением стружки G83 <W5DChipRemoving>

Цикл сверления с удалением стружки типа <**W5DChipRemoving (483)**> (G83) предполагает подход инструмента к центру отверстия на расстоянии <**Z отвода**> и последующее циклическое сверление с периодическим выводом сверла на уровень <**Z безопасная**>.



Цикл включает в себя следующее:

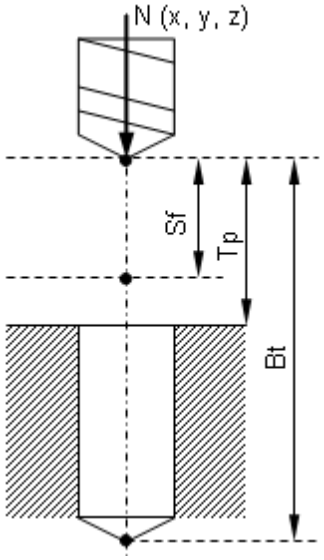
- Ускоренный подвод инструмента к центру отверстия на уровне <**Z отвода**>.
- Опускание на ускоренной подаче до <**Z безопасной**>.
- Рабочий ход на глубину равную параметру <**Шаг (S)**>.
- Выстой инструмента продолжительностью <**Выстой внизу**>.
- Ускоренный возврат инструмента до <**Z безопасной**>.

- Выстой инструмента продолжительностью <**Выстой наверху**>.
- Ускоренное перемещение до уровня, достигнутого на предыдущем рабочем ходе, с небольшим недоходом до него на величину <**Недоход (Dcl)**>.
- Рабочий ход на расстояние <**Недоход (Dcl)**> плюс <**Шаг (S)**>.
- Выстой инструмента продолжительностью <**Выстой внизу**>.
- Повторение предыдущих пяти шагов до достижения полной глубины отверстия.
- Ускоренный возврат инструмента до уровня <**Z отвода**>.

Параметры:

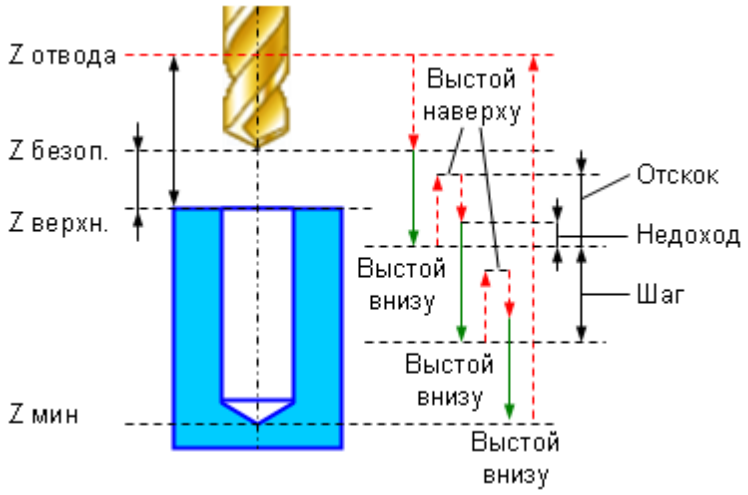
CLD массив		Описание
CLD[1]	CLD.SubCmd	Модификатор команды: ON(71) – включение стандартного цикла, CALL(52) – вызов стандартного цикла, OFF(72) – отмена стандартного цикла.
CLD[2]	CLD.SubType	Идентификатор конкретного типа цикла: W5DChipRemoving(483)
CLD[3]	CLD.CLParams(1)	Nx, координата X вектора нормали инструмента
CLD[4]	CLD.CLParams(2)	Ny, координата Y вектора нормали инструмента
CLD[5]	CLD.CLParams(3)	Nz, координата Z вектора нормали инструмента
CLD[6]	CLD.CLParams(4)	Sf, расстояние по нормали от текущего положения инструмента до уровня безопасной плоскости
CLD[7]	CLD.CLParams(5)	Tp, расстояние по нормали от текущего положения инструмента до верхнего уровня отверстия
CLD[8]	CLD.CLParams(6)	Bt, расстояние по нормали от текущего положения инструмента до нижнего уровня отверстия
CLD[9]	CLD.CLParams(7)	Единицы измерения рабочей подачи: 0 – мм/об 1 – мм/мин
CLD[10]	CLD.CLParams(8)	Величина рабочей подачи
CLD[11]	CLD.CLParams(9)	Единицы измерения подачи подвода: 0 – мм/об 1 – мм/мин
CLD[12]	CLD.CLParams(10)	Величина подачи подвода
CLD[13]	CLD.CLParams(11)	Единицы измерения подачи отвода: 0 – мм/об 1 – мм/мин
CLD[14]	CLD.CLParams(12)	Величина подачи отвода
CLD[15]	CLD.CLParams(13)	Величина задержки на нижнем уровне отверстия в секундах
CLD[16]	CLD.CLParams(14)	Величина задержки на верхнем уровне отверстия в секундах

CLD массив		Описание
CLD[17]	CLD.CLParams(15)	St, значение шага для удаления стружки
CLD[18]	CLD.CLParams(16)	Dg, величина уменьшения шага для удаления стружки на каждой итерации
CLD[19]	CLD.CLParams(17)	Dc, недоход до места начала резания на каждом шаге
CLD[50]	CLD.CLParams(48)	Какой шпиндель приводится в движение: 1 - приводной инструмент, 2 - шпиндель заготовки (токарный).



Цикл сверления с ломкой стружки G73 <W5DChipBreaking>

Цикл сверления с ломкой стружки типа <**W5DChipBreaking(473)**> (G73) предполагает подход инструмента к центру отверстия на расстоянии <**Z отвода**>. Затем осуществляется циклическое сверление с периодическим небольшим отводом инструмента с целью прерывания процесса образования стружки.



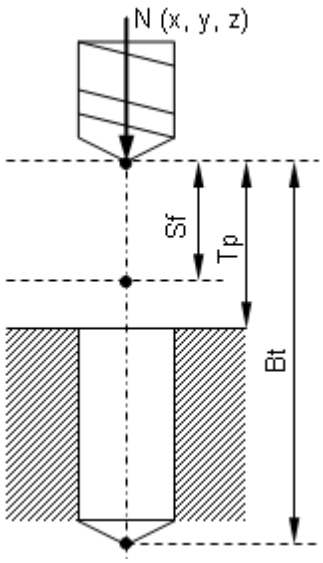
Цикл включает в себя следующее:

- Ускоренный подвод инструмента к центру отверстия на уровне <**Z отвода**>.
- Опускание на ускоренной подаче до <**Z безопасной**>.
- Рабочий ход на глубину равную параметру <**Шаг (S)**>.
- Выстой инструмента продолжительностью <**Выстой внизу**>.
- Ускоренный отвод инструмента на величину <**Отскок (Ld)**>.
- Выстой инструмента продолжительностью <**Выстой наверху**>.
- Ускоренное перемещение до уровня, достигнутого на предыдущем рабочем ходе, с небольшим недоходом до него на величину <**Недоход (Dcl)**>.
- Рабочий ход на расстояние <**Недоход (Dcl)**> плюс <**Шаг (S)**>.
- Выстой инструмента продолжительностью <**Выстой внизу**>.
- Повторение предыдущих пяти шагов до достижения полной глубины отверстия.
- Ускоренный возврат инструмента до уровня <**Z отвода**>.

Параметры:

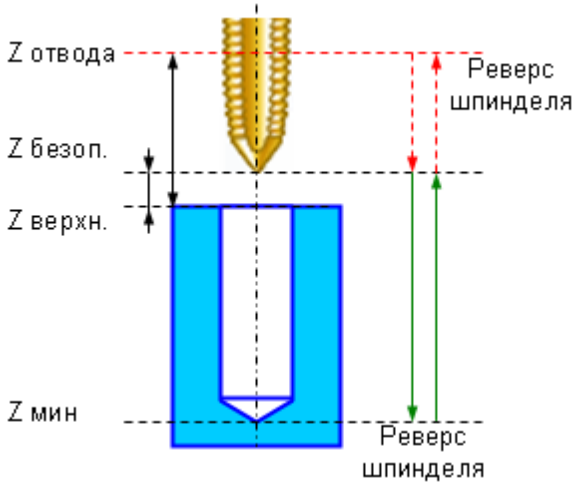
CLD массив		Описание
CLD[1]	CLD.SubCmd	Модификатор команды: ON(71) – включение стандартного цикла, CALL(52) – вызов стандартного цикла, OFF(72) – отмена стандартного цикла.
CLD[2]	CLD.SubType	Идентификатор конкретного типа цикла: W5DChipBreaking(473)
CLD[3]	CLD.CLParams(1)	Nx, координата X вектора нормали инструмента
CLD[4]	CLD.CLParams(2)	Ny, координата Y вектора нормали инструмента
CLD[5]	CLD.CLParams(3)	Nz, координата Z вектора нормали инструмента

CLD массив		Описание
CLD[6]	CLD.CLParams(4)	Sf, Расстояние по нормали от текущего положения инструмента до уровня безопасной плоскости
CLD[7]	CLD.CLParams(5)	Tp, Расстояние по нормали от текущего положения инструмента до верхнего уровня отверстия
CLD[8]	CLD.CLParams(6)	Bt, Расстояние по нормали от текущего положения инструмента до нижнего уровня отверстия
CLD[9]	CLD.CLParams(7)	Единицы измерения рабочей подачи: 0 – мм/об 1 – мм/мин
CLD[10]	CLD.CLParams(8)	Величина рабочей подачи
CLD[11]	CLD.CLParams(9)	Единицы измерения подачи подвода: 0 – мм/об 1 – мм/мин
CLD[12]	CLD.CLParams(10)	Величина подачи подвода
CLD[13]	CLD.CLParams(11)	Единицы измерения подачи отвода: 0 – мм/об 1 – мм/мин
CLD[14]	CLD.CLParams(12)	Величина подачи отвода
CLD[15]	CLD.CLParams(13)	Величина задержки на нижнем уровне отверстия в секундах
CLD[16]	CLD.CLParams(14)	Величина задержки на верхнем уровне отверстия в секундах
CLD[17]	CLD.CLParams(15)	St, значение шага для ломки стружки
CLD[18]	CLD.CLParams(16)	Dg, величина уменьшения шага для ломки стружки на каждой итерации
CLD[19]	CLD.CLParams(17)	Dc, недоход до места начала резания на каждом шаге
CLD[20]	CLD.CLParams(18)	Ld, значение отхода на каждом шаге для прерывания процесса резания
CLD[50]	CLD.CLParams(48)	Какой шпиндель приводится в движение: 1 - приводной инструмент, 2 - шпиндель заготовки (токарный).



Цикл нарезания резьбы метчиком G84 <W5DTap>

Цикл нарезания резьбы метчиком <**W5DTap(484)**> (G84) предполагает подход инструмента к центру отверстия на расстоянии <**Z отвода**>, нарезание резьбы с последующим подъемом на рабочем ходу при обратном вращении шпинделя.



Цикл нарезания резьбы типа G84 включает в себя следующее:

- Ускоренный подход инструмента к центру отверстия на уровне <**Z отвода**>.
- Ускоренное опускание до <**Z безопасной**>.
- Рабочий ход инструмента на расстояние <**Z мин**>.
- Реверс шпинделя и возврат на рабочем ходу до <**Z безопасной**>.
- Ускоренный подъем до уровня <**Z отвода**>.

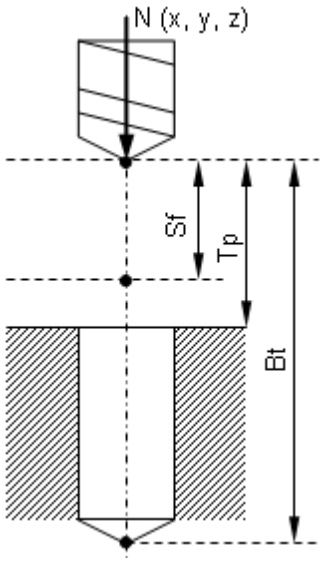
- Восстановление первоначального направления и частоты вращения шпинделя.

При нарезании резьб метчиком часто используют патроны специальной конструкции, которые повышают точность обрабатываемых отверстий – плавающие патроны. Их применяют, когда необходимо скомпенсировать несоосность обрабатываемого отверстия и инструмента. Стойки ЧПУ могут иметь разные циклы для нарезания резьб с использованием различных типов патронов. Поэтому выбор типа патрона вынесен в соответствующий параметр, который может принимать два значения: плавающий и фиксированный.

Параметры:

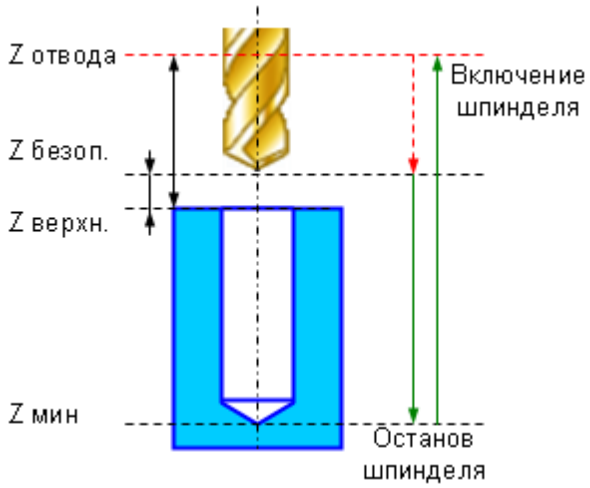
CLD массив		Описание
CLD[1]	CLD.SubCmd	Модификатор команды: ON(71) – включение стандартного цикла, CALL(52) – вызов стандартного цикла, OFF(72) – отмена стандартного цикла.
CLD[2]	CLD.SubType	Идентификатор конкретного типа цикла: W5DTap(484)
CLD[3]	CLD.CLParams(1)	Nx, координата X вектора нормали инструмента
CLD[4]	CLD.CLParams(2)	Ny, координата Y вектора нормали инструмента
CLD[5]	CLD.CLParams(3)	Nz, координата Z вектора нормали инструмента
CLD[6]	CLD.CLParams(4)	Sf, Расстояние по нормали от текущего положения инструмента до уровня безопасной плоскости
CLD[7]	CLD.CLParams(5)	Tp, Расстояние по нормали от текущего положения инструмента до верхнего уровня отверстия
CLD[8]	CLD.CLParams(6)	Bt, Расстояние по нормали от текущего положения инструмента до нижнего уровня отверстия
CLD[9]	CLD.CLParams(7)	Единицы измерения рабочей подачи: 0 – мм/об 1 –мм/мин
CLD[10]	CLD.CLParams(8)	Величина рабочей подачи
CLD[11]	CLD.CLParams(9)	Единицы измерения подачи подвода: 0 – мм/об 1 – мм/мин
CLD[12]	CLD.CLParams(10)	Величина подачи подвода
CLD[13]	CLD.CLParams(11)	Единицы измерения подачи отвода: 0 – мм/об 1 – мм/мин
CLD[14]	CLD.CLParams(12)	Величина подачи отвода
CLD[17]	CLD.CLParams(15)	Шаг резьбы
CLD[18]	CLD.CLParams(16)	Начальное угловое положение шпинделя в градусах (для

CLD массив		Описание
		многозаходных резьб)
CLD[19]	CLD.CLParams(17)	Тип патрона метчика: 0 – плавающий (компенсирующий), 1 – фиксированный (некомпенсирующий)
CLD[50]	CLD.CLParams(48)	Какой шпиндель приводится в движение: 1 - приводной инструмент, 2 - шпиндель заготовки (токарный).



Цикл сверления типа G85 <W5DBore5>

Цикл сверления типа <**W5DBore5(485)**> (G85) предполагает подход инструмента к центру отверстия, расточку на рабочем ходу с остановкой шпинделя на минимальном уровне и выход на рабочем ходу на <**уровень отвода**>.



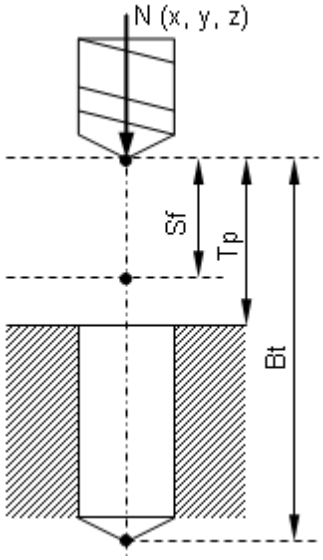
Цикл расточки типа G85 включает в себя следующие шаги:

- Ускоренный подвод инструмента к центру отверстия на уровне <**Z отвода**>.
- Опускание на ускоренном ходу до <**Z безопасной**>.
- Рабочий ход инструмента на расстояние <**Z мин**>.
- Останов шпинделя.
- Возврат инструмента на рабочем ходу до уровня <**Z отвода**>.
- Восстановление первоначальных направления и частоты вращения шпинделя.

Параметры:

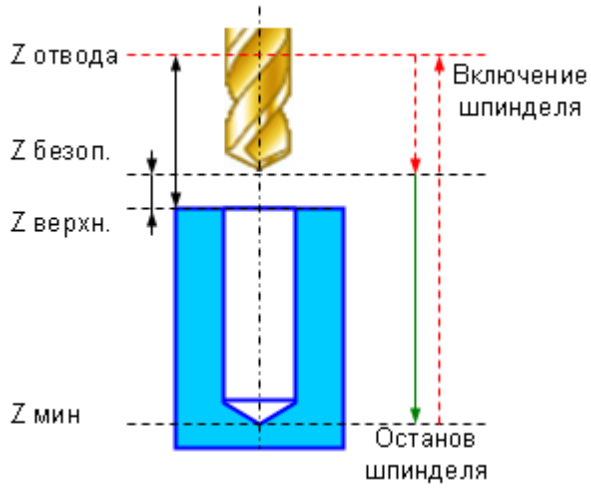
CLD массив		Описание
CLD[1]	CLD.SubCmd	Модификатор команды: ON(71) – включение стандартного цикла, CALL(52) – вызов стандартного цикла, OFF(72) – отмена стандартного цикла.
CLD[2]	CLD.SubType	Идентификатор конкретного типа цикла: W5DBore5(485)
CLD[3]	CLD.CLParams(1)	Nx, координата X вектора нормали инструмента
CLD[4]	CLD.CLParams(2)	Ny, координата Y вектора нормали инструмента
CLD[5]	CLD.CLParams(3)	Nz, координата Z вектора нормали инструмента
CLD[6]	CLD.CLParams(4)	Sf, Расстояние по нормали от текущего положения инструмента до уровня безопасной плоскости
CLD[7]	CLD.CLParams(5)	Tp, Расстояние по нормали от текущего положения инструмента до верхнего уровня отверстия
CLD[8]	CLD.CLParams(6)	Bt, Расстояние по нормали от текущего положения инструмента до нижнего уровня отверстия

CLD массив		Описание
CLD[9]	CLD.CLParams(7)	Единицы измерения рабочей подачи: 0 – мм/об 1 – мм/мин
CLD[10]	CLD.CLParams(8)	Величина рабочей подачи
CLD[11]	CLD.CLParams(9)	Единицы измерения подачи подвода: 0 – мм/об 1 – мм/мин
CLD[12]	CLD.CLParams(10)	Величина подачи подвода
CLD[13]	CLD.CLParams(11)	Единицы измерения подачи отвода: 0 – мм/об 1 – мм/мин
CLD[14]	CLD.CLParams(12)	Величина подачи отвода
CLD[50]	CLD.CLParams(48)	Какой шпиндель приводится в движение: 1 - приводной инструмент, 2 - шпиндель заготовки (токарный).



Цикл сверления типа G86 <W5DBore6>

Цикл сверления типа <**W5DBore6(486)**> (G86) предполагает подход инструмента к центру отверстия, расточку на рабочем ходу с остановкой шпинделя на минимальном уровне и выход на ускоренном ходу на <**уровень отвода**>.



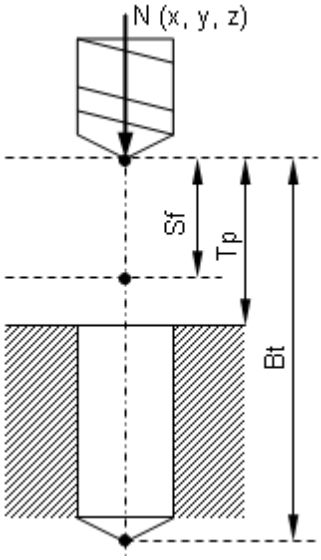
Цикл расточки типа G86 включает в себя следующие шаги:

- Ускоренный подвод инструмента к центру отверстия на уровне <**Z отвода**>.
- Опускание на ускоренном ходу до <**Z безопасной**>.
- Рабочий ход инструмента на расстояние <**Z мин**>.
- Останов шпинделя. Если включен ориентированный останов шпинделя (параметр CLD[17]), то производится позиционирование шпинделя в заданную угловую позицию и отвод на заданные небольшие величины по координатам X, Y и Z.
- Возврат инструмента на ускоренном ходу до уровня <**Z отвода** >.
- Восстановление первоначальных направления и частоты вращения шпинделя.

Параметры:

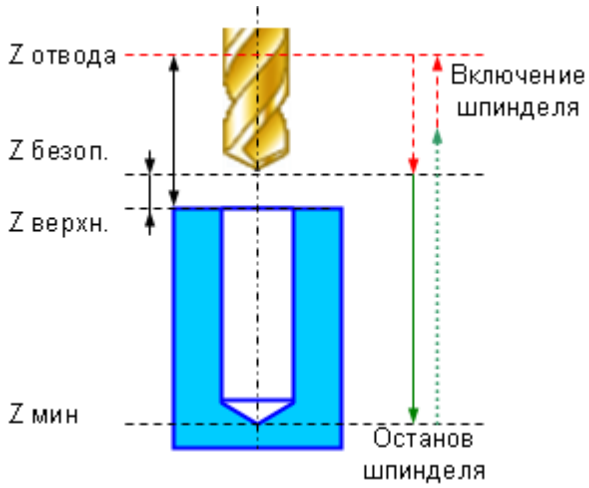
CLD массив		Описание
CLD[1]	CLD.SubCmd	Модификатор команды: ON(71) – включение стандартного цикла, CALL(52) – вызов стандартного цикла, OFF(72) – отмена стандартного цикла.
CLD[2]	CLD.SubType	Идентификатор конкретного типа цикла: W5DBore6(486)
CLD[3]	CLD.CLParams(1)	Nx, координата X вектора нормали инструмента
CLD[4]	CLD.CLParams(2)	Ny, координата Y вектора нормали инструмента
CLD[5]	CLD.CLParams(3)	Nz, координата Z вектора нормали инструмента
CLD[6]	CLD.CLParams(4)	Sf, Расстояние по нормали от текущего положения инструмента до уровня безопасной плоскости
CLD[7]	CLD.CLParams(5)	Tp, Расстояние по нормали от

CLD массив		Описание
		текущего положения инструмента до верхнего уровня отверстия
CLD[8]	CLD.CLParams(6)	Bt, Расстояние по нормали от текущего положения инструмента до нижнего уровня отверстия
CLD[9]	CLD.CLParams(7)	Единицы измерения рабочей подачи: 0 – мм/об 1 – мм/мин
CLD[10]	CLD.CLParams(8)	Величина рабочей подачи
CLD[11]	CLD.CLParams(9)	Единицы измерения подачи подвода: 0 – мм/об 1 – мм/мин
CLD[12]	CLD.CLParams(10)	Величина подачи подвода
CLD[13]	CLD.CLParams(11)	Единицы измерения подачи отвода: 0 – мм/об 1 – мм/мин
CLD[14]	CLD.CLParams(12)	Величина подачи отвода
CLD[17]	CLD.CLParams(15)	Включен ли ориентированный останов шпинделя на дне отверстия: 0 - выключен, 1 - включен.
CLD[18]	CLD.CLParams(16)	Угловое положение шпинделя при ориентированном останове на дне отверстия (в градусах)
CLD[19]	CLD.CLParams(17)	Величина отвода инструмента по координате X после ориентированного останова шпинделя на дне отверстия
CLD[20]	CLD.CLParams(18)	Величина отвода инструмента по координате Y после ориентированного останова шпинделя на дне отверстия
CLD[21]	CLD.CLParams(19)	Величина отвода инструмента по координате Z после ориентированного останова шпинделя на дне отверстия
CLD[50]	CLD.CLParams(48)	Какой шпиндель приводится в движение: 1 - приводной инструмент, 2 - шпиндель заготовки (токарный).



Цикл сверления типа G87 <W5DBore7>

Цикл сверления типа <**W5DBore7(487)**> (G87) предполагает подход инструмента к центру отверстия, расточку на рабочем ходу с остановкой шпинделя на минимальном уровне и вывод инструмента вручную на <**уровень отвода**>.



Цикл расточки типа G87 включает в себя следующие шаги.

- Ускоренный подвод инструмента к центру отверстия на уровне <**Z отвода**>.
- Опускание на ускоренном ходу до <**Z безопасной**>.
- Рабочий ход инструмента на расстояние <**Z мин**>.
- Останов шпинделя. Если включен ориентированный останов шпинделя (параметр CLD[17]), то производится позиционирование шпинделя в заданную угловую позицию и отвод на заданные небольшие величины по координатам X, Y и

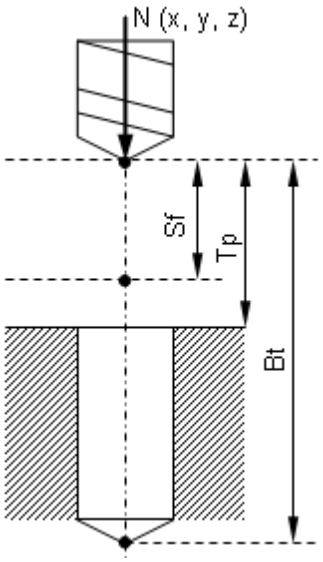
Z.

- Ручной отвод инструмента до уровня <**Z отвода**>.
- Восстановление первоначальных направления и частоты вращения шпинделя.

Параметры:

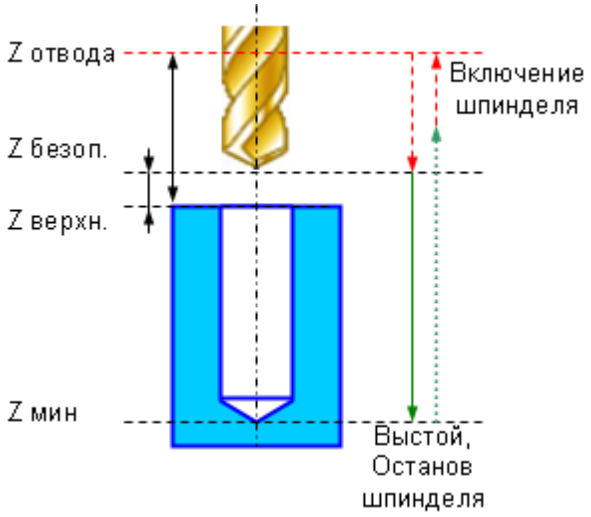
CLD массив		Описание
CLD[1]	CLD.SubCmd	Модификатор команды: ON(71) – включение стандартного цикла, CALL(52) – вызов стандартного цикла, OFF(72) – отмена стандартного цикла.
CLD[2]	CLD.SubType	Идентификатор конкретного типа цикла: W5DBore7(487)
CLD[3]	CLD.CLParams(1)	Nx, координата X вектора нормали инструмента
CLD[4]	CLD.CLParams(2)	Ny, координата Y вектора нормали инструмента
CLD[5]	CLD.CLParams(3)	Nz, координата Z вектора нормали инструмента
CLD[6]	CLD.CLParams(4)	Sf, Расстояние по нормали от текущего положения инструмента до уровня безопасной плоскости
CLD[7]	CLD.CLParams(5)	Tp, Расстояние по нормали от текущего положения инструмента до верхнего уровня отверстия
CLD[8]	CLD.CLParams(6)	Bt, Расстояние по нормали от текущего положения инструмента до нижнего уровня отверстия
CLD[9]	CLD.CLParams(7)	Единицы измерения рабочей подачи: 0 – мм/об 1 – мм/мин
CLD[10]	CLD.CLParams(8)	Величина рабочей подачи
CLD[11]	CLD.CLParams(9)	Единицы измерения подачи подвода: 0 – мм/об 1 – мм/мин
CLD[12]	CLD.CLParams(10)	Величина подачи подвода
CLD[13]	CLD.CLParams(11)	Единицы измерения подачи отвода: 0 – мм/об 1 – мм/мин
CLD[14]	CLD.CLParams(12)	Величина подачи отвода
CLD[17]	CLD.CLParams(15)	Включен ли ориентированный останов шпинделя на дне отверстия: 0 - выключен, 1 - включен.
CLD[18]	CLD.CLParams(16)	Угловое положение шпинделя при ориентированном останове на дне

CLD массив		Описание
		отверстия (в градусах)
CLD[19]	CLD.CLParams(17)	Величина отвода инструмента по координате X после ориентированного останова шпинделя на дне отверстия
CLD[20]	CLD.CLParams(18)	Величина отвода инструмента по координате Y после ориентированного останова шпинделя на дне отверстия
CLD[21]	CLD.CLParams(19)	Величина отвода инструмента по координате Z после ориентированного останова шпинделя на дне отверстия
CLD[50]	CLD.CLParams(48)	Какой шпиндель приводится в движение: 1 - приводной инструмент, 2 - шпиндель заготовки (токарный).



Цикл сверления типа G88 <W5DBore8>

Цикл сверления типа <**W5DBore8(488)**> (G88) предполагает подход инструмента к центру отверстия, расточку на рабочем ходу с выдержкой в течении заданного времени на минимальном уровне и остановкой шпинделя, вывод инструмента вручную на <**уровень отвода**>.



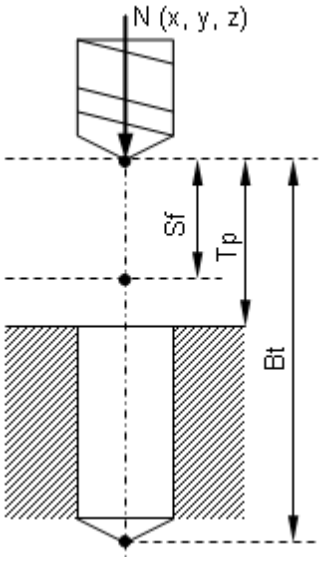
Цикл расточки типа G88 включает в себя следующие шаги.

- Ускоренный подвод инструмента к центру отверстия на уровне <**Z отвода**>.
- Опускание на ускоренном ходу до <**Z безопасной**>.
- Рабочий ход инструмента на расстояние <**Z мин**>.
- Выстой инструмента.
- Останов шпинделя.
- Ручной отвод инструмента до уровня <**Z отвода**>.
- Восстановление первоначальных направления и частоты вращения шпинделя.

Параметры:

CLD массив		Описание
CLD[1]	CLD.SubCmd	Модификатор команды: ON(71) – включение стандартного цикла, CALL(52) – вызов стандартного цикла, OFF(72) – отмена стандартного цикла.
CLD[2]	CLD.SubType	Идентификатор конкретного типа цикла: W5DBore8(488)
CLD[3]	CLD.CLParams(1)	Nx, координата X вектора нормали инструмента
CLD[4]	CLD.CLParams(2)	Ny, координата Y вектора нормали инструмента
CLD[5]	CLD.CLParams(3)	Nz, координата Z вектора нормали инструмента
CLD[6]	CLD.CLParams(4)	Sf, Расстояние по нормали от текущего положения инструмента до уровня безопасной плоскости
CLD[7]	CLD.CLParams(5)	Tp, Расстояние по нормали от текущего положения инструмента до

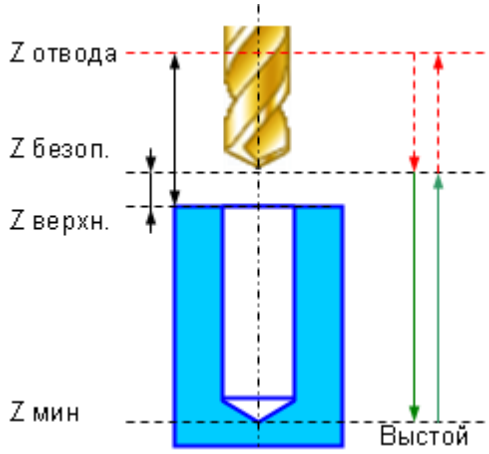
CLD массив		Описание
		верхнего уровня отверстия
CLD[8]	CLD.CLParams(6)	Bt, Расстояние по нормали от текущего положения инструмента до нижнего уровня отверстия
CLD[9]	CLD.CLParams(7)	Единицы измерения рабочей подачи: 0 – мм/об 1 – мм/мин
CLD[10]	CLD.CLParams(8)	Величина рабочей подачи
CLD[11]	CLD.CLParams(9)	Единицы измерения подачи подвода: 0 – мм/об 1 – мм/мин
CLD[12]	CLD.CLParams(10)	Величина подачи подвода
CLD[13]	CLD.CLParams(11)	Единицы измерения подачи отвода: 0 – мм/об 1 – мм/мин
CLD[14]	CLD.CLParams(12)	Величина подачи отвода
CLD[15]	CLD.CLParams(13)	Величина задержки на нижнем уровне отверстия в секундах
CLD[50]	CLD.CLParams(48)	Какой шпиндель приводится в движение: 1 - приводной инструмент, 2 - шпиндель заготовки (токарный).



Цикл сверления типа G89 <W5DBore9>

Цикл сверления типа <**W5DBore9(489)**> (G89) предполагает расточку отверстия с подходом на <**безопасное расстояние**>, выдержкой паузы на дне при включенном шпинделе с последующим выходом на рабочей подаче на <**плоскость отвода**

>.



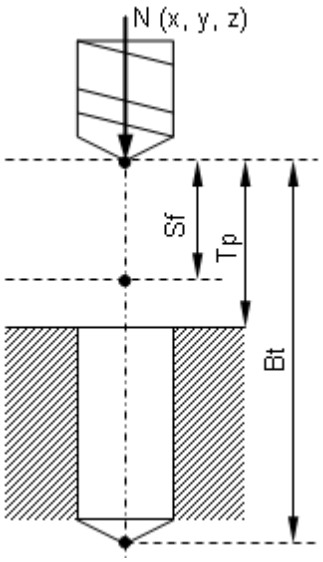
Цикл расточки типа G89 включает в себя следующие шаги:

- Ускоренный подвод инструмента к центру отверстия на уровне <**Z отвода**>.
- Опускание на ускоренном ходу до <**Z безопасной**>.
- Рабочий ход инструмента на расстояние <**Z мин**>.
- Выстой на дне отверстия.
- Возврат инструмента на рабочей подаче до уровня <**Z отвода** >.

Параметры:

CLD массив		Описание
CLD[1]	CLD.SubCmd	Модификатор команды: ON(71) – включение стандартного цикла, CALL(52) – вызов стандартного цикла, OFF(72) – отмена стандартного цикла.
CLD[2]	CLD.SubType	Идентификатор конкретного типа цикла: W5DBore9(489)
CLD[3]	CLD.CLParams(1)	Nx, координата X вектора нормали инструмента
CLD[4]	CLD.CLParams(2)	Ny, координата Y вектора нормали инструмента
CLD[5]	CLD.CLParams(3)	Nz, координата Z вектора нормали инструмента
CLD[6]	CLD.CLParams(4)	Sf, Расстояние по нормали от текущего положения инструмента до уровня безопасной плоскости
CLD[7]	CLD.CLParams(5)	Tp, Расстояние по нормали от текущего положения инструмента до верхнего уровня отверстия

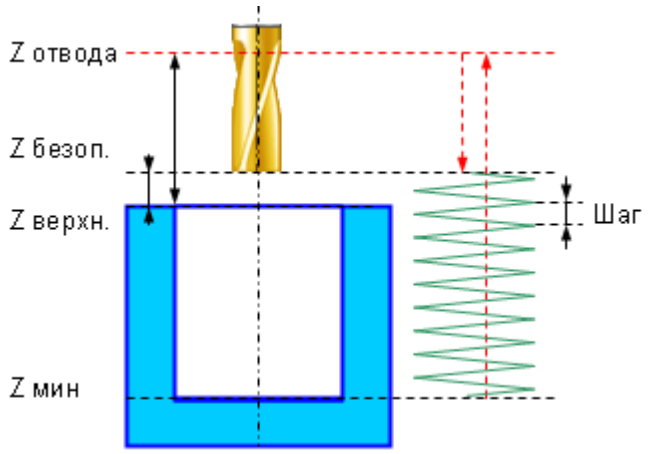
CLD массив		Описание
CLD[8]	CLD.CLParams(6)	Bt, Расстояние по нормали от текущего положения инструмента до нижнего уровня отверстия
CLD[9]	CLD.CLParams(7)	Единицы измерения рабочей подачи: 0 – мм/об 1 – мм/мин
CLD[10]	CLD.CLParams(8)	Величина рабочей подачи
CLD[11]	CLD.CLParams(9)	Единицы измерения подачи подвода: 0 – мм/об 1 – мм/мин
CLD[12]	CLD.CLParams(10)	Величина подачи подвода
CLD[13]	CLD.CLParams(11)	Единицы измерения подачи отвода: 0 – мм/об 1 – мм/мин
CLD[14]	CLD.CLParams(12)	Величина подачи отвода
CLD[15]	CLD.CLParams(13)	Величина задержки на нижнем уровне отверстия в секундах
CLD[50]	CLD.CLParams(48)	Какой шпиндель приводится в движение: 1 - приводной инструмент, 2 - шпиндель заготовки (токарный).



Цикл фрезерования резьбы <W5DThreadMill>

При помощи цикла фрезерования резьбы <**W5DThreadMill(490)**> может быть осуществлено как непосредственно фрезерование резьбы в отверстии (или на бобышке), так и обработка отверстия по спирали. Обработка по спирали бывает полезной в случаях, когда диаметр отверстия превышает диаметр инструмента. Инструмент

совершает вращение вокруг оси отверстия с одновременным перемещением вдоль оси. Диаметр спирали выбирается в соответствии с заданным диаметром отверстия и размером инструмента. Обработка может производиться в несколько проходов до достижения необходимого диаметра отверстия.



Обработка по спирали может включать в себя следующие шаги:

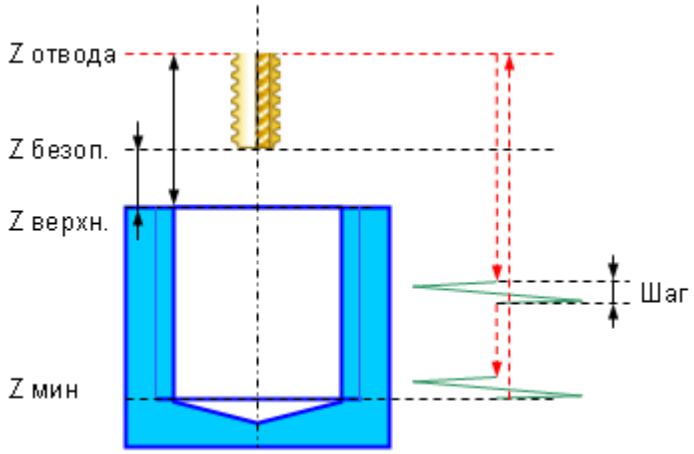
- Ускоренный подвод инструмента к центру отверстия на уровне <**Z отвода**>.
- Опускание на ускоренном ходу до уровня <**Z безопасная**>.
- Подход на рабочей подаче к началу спиралевидного участка.
- Спиралевидный рабочий ход инструмента до глубины <**Z мин**>.
- Опциональный проход по окружности на нижнем уровне. Диаметр окружности равен диаметру спирали.
- Отход к центру отверстия.
- Подъем на ускоренной подаче до уровня <**Z безопасная**>.
- При включении дополнительных черновых и чистовых проходов предыдущие пять шагов могут быть повторены до получения нужного диаметра отверстия.
- Подъем на ускоренной подаче до уровня <**Z отвода**>.

<**Резьбофрезерование**> – современная альтернатива обычному нарезанию резьбы метчиком или плашкой. Оно обладает рядом преимуществ:

- глухие и сквозные, правые и левые резьбы нарезаются одним инструментом;
- различные резьбы с одинаковым шагом могут обрабатываться одним инструментом;
- все точностные параметры могут быть обеспечены одним инструментом;
- получение точной резьбы возможно практически до дна глухого отверстия, так как фреза не имеет заходной фаски;
- обработка различных материалов одним инструментом;

- высокая надежность процесса благодаря хорошему отводу стружки;
- высокая производительность резьбофрезерования за счет высоких скоростей резания и подач;
- низкий крутящий момент на шпинделе станка даже при обработке крупных резьб.

В качестве резьбофрезерного применяются как инструменты с одни зубом, так и инструменты со множеством зубьев, что позволяет за один проход получать сразу несколько витков резьбы. Последовательность шагов при обработке инструментом с одним зубом мало чем отличается от обработки по спирали.



При использовании многозубого инструмента последовательность выполнения цикла резьбофрезерования может быть следующей:

- Ускоренный подвод инструмента к центру отверстия на уровне <**Z отвода**>.
- Опускание на ускоренном ходу до уровня <**Z безопасная**>.
- Опускание на ускоренном ходу на расстояние, равное длине рабочей части инструмента, которая определяется количеством и размером зубьев фрезы (шагом резьбы).
- Подход на рабочей подаче к началу спиралевидного участка.
- Обработка вдоль одного витка спирали с шагом равным шагу резьбы.
- Отход к центру отверстия.
- В случае, если одного витка спирали недостаточно для получения резьбы на всю глубину отверстия, опускание на длину рабочей части инструмента и обработка вдоль витка спирали повторяются до достижения требуемой глубины отверстия.
- Осуществляется подъем на ускоренной подаче до уровня <**Z отвода**>.

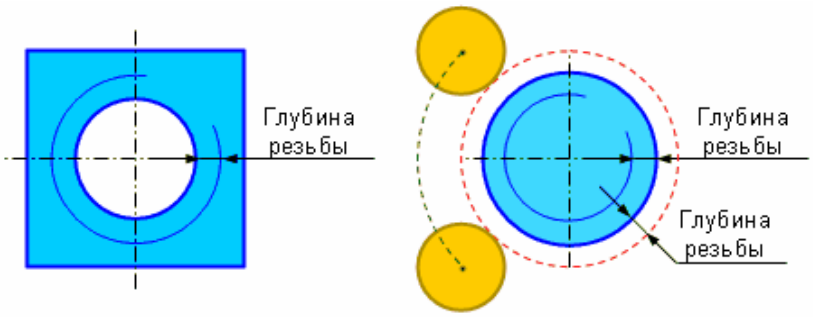
При включении дополнительных черновых и чистовых проходов вся последовательность шагов может повторяться, пока не будет обеспечена заданная глубина резьбы.

Параметр <**Тип резьбы**> определяет отверстие или бобышку следует обрабатывать. Если выбрано значение <**Внутренняя**>, то предполагается, что в отверстии нарезается внутренняя резьба. Если выбрано значение <**Внешняя**>, то предполагается нарезание внешней резьбы на бобышке.

По технологическим причинам в одних случаях резьбофрезерование должно выполняться сверху вниз, в других – снизу вверх вдоль оси отверстия. Данный цикл поддерживает оба способа.

Тип резьбы, правая или левая, определяется параметром <**Направление спирали резьбы**>. Для случая обработки отверстия по спирали удобным может оказаться задание направления спирали в зависимости от направления вращения шпинделя инструмента. Для этого имеются варианты <**Попутное**> и <**Встречное**>. При попутном типе фрезерования направление вращения инструмента и направление спирали совпадают, а при встречном – противоположны друг другу.

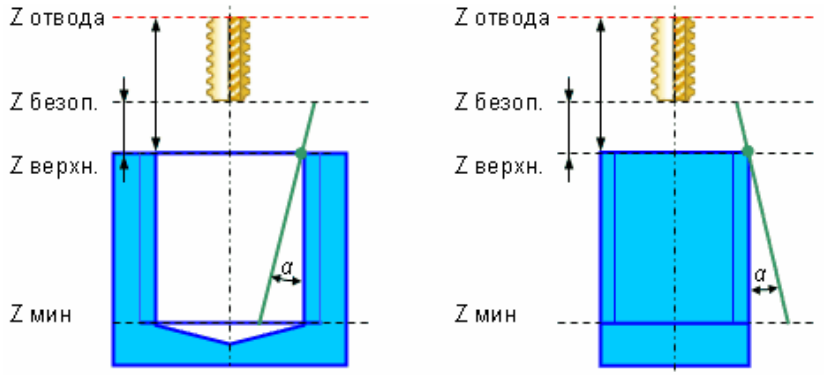
Параметр <**Глубина резьбы**>, позволяет задать соотношение между наружным и внутренним диаметрами резьбы.



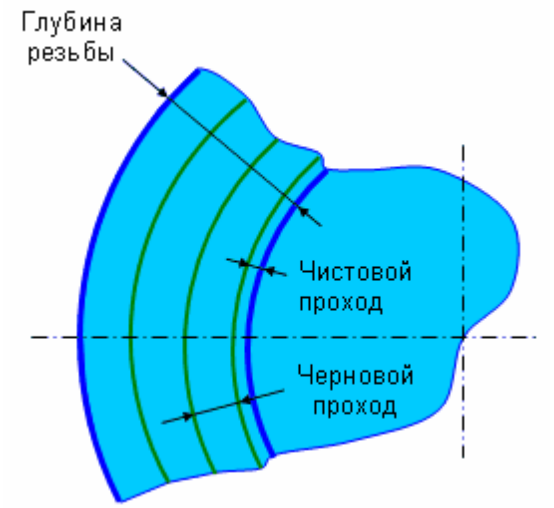
Флаг <**Проход на нижнем уровне**> управляет проходом по окружности, который выполняется при достижении дна отверстия. При включенном параметре, после достижения дна отверстия на спиральном ходе, выполняется проход вдоль полной окружности, диаметр которой равен диаметру спирали. При выключенном параметре проход по окружности не выполняется.

Фрезерование многозаходной резьбы можно быть осуществлено, если в параметре <**Количество заходов**> указано значение большее 1. При значении 1 для данного параметра выполняется цикл фрезерования однозаходной резьбы.

Цикл поддерживает фрезерование конических резьб. Для этого в параметрах задается <**угол конусности**> в градусах. Угол конуса всегда отсчитывается от верхнего уровня отверстия (бобышки). При фрезеровании внутренней резьбы положительным направлением отсчета угла конуса является направление к центру отверстия. При фрезеровании наружной резьбы положительным направлением отсчета угла конуса считается направление от центра бобышки.

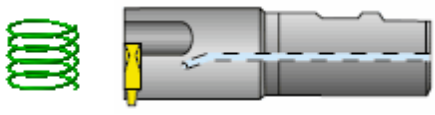


Фрезерование резьбы может производиться в несколько проходов. Для этого в списке имеются следующие параметры: <**Количество черновых ходов**>, <**шаг чернового прохода**>, <**шаг чистового прохода**>.



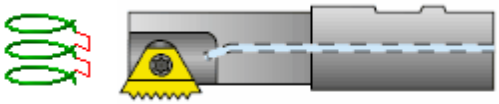
Параметр <**Тип траектории**> предназначен для цикла резьбофрезерования и определяет вид траектории в зависимости от используемого типа инструмента. Он может принимать следующие значения:

- <**Непрерывная**>. Данный тип траектории предназначен для использования с однозубым инструментом, который способен за один виток спирали сформировать только один виток резьбы. Траектория в этом случае выглядит как одна непрерывная спираль.



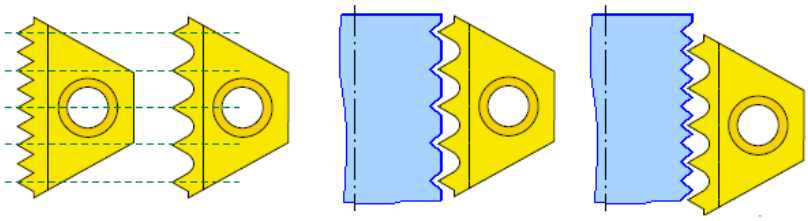
- <**С переходом вдоль оси**>. Данный тип предназначен для использования с многозубым инструментом, который может за один виток спирали сформировать сразу несколько витков

резьбы. Траектория получается в виде последовательности расположенных друг над другом витков спирали, между которыми совершается ускоренный переход вдоль оси на величину, равную длине рабочей части инструмента.



Параметр <**Рабочая длина инструмента**> актуален только для типа траектории <**С переходом вдоль оси**> и определяет длину перехода между двумя соседними витками спирали. Фактически он должен быть равен умноженному на шаг резьбы количеству витков резьбы, которые может сформировать инструмент за один виток спирали.

Вследствие технологической сложности изготовления инструментов для фрезерования мелких резьб иногда для получения резьбы с мелким шагом используют инструменты, имеющие большее чем шаг резьбы расстояние между зубьями. Причем это расстояние всегда должно быть кратно шагу резьбы. В этих случаях для получения резьбы нужного мелкого шага инструменту требуется совершить сразу несколько витков спирали. Например, если расстояние между зубьями инструмента в два раза больше шага резьбы, то он должен описать два витка спирали с шагом резьбы, т.к. за один виток спирали будет получена только половина витков резьбы. Количество витков спирали, которые должен совершить инструмент для получения нужного шага резьбы задается параметром <**Число витков**>. В большинстве случаев данный параметр равен 1, что означает равенство шага резьбы с шагом зубьев инструмента.

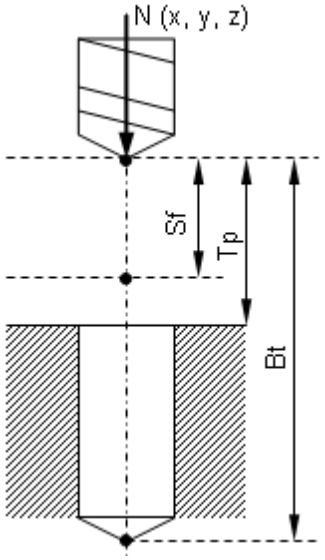


Параметры:

CLD массив		Описание
CLD[1]	CLD.SubCmd	Модификатор команды: ON(71) – включение стандартного цикла, CALL(52) – вызов стандартного цикла, OFF(72) – отмена стандартного цикла.

CLD массив		Описание
CLD[2]	CLD.SubType	Идентификатор конкретного типа цикла: W5DThreadMill(490)
CLD[3]	CLD.CLParams(1)	Nx, координата X вектора нормали инструмента
CLD[4]	CLD.CLParams(2)	Ny, координата Y вектора нормали инструмента
CLD[5]	CLD.CLParams(3)	Nz, координата Z вектора нормали инструмента
CLD[6]	CLD.CLParams(4)	Sf, Расстояние по нормали от текущего положения инструмента до уровня безопасной плоскости
CLD[7]	CLD.CLParams(5)	Tp, Расстояние по нормали от текущего положения инструмента до верхнего уровня отверстия
CLD[8]	CLD.CLParams(6)	Bt, Расстояние по нормали от текущего положения инструмента до нижнего уровня отверстия
CLD[9]	CLD.CLParams(7)	Единицы измерения рабочей подачи: 0 – мм/об 1 – мм/мин
CLD[10]	CLD.CLParams(8)	Величина рабочей подачи
CLD[11]	CLD.CLParams(9)	Единицы измерения подачи подвода: 0 – мм/об 1 – мм/мин
CLD[12]	CLD.CLParams(10)	Величина подачи подвода
CLD[13]	CLD.CLParams(11)	Единицы измерения подачи отвода: 0 – мм/об 1 – мм/мин
CLD[14]	CLD.CLParams(12)	Величина подачи отвода
CLD[16]	CLD.CLParams(14)	D, наружный диаметр резьбы
CLD[17]	CLD.CLParams(15)	St, шаг резьбы
CLD[18]	CLD.CLParams(16)	Тип резьбы: 0 – внутренняя, 1 – наружная.
CLD[19]	CLD.CLParams(17)	Направление спирали резьбы: 0 – встречное, 1 – попутное, 2 – правое, 3 – левое. В случае 0 и 1 направление резьбы зависит от направления вращения инструмента, а в случае 2 и 3 – не зависит.
CLD[20]	CLD.CLParams(18)	Последовательность обработки: 0 – сверху-вниз, 1 – снизу-вверх
CLD[21]	CLD.CLParams(19)	Количество заходов резьбы
CLD[22]	CLD.CLParams(20)	Глубина резьбы (разность между наружным и внутренним радиусами)
CLD[23]	CLD.CLParams(21)	Шаг для обработки черновыми

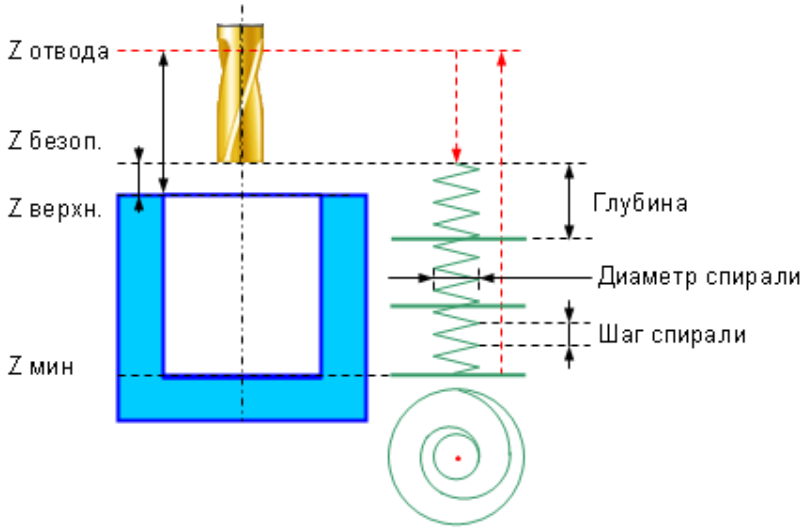
CLD массив		Описание
		ходами
CLD[24]	CLD.CLParams(22)	Количество черновых ходов
CLD[25]	CLD.CLParams(23)	Шаг чистового прохода
CLD[26]	CLD.CLParams(24)	Признак необходимости выполнить проход по окружности на дне отверстия: 0 – не выполнять проход по окружности, 1 – выполнить проход по окружности
CLD[27]	CLD.CLParams(25)	Угол конуса в градусах
CLD[28]	CLD.CLParams(26)	Тип траектории: 0 – непрерывная на всю глубину отверстия, 1 – с переходом вдоль оси (для многозубого инструмента)
CLD[29]	CLD.CLParams(27)	Рабочая длина инструмента (определяется количеством зубьев). Данная величина определяет длину перехода для способа формирования спирали «с переходом».
CLD[30]	CLD.CLParams(28)	Количество оборотов спирали для способа формирования спирали «с переходом» (Обычно равно 1, однако при нарезании резьбы инструментом, шаг которого больше шага резьбы, может быть больше 1).



Цикл выборки отверстия <W5DHolePocketing>

Цикл выборки отверстия типа <**W5DHolePocketing(491)**> предназначен для обработки круглых отверстий, диаметр которых намного больше диаметра инструмента. Выборка материала из

отверстий производится слоями. Инструмент врезается по спирали к каждому слою, а затем расширяет отверстие до требуемого диаметра движением по спирали Архимеда с чистовым проходом фрезы по окружности.



Выборка круглого колодца включает в себя следующие шаги:

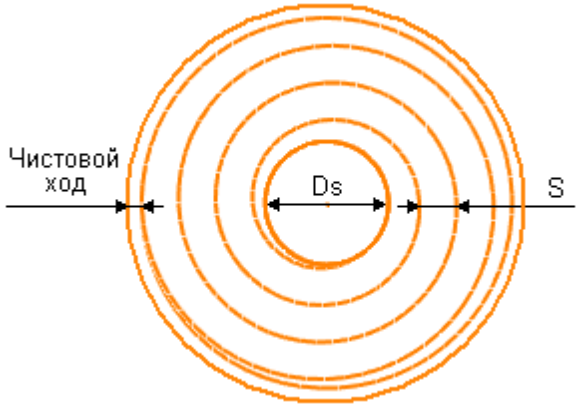
- Ускоренный подвод инструмента к центру отверстия на уровне <**Z отвода**>.
- Опускание на ускоренном ходу до <**Z безопасной**>.
- Врезание по спирали на <**глубину резания Z**>.
- Движение по спирали Архимеда на этом уровне с <**шагом S**> до выхода оси вращения инструмента на окружность с диаметром равным разности между диаметром отверстия и диаметром инструмента.
- Чистовой проход по окружности указанного диаметра без изменения уровня.
- Повторение предыдущих 3-х шагов до достижения требуемой глубины отверстия, с переходом к точке начала следующего врезания без изменения уровня.
- Отход к центру отверстия.
- Подъем на ускоренной подаче до уровня <**Z отвода**>.

Направление закручивания спирали может принимать следующие значения:

- <**Правое**>. Спираль закручена вправо. При взгляде сверху инструмент совершает вращения вокруг оси отверстия по часовой стрелке.
- <**Левое**>. Спираль закручена влево. При взгляде сверху инструмент совершает вращения вокруг оси отверстия против часовой стрелки.
- <**Встречное**>. Направление закручивания спирали зависит от направления вращения шпинделя инструмента и соответствует

встречному типу фрезерования. При встречном типе фрезерования направление вращения инструмента и направление спирали противоположны друг другу.

- <**Попутное**>. Направление закручивания спирали зависит от направления вращения шпинделя инструмента и соответствует попутному типу фрезерования. При попутном типе фрезерования направление вращения инструмента и направление спирали совпадают.

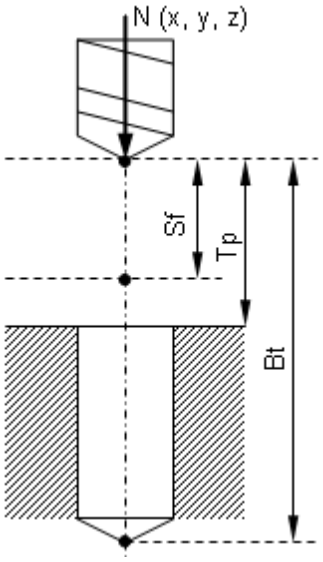


Если величина чистового прохода не равна нулю, то перед выполнением окончательного прохода по окружности на каждом слое будет произведен дополнительный проход по окружности с данным припуском. Это позволяет обеспечить равномерность толщины снимаемого слоя материала на окончательном проходе.

Параметры:

CLD массив		Описание
CLD[1]	CLD.SubCmd	Модификатор команды: ON(71) – включение стандартного цикла, CALL(52) – вызов стандартного цикла, OFF(72) – отмена стандартного цикла.
CLD[2]	CLD.SubType	Идентификатор конкретного типа цикла: W5DHolePocketing(491)
CLD[3]	CLD.CLParams(1)	Nx, координата X вектора нормали инструмента
CLD[4]	CLD.CLParams(2)	Ny, координата Y вектора нормали инструмента
CLD[5]	CLD.CLParams(3)	Nz, координата Z вектора нормали инструмента
CLD[6]	CLD.CLParams(4)	Sf, Расстояние по нормали от текущего положения инструмента до уровня безопасной плоскости
CLD[7]	CLD.CLParams(5)	Tp, Расстояние по нормали от текущего положения инструмента до верхнего уровня отверстия

CLD массив		Описание
CLD[8]	CLD.CLParams(6)	Bt, Расстояние по нормали от текущего положения инструмента до нижнего уровня отверстия
CLD[9]	CLD.CLParams(7)	Единицы измерения рабочей подачи: 0 – мм/об 1 – мм/мин
CLD[10]	CLD.CLParams(8)	Величина рабочей подачи
CLD[11]	CLD.CLParams(9)	Единицы измерения подачи подвода: 0 – мм/об 1 – мм/мин
CLD[12]	CLD.CLParams(10)	Величина подачи подвода
CLD[13]	CLD.CLParams(11)	Единицы измерения подачи отвода: 0 – мм/об 1 – мм/мин
CLD[14]	CLD.CLParams(12)	Величина подачи отвода
CLD[16]	CLD.CLParams(14)	Диаметр отверстия
CLD[17]	CLD.CLParams(15)	Шаг спирали
CLD[18]	CLD.CLParams(16)	Диаметр спирали (Ds)
CLD[19]	CLD.CLParams(17)	Направление спирали: 0 – встречное, 1 – попутное, 2 – правое, 3 – левое. В случае 0 и 1 направление спирали зависит от направления вращения инструмента, а в случае 2 и 3 – не зависит.
CLD[20]	CLD.CLParams(18)	Глубина резания по нормали
CLD[21]	CLD.CLParams(19)	Количество резов по нормали
CLD[22]	CLD.CLParams(20)	Шаг спирали Архимеда (S)
CLD[23]	CLD.CLParams(21)	Величина чистового прохода



5.1.10 Подача <FEDRAT>

Команда <**FEDRAT**> определяет величину и тип подачи инструмента. Величина подачи сохраняется до следующего оператора <**FEDRAT**>, или оператора перемещения с ускоренной подачей <**RAPID**>.

В электроэрозионных операция SprutCAM команда <**FEDRAT**> используется для определения кода условий обработки (ConditionCode). Обычно код условий обработки в электроэрозионных станках определяет значительно больше параметров обработки чем просто величину подачи проволоки. С ним, например, могут связывать различные мощностные характеристики: частоту, силу тока, режимы работы генератора и т. п. По заданному коду условий обработки стойка ЧПУ автоматически выбирает из имеющейся у нее специальной таблицы все необходимые параметры обработки.

Команда:

```
FEDRAT NM nm, K k, MMPM(315) | MMPR(316)
```

Параметры:

Параметр	CLD массив		Описание
nm	CLD[1]	CLD.NM	Величина подачи
k	CLD[2]	CLD.K	Тип подачи (рабочая, подача подхода, отхода, перехода, врезания и т.п.). В электроэрозионных операциях SprutCAM при помощи данного параметра передается код условий обработки.
MMPM(315), MMPR(316)	CLD[3]	CLD.MMPM	Единицы измерения подачи: MMPM(315) – мм/мин, MMPR(316) – мм/об

5.1.11 Конечная запись <FINI>

Команда <**FINI**> всегда является последней технологической командой во всей последовательности технологических команд проекта. Таким образом, она обрабатывается в последнюю очередь, и ее следует использовать для формирования завершающих кадров управляющей программы.

Команда:

```
FINI
```

5.1.12 Начальная точка <FROM>

Задаются координаты исходной точки. Команда обычно встречается один раз и стоит в начале описания обработки детали. Обычно координаты первой команды перемещения совпадают с координатами, переданными в команде <**From**>. В некоторых случаях требуется избежать вывода в управляющую программу координат начальной точки. В таких случаях можно проинициализировать регистры, соответствующие координатам X, Y и Z в обработке команды <**From**> без вывода кадра в управляющую программу. При появлении первой команды перемещения те координаты, значения которых совпадают с переданными в команде <**From**>, не попадут в кадр управляющей программы.

Команда:

```
FROM X x, Y y, Z z
```

Параметры:

Параметр	CLD массив		Описание
x, y, z	CLD[1] CLD[2] CLD[3]	CLD.X CLD.Y CLD.Z	координаты начальной точки

5.1.13 Возврат в нулевую точку станка <GOHOME>

По этой команде инструмент перемещается в нулевую точку станка. Чаще всего данной команде в управляющей программе соответствует G-код G28. Координаты нулевой точки станка определяются в стойке ЧПУ, а в постпроцессоре передается только код команды. Перемещение в нулевую точку станка обычно используется для отвода инструмента в безопасную зону перед сменой заготовки или инструмента.

Команда:

```
GOHOME
```

5.1.14 Линейные перемещения <GOTO.abs>

Постпроцессор передает данные о линейных перемещениях посредством команды <**ABSMOV**> (<**GOTO.abs**>). В параметрах команды содержатся новые абсолютные координаты инструмента по осям X, Y и Z.

Команда:

```
GOTO.abs X x, Y y, Z z
```

Параметры:

Параметр	CLD массив		Описание
x, y, z	CLD[1] CLD[2] CLD[3]	CLD.X CLD.Y CLD.Z	Новые координаты инструмента (абсолютные)

5.1.15 Номер шпиндельной головки <HEAD>

Команда <**HEAD**> переключает активную шпиндельную головку. Номер активируемой головки передается первым параметром команды.

Команда:

```
HEAD BOTH(83) | N n
```

Параметры:

Параметр	CLD массив		Описание
BOTH(83), n	CLD[1]	CLD.N	83 (BOTH) – одновременная работа двух головок или n – номер требуемой головки.

5.1.16 Непосредственный вывод в кадр <INSERT>

Команда <**INSERT**> предназначена для передачи текстовой информации напрямую в управляющую программу (УП). Текст из команды попадает в переменную <**CLDATA$**>, после чего может быть выдан в УП.

Команда:

```
INSERT "..."
```

5.1.17 Режим интерполяции <INTERPOLATION>

Команда <**INTERPOLATION**> активирует либо деактивирует специфические режимы интерполяции, такие как цилиндрическая, полярная многоосевая. Если параметр <**CLD[1]**> (<**CLD.SubCmd**>) равен <**ON(71)**>, то производится включение режима интерполяции, указанного в параметре <**CLD[2]**> (<**CLD.Mode**>). Если же параметр <**CLD[1]**> (<**CLD.SubCmd**>) равен <**OFF(72)**>, то производится выключение соответствующего режима.

Дополнительные параметры <**CLD[3]**> (<**CLD.P1**>), <**CLD[4]**> (<**CLD.P2**>), <**CLD[5]**> (<**CLD.P3**>) предназначены для передачи специфичных для каждого режима интерполяции параметров.

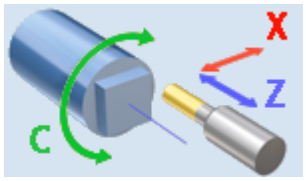
Команда:

```
INTERPOLATION ON(71) | OFF(72), CARTESIAN(9020) | POLAR(9021) |
              CYLINDRICAL(9022) | MULTIAXIS(9023),
              P1 p1, P2 p2, P3 p3
```

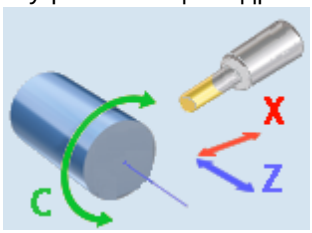
Параметры:

Параметр	CLD массив		Описание
ON(71), OFF(72)	CLD[1]	CLD.SubCmd	ON(71) – команда включения, OFF(72) – команда выключения.
CARTESIAN(9020), POLAR(9021), CYLINDRICAL(9022), MULTIAXIS(9023)	CLD[2]	CLD.Mode	Тип переключаемого режима интерполяции: CARTESIAN(9020) – декартов режим интерполяции, POLAR(9021) – режим полярной интерполяции, CYLINDRICAL(9022) – режим цилиндрической интерполяции, MULTIAXIS(9023) – режим многоосевой интерполяции
P1	CLD[3]	CLD.P1	Для цилиндрической интерполяции данный параметр определяет радиус цилиндра
P2	CLD[4]	CLD.P2	Зарезервировано
P3	CLD[5]	CLD.P3	Зарезервировано

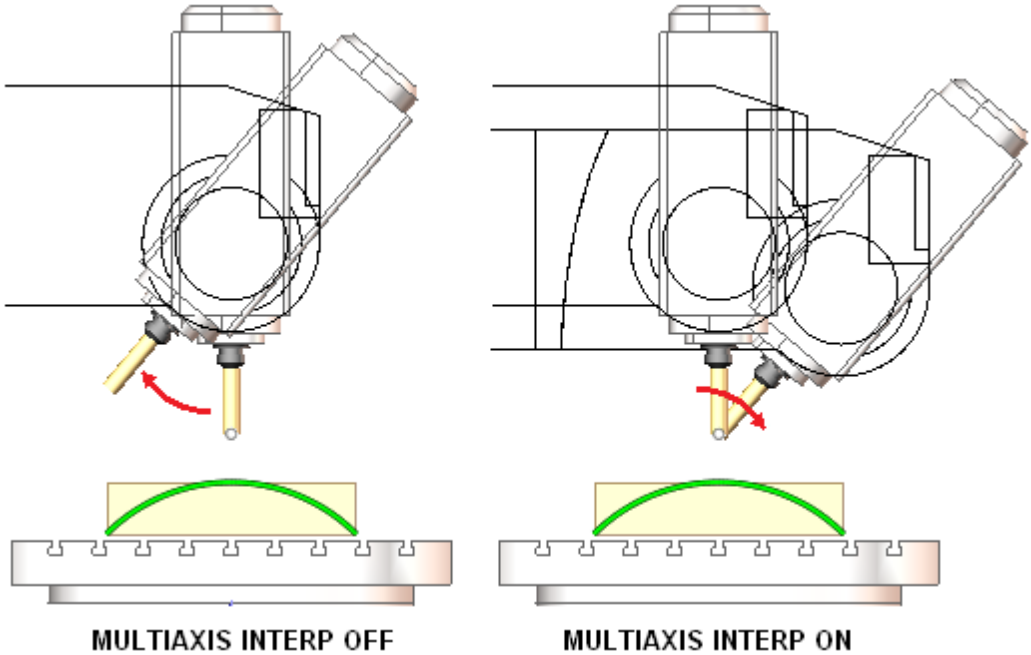
Режим <**полярной интерполяции**> характеризуется тем, что инструмент расположен параллельно оси вращения заготовки и перемещение по одной из запрограммированных линейных осей, перпендикулярных оси вращения, компенсируется стойкой ЧПУ перемещением по поворотной оси и второй линейной оси. Он чаще всего используется для обработки торцевых поверхностей на станках с отсутствующей линейной осью.



Режим <**цилиндрической интерполяции**> характеризуется заменой запрограммированного перемещения по линейной оси, перпендикулярной оси вращения заготовки, перемещением поворотной оси. Инструмент при этом расположен перпендикулярно оси вращения. Таким образом, стойка ЧПУ позволяет программированием виртуального плоского контура осуществлять обработку реальной цилиндрической поверхности.



Режим <**многоосевой интерполяции**> реализуется многими современными стойками ЧПУ, используемыми для непрерывной многокоординатной обработки. Суть его состоит в том, чтобы абстрагироваться от реальной кинематики станка, и сделать управляющую программу максимально независимой от нее, а также от погрешностей расположения заготовки относительно центров вращения поворотных осей. При выключенном режиме <**многоосевой интерполяции**> каждое запрограммированное перемещение поворотной оси осуществляется вокруг собственного центра вращения и независимо от перемещения линейных осей. В случае когда режим включен любое перемещение поворотных осей вызывает автоматически рассчитываемые стойкой ЧПУ компенсирующие перемещения линейных осей таким образом, чтобы положение кончика инструмента относительно заготовки оставалось неизменным.



В различных стойках ЧПУ команды на включение данного режима отличаются. Например, в Heidenhain подобный режим именуется "Tool center point management" и переключается командами TCPM, M128, M129. В стойках Sinumerik режим включается и выключается командами TRAORI, TRAFOOF.

5.1.18 Загрузка инструмента <LOADTL>

Команда <**LOADTL**> загружает инструмент с номером, передаваемым в качестве первого параметра. При обработке данной команды следует формировать кадры управляющей программы, которые перемещают револьверную головку либо выбирают позицию в магазине инструментов и активируют смену инструмента. Также в данной команде можно формировать кадры отвода инструмента в безопасную позицию для смены.

Команда:

```
LOADTL N n, X x, Y y, Z z, D d, M m, K k, L l, P p, A a, R r,H h,RC rc,
       PLANE XY(33) | YZ(37) | XZ(41) | YX(133) | ZY(137) | ZX(141),
       FEEDCOLOR c1, FEEDCOLOR c1, DUR d2
```

Параметры:

Параметр	CLD массив		Описание
n	CLD[1]	CLD.N	Идентификатор (номер) инструмента
x, y, z	CLD[2], CLD[3], CLD[4]	CLD.X CLD.Y CLD.Z	Смещение инструмента по осям LX, LY, LZ
d	CLD[5]	CLD.D	Диаметр инструмента
m	CLD[6]	CLD.M	Номер корректора на длину. Если настроечная точка – кончик инструмента, то M<0.
k	CLD[7]	CLD.K	Номер корректора на радиус. Если настроечная точка – центр инструмента, то K<0.
L	CLD[8]	CLD.L	Длина (вылет) инструмента
P	CLD[9]	CLD.P	Ширина инструмента
a	CLD[10]	CLD.A	Угол врезания инструмента (при внешней выборке 180<a<0, при внутренней выборке 0<a<90)
r	CLD[11]	CLD.R	Радиус скругления инструмента
h	CLD[12]	CLD.H	Высота фасонной части осевого инструмента
rc	CLD[13]	CLD.Rc	Радиус у цилиндрической части осевого инструмента.
XY, YZ, XZ, YX, ZY, ZX	CLD[14]		Плоскость инструмента. Для осевого инструмента это плоскость, которой перпендикулярна ось инструмента. Для токарного инструмента это плоскость, в которой производится обработка данным инструментом. 33 – XY, 37 – YZ, 41 – XZ, 133 – YX, 137 – ZY, 141 – ZX.
	CLD[15],		Зарезервировано

	CLD[16]		
d2	CLD[17]	CLD.Dur	Стойкость инструмента в минутах. 0 – не учитывать
HID	CLD[18]	CLD.HID	Идентификатор (номер) держателя инструмента.

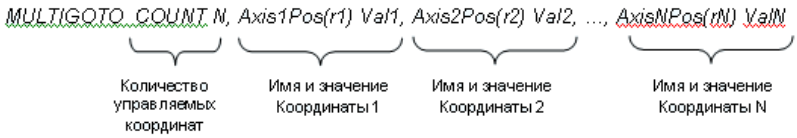
Параметры, доступные через оператор Cmd

TCLDLoadTool: ComplexType	Команда загрузки инструмента	
	ToolID: Integer	Cmd.Int["ToolID"] - Идентификатор (номер) инструмента
	RadCorrNum: Integer	Cmd.Int["RadCorrNum"] - Номер корректора на радиус инструмента
	LenCorrNum: Integer	Cmd.Int["LenCorrNum"] - Номер корректора на длину инструмента
	HolderID: Integer	Cmd.Int["HolderID"] - Идентификатор (номер) держателя инструмента
	RevolverID: String	Cmd.Str["RevolverID"] - Строковый идентификатор револьверной головки

5.1.19 Многокоординатные перемещения <MULTIGOTO>

<**MULTIGOTO**> представляет собой обобщенную технологическую команду, предназначенную для перемещений управляемых координат станка. Команда объединяет в себе возможности команд линейного перемещения (<**GOTO**>) и вращения поворотных осей (<**ROTABL**>), а также дает возможность управлять координатами, специфичными для конкретного станка (магазином инструментов, револьверной головкой, патроном и т.п.).

Команда:



Параметры:

Параметр	CLD массив	Описание
N	CLD[1]	Количество управляемых координат станка, присутствующих в данной команде.
r1	CLD[2]	Номер регистра, соответствующего управляемой координате станка с именем Axis1Pos
Val1	CLD[3]	Значение управляемой координаты

Параметр	CLD массив	Описание
		станка с именем Axis1Pos.
r2	CLD[4]	Номер регистра, соответствующего управляемой координате станка с именем Axis2Pos
Val2	CLD[5]	Значение управляемой координаты станка с именем Axis2Pos.
…	…	…
rN	CLD[2*N]	Номер регистра, соответствующего управляемой координате станка с именем AxisNPos
ValN	CLD[2*N+1]	Значение управляемой координаты станка с именем AxisNPos.

Параметры, доступные через оператор Cmd

TCLDMultiGoto: ComplexType	Команда перемещения по осям станка			
	Axes: Array, Key="AxisID"	Cmd.Ptr["Axes"] - Массив структур типа Axis. Одна команда может содержать перемещения сразу по нескольким осям, положение каждой из которых хранится в данном массиве.		
		Axis: ComplexType	Cmd.Ptr["Axes"].Item[Index] или Cmd.Ptr["Axes(<AxisName>)"] - Отдельный элемент массива Axes. Содержит информацию о перемещении по одной оси станка. Доступ к элементам массива возможен либо по индексу, либо по ключевому полю. Здесь <AxisName> - значение ключевого поля, которое должно совпадать со значением поля AxisID.	
			AxisID: String	Cmd.Str["Axes(<AxisName>).AxisID"] - Идентификатор управляемой координаты (оси) станка, для которой задается новое положение. Определяется схемой станка.
			Value: Double	Cmd.Flt["Axes(<AxisName>).Value"] - Новое положение оси станка, в которое она перемещается.

Список управляемых координат передаваемых в CLData определяется кинематической схемой станка SprutCAM.

Для более удобной работы с командой многокоординатного перемещения <**MULTIGOTO**> вместо массива <**CLD**> рациональнее использовать специально созданный для этой команды предопределенный массив <**GMA**>, либо обращаться к параметрам через оператор **Cmd**.

Пример простой программы обработки команды <**MULTIGOTO**> с использованием оператора **Cmd** приведен ниже.

```
program MultiGoto
  if cmd.Ptr["Axes(AxisXPos)"]>0 then begin
    X = cmd.Flt["Axes(AxisXPos).Value"]
```

```
  end
  if cmd.Ptr["Axes(AxisYPos)"]>0 then begin
    Y = cmd.Flt["Axes(AxisYPos).Value"]
  end
  if cmd.Ptr["Axes(AxisZPos)"]>0 then begin
    Z = cmd.Flt["Axes(AxisZPos).Value"]
  end
  if cmd.Ptr["Axes(AxisCPos)"]>0 then begin
    C = cmd.Flt["Axes(AxisCPos).Value"]
  end
  OutBlock
end
```

5.1.20 Условный пропуск <OPSKIP>

Команда <**OPSKIP**> активирует и деактивирует режим формирования кадров управляющей программы с признаком условного пропуска. Обычно в качестве признака условного пропуска кадра является наличие символа <**/**> в самом его начале. Такие кадры будут обрабатываться стойкой ЧПУ, только если в ней выключен режим пропуска кадров, иначе они будут игнорироваться.

Команда:

```
OPSKIP ON(71)|OFF(72)
```

Параметры:

Параметр	CLD массив		Описание
ON(71), OFF(72)	CLD[1]	CLD.onoff	71 (ON) – включение условного пропуска, 72 (OFF) – отключение условного пропуска.

5.1.21 Дополнительный останов <OPSTOP>

Команда <**OPSTOP**> предназначена для формирования кадров опционального прерывания выполнения управляющей программы. Обычно ей соответствует код M01 управляющей программы. При достижении кадра опционального останова исполнение программы прерывается только в том случае, если в стойке ЧПУ включен соответствующий режим. При выключении данного режима исполнение программы не прерывается.

Команда:

```
OPSTOP
```

5.1.22 Определение системы координат <ORIGIN>

Команда <**ORIGIN**> определяет систему координат, в которой производится дальнейшая обработка. Она либо выбирает одну из стандартных систем координат заготовки (G54 – G59 или другие), либо определяет смещения и углы поворота локальной системы координат относительно ранее установленной системы координат заготовки. В последнем случае, в зависимости от конкретной стойки ЧПУ, нужно формировать команды TRANS, ROT (Sinumeric), CYCLE7, CYCLE19, PLANE (Heidenhain), G52, G68 (Fanuc) и т.п.

Если параметр <**CLD[4]**> (<**CLD.PPFun**>, <**Cmd.Int["OriginType"]** >) равен нулю, то команда производит установку стандартной системы координат заготовки. Номер стандартной системы координат содержится в параметре <**CLD[5]**> (<**CLD.N**>, <**Cmd.Int ["CSNumber"]**>).

Если параметр <**CLD[4]**> (<**CLD.PPFun**>, <**Cmd.Int["OriginType"] =1**>) равен <**PPFun(1079)**>, то команда задает смещения локальной системы координат относительно стандартной системы координат заготовки по осям X, Y и Z, а также углы поворота вокруг этих осей.

Команда:

```
ORIGIN X x, Y y, Z z, PPFUN f, N n, A a, B b, C c
```

Параметры:

Параметр	CLD массив		Описание
x, y, z	CLD[1] CLD[2] CLD[3]	CLD.X CLD.Y CLD.Z	Смещение локальной системы координат по осям X, Y и Z соответственно относительно неподвижной системы координат заготовки .
f	CLD[4]	CLD.PPFun	Режим установки системы координат: 0 – выбор одной из стандартных систем координат заготовки, PPFun(1079) – преобразование локальной системы координат.
n	CLD[5]	CLD.N	Номер активируемой стандартной системы координат: 54 (для G54), 55 (для G55), 59 (для G59) и т.д.
a	CLD[6]	CLD.A	Углы поворота локальной системы координат вокруг осей X, Y и Z соответственно относительно неподвижной системы координат заготовки
b	CLD[7]	CLD.B	
c	CLD[8]	CLD.C	

Параметр	CLD массив		Описание
			.

Параметры, доступные через оператор Cmd

TCLDOrigin: ComplexType	Команда задания системы координат			
	OriginType: Integer	Cmd.Int["OriginType"] - Тип команды смены системы координат: 0 (SelectStandardLCS) - команда выбора стандартной системы координат заготовки (G54-G59), 1 (TransformLCS) - команда преобразования локальной системы координат.		
	CSNumber: Integer	Cmd.Int["CSNumber"] - Номер стандартной системы координат заготовки: 54 (для G54), 55 (для G55), 59 (для G59) и т.д.		
	MCS: ComplexType	Cmd.Ptr["MCS"] - Параметры определяющие преобразование новой локальной системы координат относительно неподвижной системы координат заготовки. Содержит геометрические (пространственные) координаты.		
		OriginPoint: ComplexType	Cmd.Ptr["MCS.OriginPoint"] - Смещение начальной точки системы координат	
			X: Double	Cmd.Flt["MCS.OriginPoint.X"] - смещение по оси X
			Y: Double	Cmd.Flt["MCS.OriginPoint.Y"] - смещение по оси Y
			Z: Double	Cmd.Flt["MCS.OriginPoint.Z"] - смещение по оси Z
		RotAngles: ComplexType	Cmd.Ptr["MCS.RotAngles"] - Углы поворота системы координат вдоль осей (в градусах)	
			A: Double	Cmd.Flt["MCS.RotAngles.A"] - угол поворота вокруг оси X
			B: Double	Cmd.Flt["MCS.RotAngles.B"] - угол поворота вокруг оси Y
			C: Double	Cmd.Flt["MCS.RotAngles.C"] - угол поворота вокруг оси Z
	WCS: ComplexType	Cmd.Ptr["WCS"] - Параметры определяющие преобразование новой локальной системы координат относительно подвижной системы координат заготовки. Содержит геометрические (пространственные) координаты.		
		OriginPoint: ComplexType	Cmd.Ptr["WCS.OriginPoint"] - Смещение начальной точки системы координат	
			X: Double	Cmd.Flt["WCS.OriginPoint.X"] - смещение по оси X
			Y: Double	Cmd.Flt["WCS.OriginPoint.Y"] - смещение по оси Y
			Z:	Cmd.Flt["WCS.OriginPoint.Z"] - смещение

			Double	по оси Z
		RotAngles: ComplexType	Cmd.Ptr["WCS.RotAngles"] - Углы поворота системы координат вдоль осей (в градусах)	
			A: Double	Cmd.Flt["WCS.RotAngles.A"] - угол поворота вокруг оси X
			B: Double	Cmd.Flt["WCS.RotAngles.B"] - угол поворота вокруг оси Y
			C: Double	Cmd.Flt["WCS.RotAngles.C"] - угол поворота вокруг оси Z
	Axes: Array, Key="AxisID"	Cmd.Ptr["Axes"] - Массив структур типа Axis. Задает положения реальных (машинных) осей станка при которых взаимное положение заготовки и инструмента будет соответствовать данной локальной системе координат. Одна команда может содержать положение сразу для нескольких осей.		
		Axis: ComplexType	Cmd.Ptr["Axes"].Item[Index] или Cmd.Ptr["Axes (<AxisName>)"] - Отдельный элемент массива Axes. Содержит информацию о положении одной оси станка. Доступ к элементам массива возможен либо по индексу, либо по ключевому полю. Здесь <AxisName> - значение ключевого поля, которое должно совпадать со значением поля AxisID.	
			AxisID: String	Cmd.Str["Axes(<AxisName>).AxisID"] - Идентификатор управляемой координаты (оси) станка, для которой задается положение. Определяется схемой станка.
			Value: Double	Cmd.Flt["Axes(<AxisName>).Value"] - Значение оси станка.
	IsSpatial: Integer	Cmd.Int["IsSpatial"] - параметр, который определяет какие из координат геометрические или машинные имеют приоритет. Он может принимать два значения: 0 - приоритетом обладают машинные координаты, задаваемые через массив **Axes**, 1 - преимущество имеют геометрические координаты, хранящиеся в структурах **MCS** и **WCS**.		
	PositioningMode: Integer	Cmd.Int["PositioningMode"] - параметр, который определяет вызывает ли данная команда реальные перемещения осей станка или только задает новую локальную систему координат без перемещения осей. Он может принимать 3 значения: 0 - STAY преобразование локальной системы координат не приводит к перемещению осей станка, 1 - TURN преобразование локальной системы координат перемещает только поворотные оси станка, 2 - MOVE преобразование локальной системы координат перемещает все необходимые оси станка таким образом, чтобы сохранилось расположение кончика инструмента относительно заготовки.		

В зависимости от конкретной стойки ЧПУ и от различных ее

параметров для установки повернутой локальной системы координат в форматах команд могут использоваться либо пространственные (геометрические) углы, либо реальные значения поворотных осей станка. Поэтому в параметрах команды ORIGIN значения углов продублированы - в комплексных параметрах **Cmd.Ptr["MCS"]** и **Cmd.Ptr["WCS"]** указываются геометрические углы, а в массиве **Cmd.Ptr["Axes"]** приводятся соответствующие значения реальных осей станка. В некоторых случаях положение локальной системы координат, заданной пространственными углами может быть осуществлено несколькими позициями реальных поворотных осей станка. Некоторые стойки ЧПУ позволяют явно указывать какое решение следует выбирать. В этом случае выбор возможного решения в постпроцессоре может быть осован на анализе реальных значений поворотных осей в массиве **Cmd.Ptr["Axes"]**.

Примечание: *пространственные углы поворота в параметрах команды всегда задаются в той системе координат, которая была до начала преобразований.*

Если формат стойки позволяет использовать разные типы углов, то у пользователя существует возможность явно указать в настройках CAM-системы пространственные углы или значения реальных осей станка предпочтительней использовать. В постпроцессор данная информация приходит через параметр **Cmd.Int["IsSpatial"]**. Если данный параметр равен нулю, то приоритетом пользуются значения реальных осей станка. Если же параметр равен 1, то предпочтение следует отдать пространственным углам.

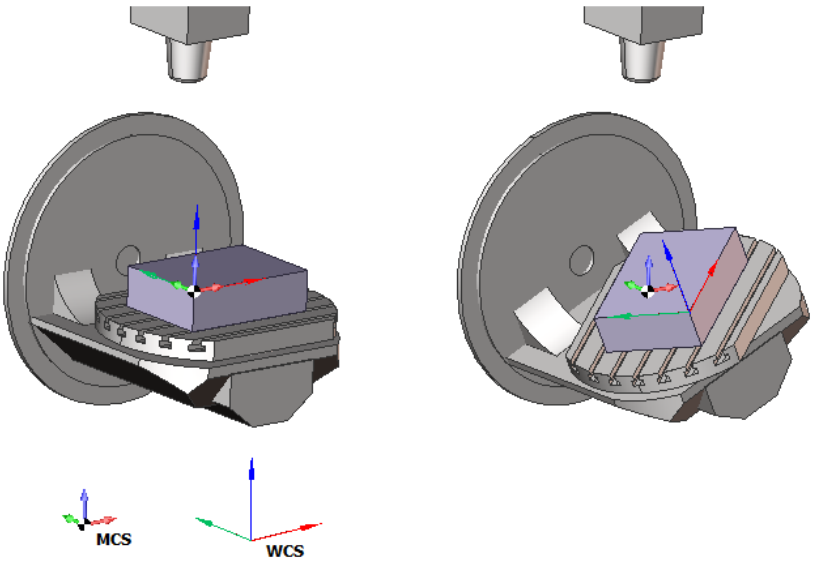
Кроме того, в некоторых стойках ЧПУ команда на включение локальной системы координат в определенных случаях может вызывать перемещения осей станка. Поэтому в команде ORIGIN в параметре **Cmd.Int["PositioningMode"]** дополнительно указывается информация о поведении реальных осей станка в момент включения локальной системы координат. Если параметр равен 0 (STAY), то команда не приводит к перемещению осей станка, а только устанавливает локальную систему координат. Оси станка должны быть при необходимости спозиционированы отдельно. Если данный параметр принимает значение 1 (TURN), то установка локальной системы координат сопровождается перемещением только поворотных осей станка. И, наконец, при параметре равном 2 (MOVE) в момент установки локальной системы координат происходит перемещение всех необходимых осей станка таким образом, чтобы взаимное расположение кончика инструмента и заготовки остались неизменными. Значение параметра **Cmd.Int["PositioningMode"]**, которое приходит в постпроцессор, определяется в настройках CAM-системы. Они могут быть изменены пользователем чтобы сделать поведение системы максимально приближенным к поведению конкретного станка.

Различные системы ЧПУ в зависимости от своей функциональности и от своих настроек могут вести себя по разному при изменении положения поворотных осей станка, задающих ориентацию заготовки. В одних случаях система координат, в которой производится обработка, может оставаться неподвижной при поворотах стола, а в других случаях может перемещаться совместно с заготовкой. С учетом данной особенности параметры локальной системы координат в команде ORIGIN приводятся для обоих описанных случаев: комплексный

параметр **Cmd.Ptr["MCS"]** содержит значения смещений и поворотов относительно неподвижной системы координат заготовки , а комплексный параметр **Cmd.Ptr["WCS"]** - относительно подвижной системы координат заготовки.

Под <**неподвижной системой координат заготовки (MCS)**> понимается система координат, положение которой не изменяется при изменении углов поворота поворотного стола станка. Так обычно работают станки, на которых стойка ЧПУ не поддерживает специальный режим многокоординатной обработки. В данном случае управляющая программа получается сильно зависимой от кинематики станка и от расположения заготовки относительно центров вращения поворотных осей.

Под <**подвижной системой координат заготовки (WCS)**> понимается система координат, которая перемещается вместе с заготовкой при изменении углов поворота поворотного стола. В такой системе координат могут работать станки, обладающие современными стойками ЧПУ со включенной опцией многоосевой обработки. В данном случае управляющая программа получается независимой от кинематики станка, неточности позиционирования заготовки компенсируются стойкой.



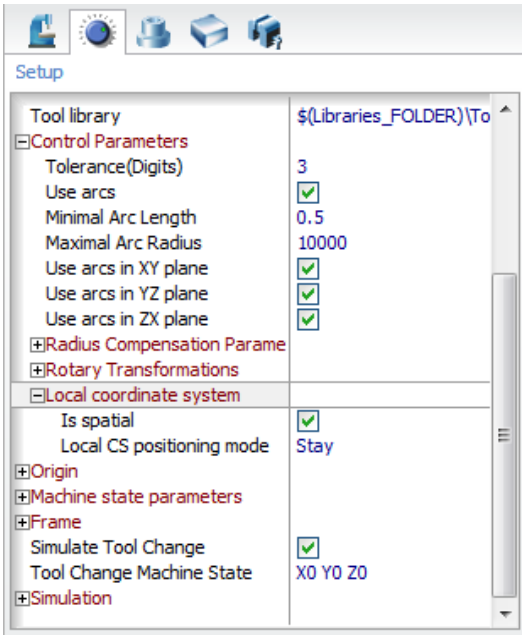
Настройки CAM-системы при работе с локальными системами координат.

В CAM-системе поведение при использовании локальных систем координат может задаваться для каждого станка индивидуально. Для этого в xml-файле станка следует выставить соответствующие опции, расположенные в секции <**ControlData.LocalCS**>. Если в конкретном файле станка данные опции не заданы, то они наследуются из значений по умолчанию для абстрактного станка системы.

```
<SCType ID="MachineName" type="AbstractMachine
```

```
                ...
                <ControlData>
                        ...
                        <LocalCS>
                                <IsSpatial DefaultValue="Tru
                                <PositioningMode DefaultValu
                        </LocalCS>
                        ...
                </ControlData>
                ...
        <SCType>
```

Указанные параметры станка могут быть отредактированы пользователем в конкретном проекте по своему усмотрению. Они доступны в инспекторе свойств станка.



Примеры обработчиков команды <**ORIGIN**>:

```
program Origin
  if Cmd.Int["OriginType"]=0 then begin !G54-G59
    Output "G" + Cmd.Str["CSNumber"]
  end else begin                        !G52 X Y Z
    X = Cmd.Flt["MCS.OriginPoint.X"]
    Y = Cmd.Flt["MCS.OriginPoint.Y"]
    Z = Cmd.Flt["MCS.OriginPoint.Z"]
    Output "G52 X" + Str(X) + " Y" + Str(Y) + " Z" + Str(Z)
  end
end

program Origin
  if CLD[4]=0 then begin ! Select standart LCS (G54-G59)
```

```
    ! Do nothing for Heidenhain
  end else if CLD[4]=1079 then begin ! Transform LCS
    Output "CYCL DEF 7.0 DATUM SHIFT"
    Output "CYCL DEF 7.1 X" + Str(Cmd.Flt["WCS.OriginPoint.X"])
    Output "CYCL DEF 7.2 Y" + Str(Cmd.Flt["WCS.OriginPoint.Y"])
    Output "CYCL DEF 7.3 Z" + Str(Cmd.Flt["WCS.OriginPoint.Z"])

    if (abs(Cmd.Flt["WCS.RotAngles.A"])>0.0001) or
       (abs(Cmd.Flt["WCS.RotAngles.B"])>0.0001) or
       (abs(Cmd.Flt["WCS.RotAngles.C"])>0.0001)
    then begin
      Output "CYCL DEF 19.0 WORKING PLANE"
      if Cmd.Int["IsSpatial"]>0 then begin ! Spatial coordinates
        Output "CYCL DEF 19.1 A" + str(Cmd.Flt["WCS.RotAngles.A"]) +
                              " B" + str(Cmd.Flt["WCS.RotAngles.B"]) +
                              " C" + str(Cmd.Flt["WCS.RotAngles.C"])
      end else begin ! Machine axes
        OutStr$ = "CYCL DEF 19.1"
        if Cmd.Ptr["Axes(AxisBPos)"]>0 then begin
          OutStr$ = OutStr$ + " B" + str(Cmd.Flt["Axes(AxisBPos).Value"])
        end
        if Cmd.Ptr["Axes(AxisCPos)"]>0 then begin
          OutStr$ = OutStr$ + " C" + str(Cmd.Flt["Axes(AxisCPos).Value"])
        end
        Output OutStr$
      end
    end
  end
end
```

5.1.23 Смена палеты <PALETA>

Команда <**PALETA**> предназначена для формирования кадров управляющей программы, осуществляющих смену палеты. Номер новой палеты передается первым параметром команды.

Команда:

```
PALETA N n
```

Параметры:

Параметр	CLD массив		Описание
n	CLD[1]	CLD.N	Номер палеты

5.1.24 Номер детали <PARTNO>

Команда <**PARTNO**> всегда является первой командой в списке технологических команд проекта. Поэтому данную команду следует использовать для формирования начальных кадров управляющей программы. В качестве параметра команды передается текстовая строка, содержащая имя файла проекта SprutCAM. Данная строка записывается в предопределенную переменную <**CLDATA$**>.

Команда:

```
PARTNO "..."
```

5.1.25 Рабочая плоскость <PLANE>

Команда <**PLANE**> устанавливает активную плоскость обработки. Обычно по данной команде формируются кадры выбора активной плоскости G17, G18, G19. Активная плоскость используется при круговой интерполяции (определяет плоскость расположения дуг окружностей), при отработке коррекции на радиус инструмента, при задании циклов обработки отверстий и в других случаях.

Команда:

```
PLANE XY(33) | YZ(37) | XZ(41) | YX(133) | ZY(137) | ZX(141)
```

Параметры:

Параметр	CLD массив		Описание
XY(33) YZ(37) XZ(41) YX(133) ZY(137) ZX(141)	CLD[1]	CLD.Plane	Код плоскости: 33 – XY, 37 – YZ, 41 – XZ, 133 – YX, 137 – ZY, 141 – ZX.

Для создания аналогичных ISO постпроцессоров в масках возможно использование предопределённых значений ISO.G, которые соответствуют параметрам команды:

Значение параметра CLD	Значение ISO
CLD[1] = 33	ISO.G = 17
CLD[1] = 37	ISO.G = 19
CLD[1] = 41	ISO.G = 18

5.1.26 Постпроцессорная функция <PPFUN>

Постпроцессорная функция <**PPFUN**> является универсальной командой генератора постпроцессоров, которая может иметь различное назначение в зависимости от ее типа. Тип постпроцессорной функции определяется первым параметром команды <**CLD[1]**> (<**CLD.SubCode**>). Все остальные параметры команды от <**CLD[2]**> до <**CLD[257]**> могут менять свое назначение для каждого конкретного типа функции.

Команда:

```
PPFUN PPFUN(500) | STARTSUB(50) | ENDSUB(51) | CALLSUB(52) |
      REPSTART(53) | REPEND(54) | JUMP(55) | TECHINFO(58) | ENDTECHINFO(59)
      WEDMConditions(56) {, a} {, b} {, c} {, d} …
```

Параметры:

Параметр	CLD массив		Описание
PPFUN(500), STARTSUB(50), ENDSUB(51), CALLSUB(52), REPSTART(53), REPEND(54), JUMP(55), TECHINFO(58), WEDMConditions(56)	CLD[1]	CLD.SubCode	Тип постпроцессорной функции
a, b, c, d…	CLD[2] – CLD[257]		Список необязательных параметров. Состав параметров индивидуален для каждого типа постпроцессорной функции.

В настоящее время используются следующие типы постпроцессорных функций специального назначения:

- Команда контроля стойкости инструмента <**PPFUN PPFUN(500)**>
- Команда начала NC-подпрограммы <**PPFUN STARTSUB(50)**>
- Команда окончания NC-подпрограммы <**PPFUN ENDSUB(51)**>
- Команда вызова NC-подпрограммы <**PPFUN CALLSUB(52)**>
- Команда задания технологических параметров операции <**PPFUN TECHINFO(58)**>
- Команда окончания технологической информации операции <**PPFUN ENDTECHINFO(59)**>
- Команда определения условий обработки электроэрозионных операций <**PPFUN WEDMConditions(56)**>

Команда контроля стойкости инструмента <PPFUN PPFUN(500)>

Команда <**PFUN PPFUN(500)**> формируется, когда необходимо выполнить смену инструмента в связи с его износом. В этом случае параметр <**CLD[2]**> принимает значение стойкости инструмента в минутах.

Команда начала NC-подпрограммы <PPFUN STARTSUB(50)>

Команда начала NC-подпрограммы <**PPFUN STARTSUB(50)**>, наряду с командой окончания NC-подпрограммы <**PPFUN ENDSUB (51)**>, используется для выделения из общей последовательности технологических команд, частей, являющихся NC-подпрограммами. Все технологические команды, находящиеся между <**PPFUN STARTSUB(50)**> и <**PPFUN ENDSUB(51)**>, считаются принадлежащими не главной программе, а NC-подпрограмме. Команды, принадлежащие главной программе транслируются последовательно одна за другой, начиная с первой. Команды, принадлежащие NC-подпрограммам пропускаются. Они начинают транслироваться только при вызове специальных операторов вывода NC-подпрограмм из команд обработчиков главной программы.

Вторым параметром <**CLD[2]**> команды передается уникальный номер NC-подпрограммы.

Команда окончания NC-подпрограммы <PPFUN ENDSUB(51)>

Команда окончания NC-подпрограммы <**PPFUN ENDSUB(51)**>, наряду с командой начала NC-подпрограммы <**PPFUN STARTSUB (50)**>, используется для выделения из общей последовательности технологических команд, частей, являющихся NC-подпрограммами . Все технологические команды, находящиеся между <**PPFUN STARTSUB(50)**> и <**PPFUN ENDSUB(51)**>, считаются принадлежащими не главной программе, а NC-подпрограмме. Команды, принадлежащие главной программе транслируются последовательно одна за другой, начиная с первой. Команды, принадлежащие NC-подпрограммам пропускаются. Они начинают транслироваться только при вызове специальных операторов вывода NC-подпрограмм из команд обработчиков главной программы.

Вторым параметром <**CLD[2]**> команды передается уникальный номер NC-подпрограммы.

Команда вызова NC-подпрограммы <PPFUN CALLSUB(52)>

Команда вызова NC-подпрограммы <**PPFUN CALLSUB(52)**> используется в тех случаях, когда управляющая программа состоит из нескольких NC-подпрограмм, и когда в процессе выполнения одной программы требуется передать управление другой программе. В таких случаях внутри обработчика команды < **PPFUN CALLSUB(52)**> следует вызывать специальные операторы вывода NC-подпрограмм. Они заставляют систему начать транслировать команды, которые принадлежат NC-подпрограммам. Иначе, технологические команды, принадлежащие NC-подпрограммам, транслятором игнорируются.

Вторым параметром <**CLD[2]**> команды передается уникальный номер NC-подпрограммы.

Команда задания технологических параметров операции <PPFUN TECHINFO(58)>

Команда задания технологических параметров операции <**PPFUN TECHINFO(58)**> служит для передачи в постпроцессор дополнительной технологической информации об операции SprutCAM. Она обычно располагается в самом начале каждой последовательности технологических команд, соответствующих одной операции.

Параметры:

Массив CLD	Параметр	Описание переменной	Группа
1	SubCode	58 (TechInfo)	
2	2	Версия CLData	
3	3	Минимальная координата по X	Оболочка траектории
4	4	Минимальная координата по Y	
5	5	Минимальная координата по Z	
6	6	Максимальная координата по X	
7	7	Максимальная координата по Y	
8	8	Максимальная координата по Z	
9	9	Уровень безопасной плоскости (уровень ускоренных перемещений в эрозионных операциях)	Уровни
10	10	Верхний уровень обработки	
11	11	Нижний уровень обработки	
12	12	Припуск	Заготовка (Оболочка всех операций)
13	13	Минимальная координата по X	
14	14	Минимальная координата по Y	
15	15	Минимальная координата по Z	
16	16	Максимальная координата по X	
17	17	Максимальная координата по Y	
18	18	Максимальная координата по Z	
19	19	Точность CLData (количество знаков после запятой)	Точности
20	20	Система счисления: 0 – мм	

		1 – дюйм	
21	21	Минимальная длина дуги	
22	22	Отклонение от детали	
23	23	Отклонение в деталь	
24	24	Припуск в операции	
25	25	Тип инструмента: 0 – цилиндрический 1 – сферический 2 – тороидальный 3 – с двумя радиусами 4 – с двумя радиусами и ограничением 5 – с углом и двумя радиусами 6 – с углом, двумя радиусами и ограничением 7 – гравер 8 – сверло 9 – составной инструмент 10 – инструмент с обратным радиусом 11 – резак 12 – метчик 13 – центровочное сверло 14 – зенковка 15 – цековка 16 – развертка 17 – резьбонарезная фреза 100 – проходной резец 101 – расточной резец 102 – наружный резьбонарезной резец 103 – внутренний резьбонарезной резец 104 – наружный канавочный резец 105 – внутренний канавочный резец 106 – торцевой канавочный резец	Инструмент
26	26	Номер инструмента	
27	27	Диаметр инструмента (диаметр проволоки в эрозионных операциях)	
28	28	Длина рабочей части	
29	29	Угол рабочей части (угол конической части в радианах)	
30	30	Радиус торца (радиус скругления фрезы, радиус кончика резца)	
31	31	Номер корректора на длину. 0, если коррекция на длину выключена	
32	32	Номер корректора на радиус. 0, если коррекция на радиус выключена	
33	33	Величина коррекции на длину	
34	34	Величина коррекции на радиус	
35	35	Точка настройки осевого инструмента: 0 – центр инструмента, 1 – произвольная точка, заданная смещением относительно центра инструмента, 2 – кончик инструмента. Точка настройки токарного инструмента: 0 – левая, 1 – правая, 2 – верхняя, 3 – нижняя,	

		4 – верхняя левая, 5 – верхняя правая, 6 – нижняя левая, 7 – нижняя правая, 8 – центральная, 9 – произвольная.	
36	36	Частота вращения шпинделя (об/мин)	Скорости
37	37	Подача ускоренного перемещения (мм/мин или дюйм/мин)	
38	38	Подача рабочая	
39	39	Подача врезания	
40	40	Подача подхода	
41	41	Подача отхода	
43	43	Подача перехода от строчки к строчке	
43	43	Подача возврата	
44	44	Длина траектории	Статистика траектории
45	45	Время операции (8 байт в формате Delphi)	
46	46	Время всей обработки	
47	47	Уровень верхней направляющей эрозионного станка	Уровни
48	48	Уровень нижней направляющей эрозионного станка	
51	51	Ось инструмента: 1 – X, 2 – Y, 3 – Z (Для токарных операций равна 2, для токарного сверления равна 1, для фрезерных операций равна 3)	
52	52	Координата X	Точка смены инструмента
53	53	Координата Y	
54	54	Координата Z	
55	55	Координата X	Промежуточная точка
56	56	Координата Y	
57	57	Координата Z	
58	58	Номер включенного трубопровода охлаждения: 0 – охлаждение выключено, 1 – жидкость, 2 – туман, 3 – инструмент.	
60	60	Технологическая группа операции: 0 – не установлена, 1 – фрезерная, 2 – токарная, 3 – вспомогательная, 4 – электроэрозионная.	
80	80	Шаг вдоль оси инструмента для многослойной обработки (только для операции 2D Контур)	Слои
81	81	Припуск вдоль оси инструмента на чистовой проход (только для операции 2D Контур)	Слои

Параметры, доступные через оператор Cmd

TCLDPP	Команда задания технологических параметров операции

Fun: Complex Type	**PPFun**: Array, Key="RecordID"	**TPPFun TechInfo Parameters**: Complex Type	Cmd.Ptr["PPFun"].Item[1] или Cmd.Ptr["PPFun (TechInfo)"] - Отдельный элемент массива PPFun. Содержит информацию о технологических параметрах операции. Доступ к элементам массива возможен либо по индексу, либо по ключевому полю <RecordID>. Для команды задания технологических параметров оно всегда имеет значение "TechInfo".		
			RecordID: String	Cmd.Str["PPFun(TechInfo).RecordID"] - Идентификатор типа постпроцессорной функции. В данном случае всегда имеет значение "TechInfo".	
			RecordCode: Integer	Cmd.Int["PPFun(TechInfo).RecordCode"] - Код, определяющий тип постпроцессорной функции. В данном случае всегда имеет значение "58".	
			Operation: Array	Cmd.Ptr["PPFun(TechInfo).Operation"] - практически полный список параметров технологической операции. В качестве дочерних элементов данного пункта служат самые различные параметры операции: тип операции, ее имя, группа, свойства инструмента, припуски, точности вычислений, способы подходов-отходов, уровни и шаг обработки и т.д. Конкретный набор параметров зависит от типа операции. Тип операции может быть получен командой Cmd.Ptr["PPFun(TechInfo). Operation(1)"].Name.	
			Tool: Complex Type	Cmd.Ptr["PPFun(TechInfo).Tool"] - Параметры, относящиеся к инструменту, используемому в технологической операции.	
				ID: Integer	Cmd.Int["PPFun(TechInfo).Tool. ID"] - идентификатор (номер) инструмента
				HolderID: Integer	Cmd.Int["PPFun(TechInfo).Tool. HolderID"] - идентификатор (номер) держателя инструмента
				RevolverID: String	Cmd.Str["PPFun(TechInfo).Tool. RevolverID"] - строковый идентификатор револьверной головки
			Workpiece: Complex Type	Cmd.Ptr["PPFun(TechInfo).Workpiece"] - Параметры, относящиеся к заготовке, используемой в технологической операции.	
				HolderID: String	Cmd.Str["PPFun(TechInfo). Workpiece.HolderID"] - строковый идентификатор

					держателя заготовки.

Ниже приведен пример доступа к свойствам операции при помощи оператора Cmd.

```
program PPFun
  if cld[1]=58 then begin ! TechInfo
    call AnalysePPFun
  end
end

sub AnalysePPFun
  OpType: String
  OpGroup: Integer

  ! String name of operation type: TSTRoughingWaterlineOp, TST2DContouringOp, TSTDrillOp,
  OpType = Cmd.Ptr["PPFun(TechInfo).Operation(1)"].Name

  ! OperationGroup:
  !   1 - Milling
  !   2 - Lathe
  !   3 - Auxiliary
  !   4 - WireEDM
  OpGroup = Cmd.Int["PPFun(TechInfo).Operation(1).OperationGroup"]

  Output "; Operation type = " + OpType
  Output "; Operation comment = " + Cmd.Str["PPFun(TechInfo).Operation(1).Comment"]
  if OpGroup=1 then begin ! Milling
    Output "; Tool overhang = " + Str(Cmd.Flt["PPFun(TechInfo).Operation(1).ToolSection.To

    if OpType="TSTRoughingWaterlineOp" then begin
      Output "; Axial stock = " + Str(Cmd.Flt["PPFun(TechInfo).Operation(1).Stocks.Axial"]
      Output "; Radial stock = " + Str(Cmd.Flt["PPFun(TechInfo).Operation(1).Stocks.Radial
    end
  end else if OpGroup=2 then begin ! Lathe
    Output "; Tool X overhang = " + Str(Cmd.Flt["PPFun(TechInfo).Operation(1).ToolSection.
    Output "; Tool Y overhang = " + Str(Cmd.Flt["PPFun(TechInfo).Operation(1).ToolSection.
    Output "; Tool Z overhang = " + Str(Cmd.Flt["PPFun(TechInfo).Operation(1).ToolSection.

    Output "; Axial stock = " + Str(Cmd.Flt["PPFun(TechInfo).Operation(1).LatheStock.Axial
    Output "; Radial stock = " + Str(Cmd.Flt["PPFun(TechInfo).Operation(1).LatheStock.Radi
  end
subend
```

Команда окончания технологической информации операции <PPFUN ENDTECHINFO(59)>

Данная команда обычно является последней в списке технологических команд данной операции и служит для обнаружения конца операции. Команда содержит набор параметров полностью совпадающий с параметрами команды ≤ **PPFUN TECHINFO(58)**≥. В нем содержится технологическая информация об операции SprutCAM.

Команда определения условий обработки электроэрозионных операций <PPFUN WEDMConditions(56)>

Команда определения условий обработки электроэрозионных операций <**PPFUN WEDMConditions(56)**> генерируется электроэрозиоными операциями SprutCAM. Она служит для передачи внутрь постпроцессора таблицы условий обработки электроэрозионной операции. Каждая строка в таблице условий обработки соответствует условиям обработки для одного прохода. Соответственно количество строк в таблице равно используемому количеству проходов. Каждая строка таблицы передается в постпроцессор отдельной командой <**PPFUN WEDMConditions(56)** >. Таким образом, в начале списка технологических команд каждой электроэрозионной операции SprutCAM должно присутствовать столько команд определения условий обработки, сколько строк в таблице условий обработки.

Параметры:

Параметр	**CLD массив**		**Описание**
Идентификатор команды	CLD[1]	CLD.SubCode	WEDMCONDITIONS(56) – строка таблицы условий обработки для электроэрозионной операции
Порядковый номер прохода	CLD[2]		Порядковые номера проходов обычно назначаются последовательно, начиная с 1. Однако порядковый номер может начинаться и не с 1, если в SprutCAM указан параметр "начальный номер прохода", отличный от 1
Код условий обработки прохода	CLD[3]		Код условий обработки определяет режимы резания для данного прохода. Переключение кода условий обработки по ходу программы производится командой FEDRAT.
Код смещения (номер корректора)	CLD[4]		Код смещения определяет номер корректора, из которого следует брать величину коррекции для данного прохода. Переключение номера корректора по ходу программы производится командой CUTCOM.
Величина коррекции	CLD[5]		Величина коррекции для данного номера корректора
Величина подачи	CLD[6]		Величина подачи проволоки в мм/мин (дюйм/мин)
Дополнительные параметры	CLD[100]-CLD[256]		Список дополнительных параметров для каждого прохода. Набор дополнительных

Параметр	CLD массив		Описание
			параметров может быть индивидуальным для каждого конкретного постпроцессора.

5.1.27 Печать постпроцессора <PPRINT>

Текст попадает в предопределенную переменную <**CLDATA$**>, после чего может быть выдан в окно оператора как комментарий, поясняющий действия постпроцессора.

Команда:

```
PPRINT "..."
```

5.1.28 Быстрый ход <RAPID>

Команда <**RAPID**> активирует режим ускоренных перемещений (часто соответствует G-коду G00). Все последующие перемещения инструмента до ближайшей команды переключения подачи <**FEDRAT**> будут совершаться в ускоренном режиме.

Команда:

```
RAPID N n
```

Параметры:

Параметр	CLD массив		Описание
n	CLD[1]	CLD.N	Величина ускоренной подачи

5.1.29 Поворот стола <ROTABL>

Команда <**ROTABL**> предназначена для позиционирования поворотной оси станка. Данная команда CLData является устаревшей и может присутствовать только в старых проектах SprutCAM. В новых проектах перемещения поворотных осей передаются через команду <**MULTIGOTO**>.

Команда:

```
ROTABL ABS(0), A ab, CCLW(59)|CLW(60), PLANE XY(33)|YZ(37)|XZ(41)
```

или

```
ROTABL INCR(66), B ab, CCLW(59)|CLW(60), PLANE XY(33)|YZ(37)|XZ(41)
```

Параметры:

Параметр	CLD массив		Описание
ABS(0) INCR(66)	CLD[1]	CLD.Incr	Тип поворота: 0 – (ABS) абсолютное, 66 – (INCR) относительное
ab	CLD[2]	CLD.AB	угол поворота стола или приращение этого угла
CCLW(59), CLW(60)	CLD[3]	CLD.CCLW	направление вращения против или по часовой стрелке
XY(33), YZ(37), XZ(41)	CLD[4]	CLD.Plane	код плоскости

Для создания аналогичных ISO постпроцессоров в масках возможно использование предопределённых значений ISO.A, ISO.B, ISO.C которые соответствуют параметрам команды:

Значение параметра CLD	Значение ISO
CLD[4] = 33 CLD[1] = 0 CLD[3] = 59	ISO.C = CLD[2]
CLD[4] = 33 CLD[1] = 0 CLD[3] = 60	ISO.C = -CLD[2]
CLD[4] = 33 CLD[1] = 66	ISO.C = CLD[2]
CLD[4] = 41 CLD[1] = 0 CLD[3] = 59	ISO.B = CLD[2]
CLD[4] = 41 CLD[1] = 0 CLD[3] = 60	ISO.B = -CLD[2]
CLD[4] = 41 CLD[1] = 66	ISO.B = CLD[2]
CLD[4] = 37 CLD[1] = 0 CLD[3] = 59	ISO.A = CLD[2]
CLD[4] = 37 CLD[1] = 0 CLD[3] = 60	ISO.A = -CLD[2]
CLD[4] = 37 CLD[1] = 66	ISO.A = CLD[2]

5.1.30 Точка смены инструмента <SAFPOS>

В команде <**SAFPOS**> задаются координаты точки, в которую приводится центр инструмента для смены инструмента (команда <**LOADTL**>).

Команда:

```
SAFPOS X x, Y y, Z z, N n
```

Параметры:

Параметр	CLD массив		Описание
x, y, z	CLD[1] CLD[2] CLD[3]	CLD.X CLD.Y CLD.Z	координаты точки смены инструмента
N	CLD[4]	CLD.N	номер точки смены инструмента в станке

5.1.31 Выбор активной державки заготовки <SELWORKPIECE>

Команда <**SELWORKPIECE**> осуществляет установку активной державки заготовки. Под державкой заготовки в данном случае понимается узел станка, в котором производится закрепление заготовки. Для токарных многошпиндельных станков данная команда определяет активный шпиндель. Для фрезерных станков державкой заготовки может являться, например, палета. Все последующие команды обработки будут относиться к обработке заготовки, закрепленном именно в активной державке.

Команда:

```
SELWORKPIECE WorkpieceHolderID
```

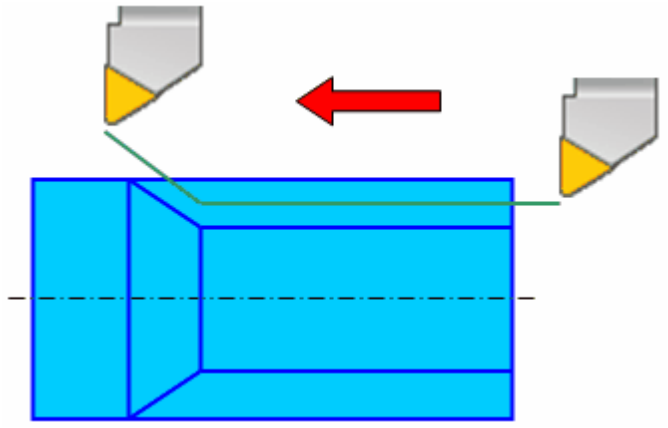
Единственным параметром <**WorkpieceHolderID**> команды является текстовый идентификатор державки заготовки. Значение этого идентификатора определяется используемой схемой станка SprutCAM. Данное значение записывается в предопределенную переменную <**CLDATA$**>. Ниже приведен простой пример программы обработки для данной команды.

```
program SelWorkpiece
  if CLData$="MainSpindle" then
    Output "SETMS(1)" ! Активация главного шпинделя
  else
    Output "SETMS(2)" ! Активация противошпинделя
end
```

5.1.32 Нарезание резьбы за один проход <SINGLETHREAD>

Команда <**SINGLETHREAD**> активирует режим непрерывного нарезания цилиндрической или конической резьбы с постоянным шагом токарным резцом. Часто данной команде соответствуют G-

коды G32 или G33.



При активации режима нарезания резьбы включается специальный режим синхронизации между перемещением инструмента и вращением шпинделя. Все последующие перемещения инструмента до ближайшей команды переключения подачи <FEDRAT> или команды включения режима ускоренных перемещений <RAPID> будут совершаться в данном режиме нарезания резьбы. Если инструмент совершает перемещение параллельно оси вращения шпинделя, то формируется цилиндрическая резьба. Если инструмент перемещается одновременно вдоль оси и перпендикулярно оси вращения получается участок конической резьбы. Возможно и нарезание специальной торцевой резьбы, когда инструмент совершает перемещения только перпендикулярно оси вращения шпинделя. В этом случае на торце формируется канавка в форме спирали Архимеда.

В параметрах команды задается ориентация резьбы, величина шага резьбы, а также начальное угловое положение шпинделя. Шаг резьбы может задаваться двумя способами: либо указанием расстояния между двумя соседними витками, либо указанием количества витков, приходящегося на единицу длины.

Начальное угловое положение шпинделя задается в градусах и определяет величину угла, который отсчитывается от нулевого импульса датчика шпинделя до начала нарезания резьбы. Задание различных значений угла для разных проходов позволяет получить многозаходную резьбу.

Команда:

```
SINGLETHREAD OD(0) | ID(1) | FACE(2), STEP(0) | COUNT(1), f, ANGLE q
```

Параметры:

Параметр	CLD массив		Описание
OD(0), ID(1), FACE(2)	CLD[1]	CLD.Orientation	Ориентация резьбы: OD(0) – наружная, ID(1) – внутренняя, FACE(2) – торцевая.

Параметр	CLD массив		Описание
STEP(0), COUNT(1)	CLD[2]	CLD.IsCount	Способ задания шага резьбы: STEP(0) – расстоянием, COUNT(1) – количеством витков на единицу длины.
f	CLD[3]	CLD.Value	Значение шага резьбы.
q	CLD[4]	CLD.StartAngle	Начальное угловое положение шпинделя в градусах (используется для многозаходной резьбы).

5.1.33 Шпиндель <SPINDL>

В команде <**SPINDL**> передается информация о режимах включения и выключении шпинделя. Она позволяет задавать частоту или диапазон частот вращения шпинделя, линейную скорость резания и др.

Команда:

```
SPINDL ON(71) | OFF(72) | ORIENT(246), NO n, K k,
       MODE RPM(0) | RANGE(1) | CSS(2) {, SPEED c}
```

Параметры:

Параметр	CLD массив		Описание
ON(71), OFF(72), ORIENT(246)	CLD[1]	CLD.OnOff	Тип команды: ON(71) – включение шпинделя, OFF(72) – выключение шпинделя, ORIENT(246) – ориентированный останов шпинделя.
n	CLD[2]	CLD.NO	Частота вращения, угол поворота шпинделя или максимальная частота вращения шпинделя в режиме CSS.
k	CLD[3]	CLD.K	Диапазон частот вращения
RPM(0), RANGE(1), CSS(2)	CLD[4]	CLD.Mode	Режим задания частоты вращения шпинделя: RPM(0) – в об/мин, RANGE(1) – диапазон частот вращения, CSS(2) – режим постоянной скорости резания.
c	CLD[5]	CLD.CSS	Значение постоянной скорости резания для режима CSS.

Для создания аналогичных ISO постпроцессоров в масках возможно использование предопределённых значений ISO.M, которые соответствуют параметрам команды:

Значение параметра CLD	Значение ISO
CLD[1] = 71	ISO.M = 3

CLD[1] = 72	ISO.M = 5

5.1.34 Останов <STOP>

Команда <**STOP**> предназначена для формирования кадров прерывания выполнения управляющей программы. Обычно ей соответствует код M00 управляющей программы. При достижении кадра останова исполнение управляющей программы прерывается.

Команда:

```
STOP
```

5.1.35 Перехват заготовки <TAKEOVER>

Команда <**TAKEOVER**> осуществляет перехват заготовки из активной державки заготовки в державку, которая указана в параметре команды. Активная державка заготовки определяется командой <**SELWORKPIECE**>. Под державкой заготовки в данном случае понимается узел станка, в котором производится закрепление заготовки. Для токарных и токарно-фрезерных многошпиндельных станков данная команда определяет перехват заготовки из одного шпинделя в другой.

Команда:

```
TAKEOVER WorkpieceHolderID
```

Единственным параметром <**WorkpieceHolderID**> команды является текстовый идентификатор державки заготовки. Значение этого идентификатора определяется используемой схемой станка SprutCAM. Данное значение записывается в предопределенную переменную <**CLDATA$**>.

5.1.36 Ожидание точки синхронизации <WAIT>

Команда <**WAIT**> используется для синхронизации нескольких параллельных потоков управления (нескольких управляющих программ, каждая из которых может управлять отдельным рабочим органом станка). По этой команде в управляющую программу следует вывести кадры-метки, осуществляющие приостановку выполнения программы до момента достижения аналогичной метки

в другой (параллельно выполняющейся) программе.

Команда:

```
WAIT ID(<SynchPointID>), Index(<SynchPointIndex>)
```

Параметры, доступные через оператор Cmd

TCLDSyncPoint: ComplexType	Команда ожидания точки синхронизации	
	SyncPointID: String	Cmd.Str["SyncPointID"] - Уникальный идентификатор точки синхронизации, определяемый пользователем.
	Index: Integer	Cmd.Int["Index"] - Порядковый номер точки синхронизации, автоматически формируемый системой.