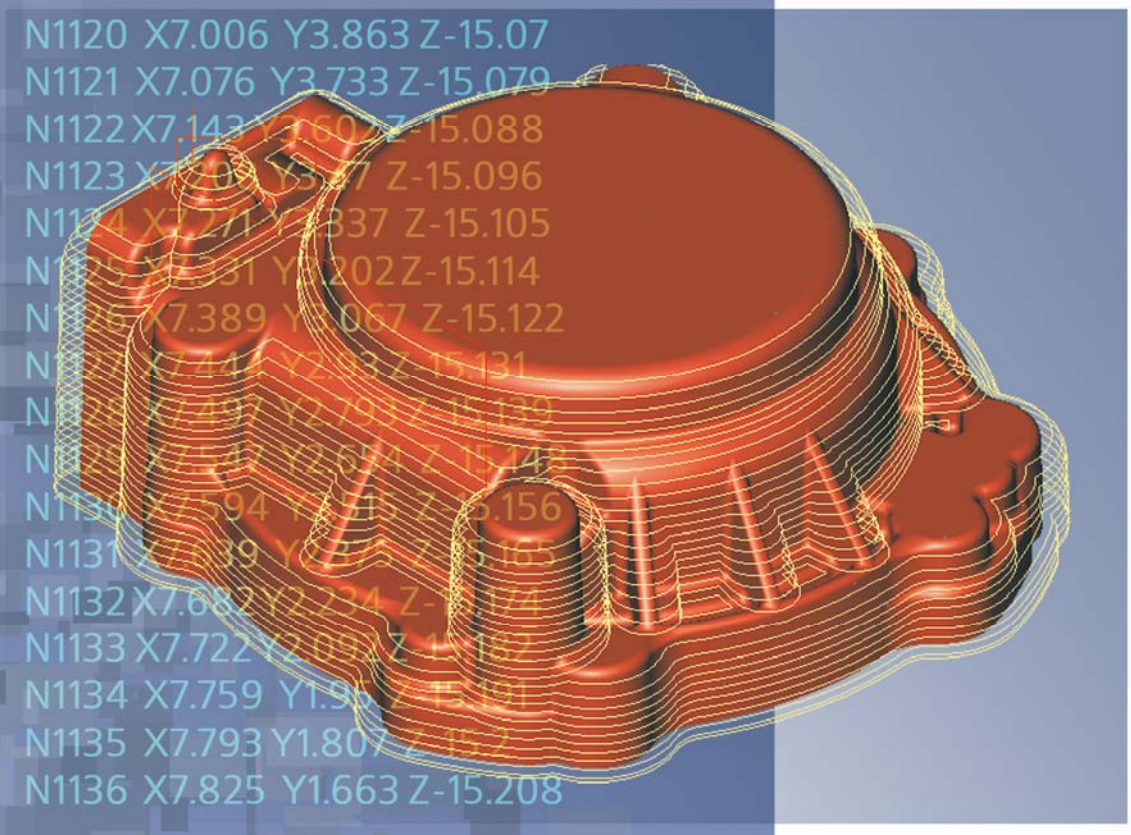
SPRUT TECHNOLOGY

Генератор постпроцессоров



СОДЕРЖАНИЕ

СОДЕРЖАНИЕ 3
ВВЕДЕНИЕ 7
НАЗНАЧЕНИЕ ГЕНЕРАТОРА ПОСТПРОЦЕССОРОВ 7
НАБОР ФАЙЛОВ ГЕНЕРАТОРА ПОСТПРОЦЕССОРОВ 7
1 ОБЩАЯ ОРГАНИЗАЦИЯ РАБОТЫ 8
1.1 ПРИНЦИП РАБОТЫ ГЕНЕРАТОРА ПОСТПРОЦЕССОРОВ 8
1.2 ГЛАВНОЕ ОКНО 10
1.2.1 Основное меню 11
1.2.2 Главная панель 13
1.2.3 Индикатор процесса 15
1.2.4 Системные установки 15
1.2.5 Настройка редактора 16
1.2.6 Заполнение общих параметров 17
1.2.7 Обязательное заполнение пользователем информации о системе ЧПУ и станке 19
1.2.8 Определение структуры и формата кадра (формирование списка регистров) 22
1.2.9 Шаблоны трансляции технологических команд 25
1.2.10 Программы обработки технологических команд 26
1.2.11 Подпрограммы 27
1.2.12 Трансляция программ обработки технологических команд 27
1.2.13 Работа с файлами технологических команд 27
1.2.14 Тестовая генерация управляющих программ 28
1.2.15 Отладка программ 29
1.2.16 Обратная интерпретация программ 31
2 ШАБЛОНЫ 33
2.1 ОСНОВНЫЕ ПОНЯТИЯ 33
2.2 СОСТАВ ШАБЛОНА 34
2.2.1 Элемент шаблона 34
2.2.2 Регистры в шаблонах 35
2.2.3 Модификатор 36
2.2.4 Выражение 36
2.2.5 Вложенный шаблон 37
2.2.6 Разделители элементов шаблона 38
2.2.7 Занесение значений в переменные 38
2.2.8 Отложенные шаблоны 38
2.3 УПРАВЛЕНИЕ ШАБЛОНОМ 41
2.3.1 Переключатели шаблона 41
2.3.2 Преобразование шаблона в подпрограмму 41
2.3.3 Визуальные средства формирования шаблона 41
3 ОПИСАНИЕ ЯЗЫКА 44
3.1 ОСНОВНЫЕ ПОНЯТИЯ 44
3.1.1 Условные обозначения 44
3.1.2 Программы обработки технологических команд, комментарии в программе 44
3.1.3 Подпрограммы 45
3.1.4 Понятие оператора языка 45
3.1.5 Набор символов 45

3.1.6 Переменные 46
3.1.7 Массивы 46
3.1.8 Математические выражения и функции 47
3.1.9 Предопределенные переменные 48
3.1.10 Предопределённые функции 50
3.2 ОПЕРАТОРЫ 52
3.2.1 Оператор начала программы обработки технологической команды PROGRAM 52
3.2.2 Оператор присваивания = ... 52
3.2.3 Оператор вывода PRINT 53
3.2.4 Оператор ввода INPUT 54
3.2.5 Оператор условного выполнения IF 54
3.2.6 Оператор множественного условного выполнения CASE 56
3.2.7 Оператор перехода на метку JUMP 57
3.2.8 Оператор цикла FOR 57
3.2.9 Оператор цикла REPEAT 58
3.2.10 Оператор цикла WHILE 59
3.2.11 Составной оператор 59
3.2.12 Оператор вызова программы CALL 60
3.2.13 Оператор начала подпрограммы SUB 60
3.2.14 Оператор окончания подпрограммы SUBEND 61
3.2.15 Оператор начала процедуры PROC 61
3.2.16 Оператор возврата из процедуры RETURN 62
3.2.17 Оператор вывода кадра OUTBLOCK 62
3.2.18 Оператор формирования кадра FORMBLOCK 63
3.2.19 Оператор прямого вывода в кадр OUTPUT 63
3.2.20 Оператор замены подстроки в строке REPLACE 64
3.2.21 Оператор формирования кадра по шаблону MASK 64
3.2.22 Операторы работы с NC-подпрограммами 65
3.2.23 Оператор вывода NC-подпрограммы NCSUB.OUTPUT 66
3.2.24 Оператор определения имени NC-подпрограммы NCSUB.NAME 68
3.2.25 Оператор определения начальной метки NC-подпрограммы NCSUB.STARTLABEL 68
3.2.26 Оператор определения конечной метки NC-подпрограммы NCSUB.ENDLABEL 70
3.2.27 Методика работы с NC-подпрограммами 71
4 ПРИЛОЖЕНИЯ 75
4.1 СЛОВАРЬ СЛОВ CLDATA (ТЕХНОЛОГИЧЕСКИХ КОМАНД ПОДДЕРЖИВАЕМЫХ SPRUTCAM) 75
4.2 ФОРМАТ ТЕХНОЛОГИЧЕСКИХ КОМАНД 77
4.2.1 Номер детали 77
4.2.2 Конечная запись 77
4.2.3 Постпроцессорная функция 77
4.2.4 Комментарий 79
4.2.5 Линейные перемещения 80
4.2.6 Перемещение по окружности 80
4.2.7 Рабочая плоскость 81
4.2.8 Начальная точка 81
4.2.9 Возврат в исходное положение 81
4.2.10 Точка смены инструмента 82
4.2.11 Поворот стола 82
4.2.12 Компенсация инструмента 83

4.2.13 Загрузка инструмента 83
4.2.14 Выбор инструмента 84
4.2.15 Шпиндель 84
4.2.16 Останов 85
4.2.17 Дополнительный останов 85
4.2.18 Охлаждение 85
4.2.19 Подача 86
4.2.20 Быстрый ход 86
4.2.21 Выстой 86
4.2.22 Переход в абсолютную или относительную систему координат 86
4.2.23 Начальные координаты 87
4.2.24 Отмена и восстановление цикла 87
4.2.25 Цикл сверления типа G81 87
4.2.26 Цикл сверления типа G82 88
4.2.27 Цикл сверления типа G84 88
4.2.28 Цикл сверления типа G85 89
4.2.29 Цикл сверления типа G86 89
4.2.30 Цикл сверления типа G87 90
4.2.31 Цикл сверления типа G88 90
4.2.32 Цикл сверления типа G89 91
4.2.33 Глубокое сверление с полным выводом инструмента для удаления стружки 91
4.2.34 Глубокое сверление с отводами сверла для ломки стружки 91
4.2.35 Нарезание резьбы 92
4.2.36 Смена палеты 93
4.2.37 Головка 93
4.2.38 Коэффициенты сплайн-кривой 93
4.2.39 Начало сплайнового участка траектории 93
4.2.40 Вставка 94
4.2.41 Условный пропуск 94
4.2.42 Печать постпроцессора 94
4.2.43 Расширенный цикл 94
4.2.44 Нарезание резьбы за один проход 101

ВВЕДЕНИЕ

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.

НАБОР ФАЙЛОВ ГЕНЕРАТОРА ПОСТПРОЦЕССОРОВ

Inp.exe – исполняемый модуль генератора постпроцессоров.

InpD.dll – исполняющая система постпроцессора.

***.spp** – файлы исходных текстов программ обработки технологических команд и настройки постпроцессора на конкретную систему **ЧПУ**.

***.stc** – файлы проектов системы **SprutCAM**.

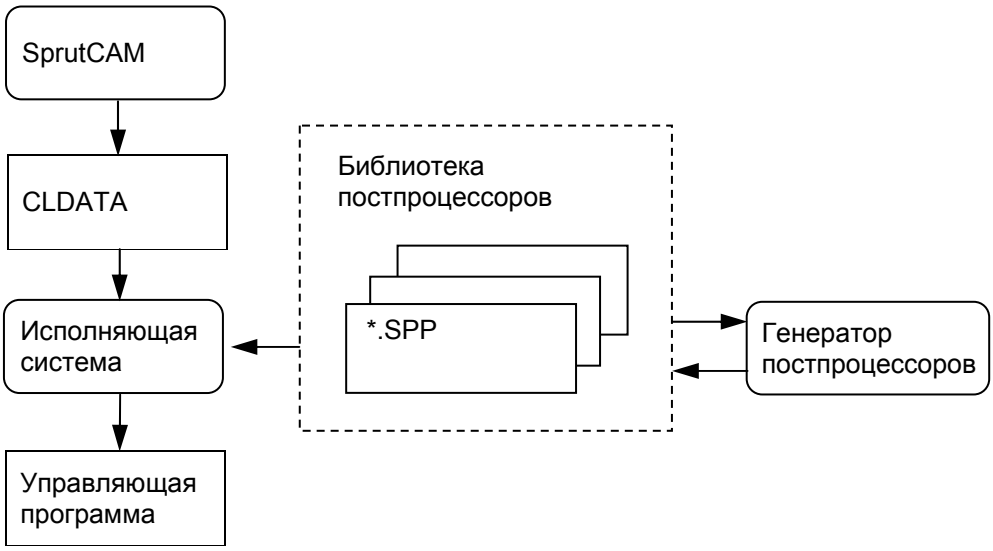
***.mcd** – связанные с проектом **SprutCAM** файлы технологических команд.

***.inp** ***.ppp** – файлы настройки для предыдущей версии постпроцессора.

1 ОБЩАЯ ОРГАНИЗАЦИЯ РАБОТЫ

1.1 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** 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;

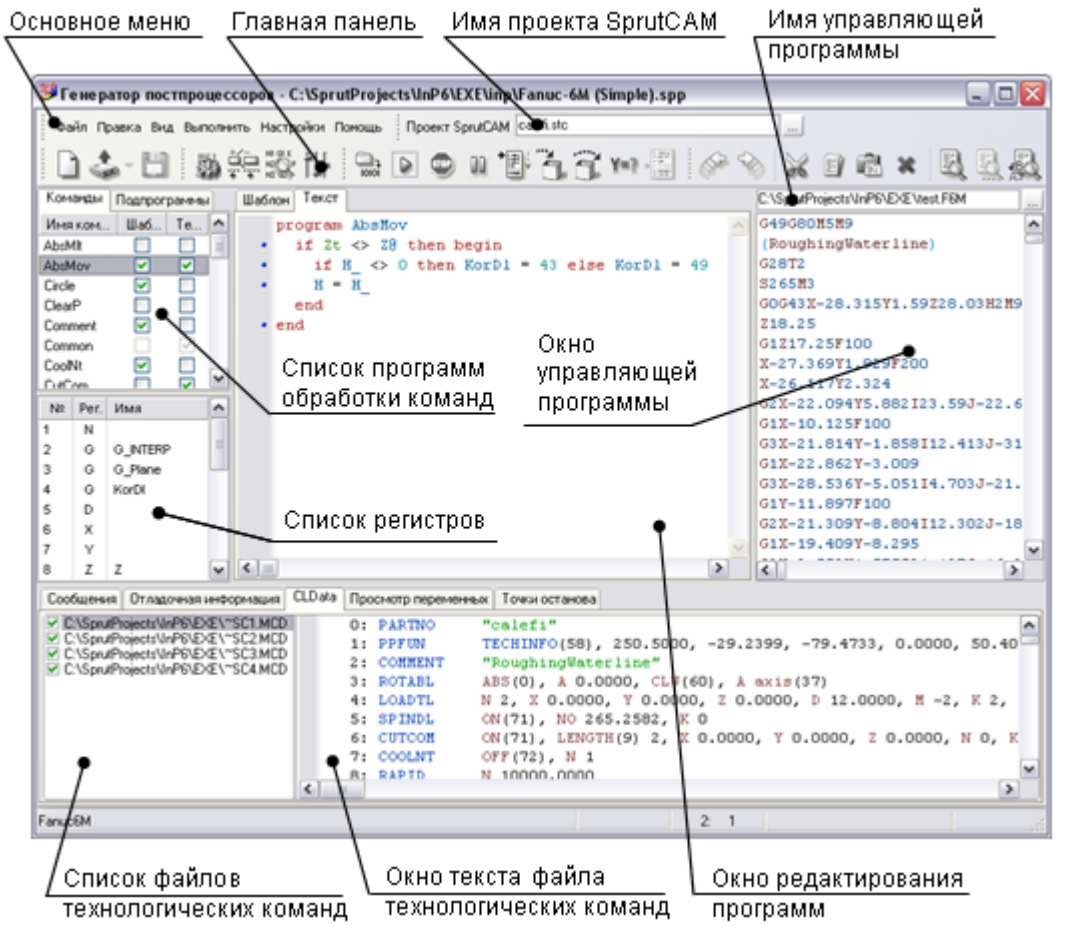
Исходные тексты программ обработки технологических команд, шаблоны, данные о станке и системе ЧПУ, маски, список и формат регистров сохраняются в файле с именем конкретного постпроцессора и расширением ***.SPP**. Данные из этого файла используются исполняющей системой постпроцессора при формировании 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**. При выполнении вызванной программы могут быть изменены значения регистров и внутренних переменных и сформированы кадры управляющей программы.

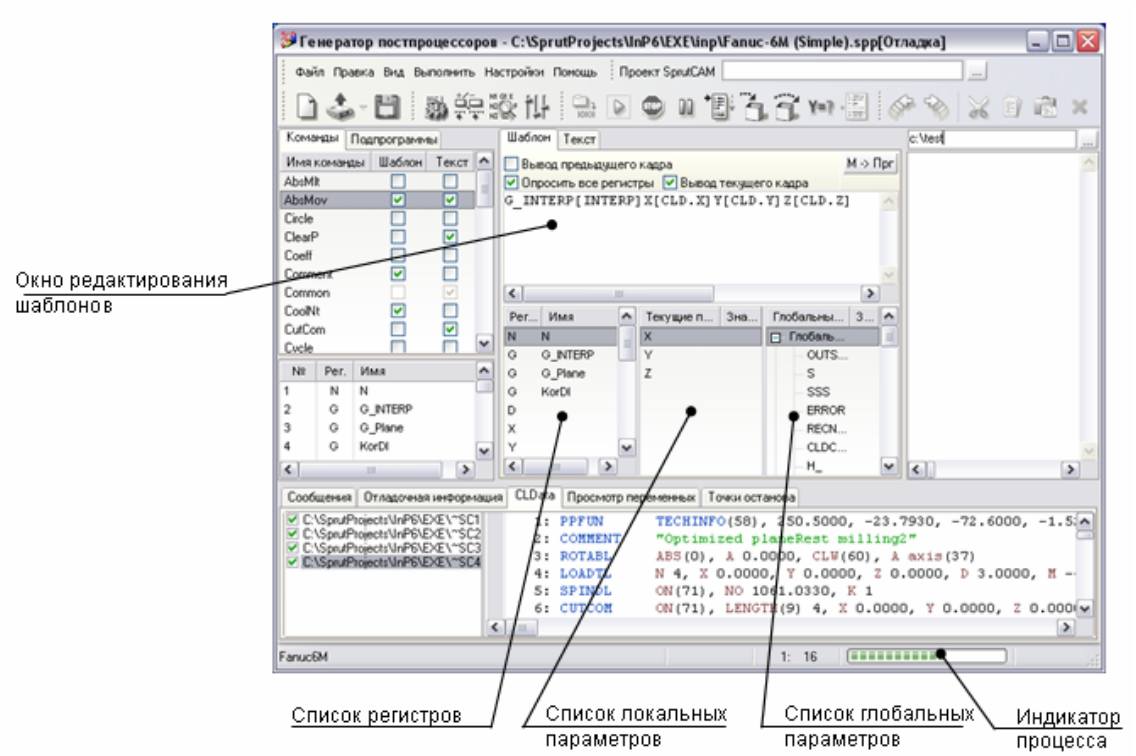
В программах обработки команд формирование кадра управляющей программы производится соответствующими операторами, причем в кадр выводятся идентификаторы и значения только тех регистров, которые были изменены после формирования предыдущего кадра.

1.2 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.



1.2.1 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 введённого 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.

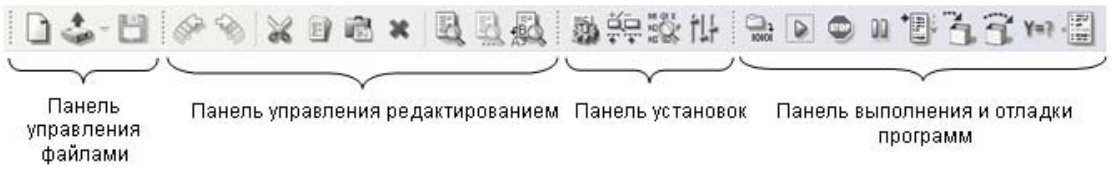
- **<Выполнить до курсора>** - выполнить программу обработки технологических команд до курсора.
- **<Шагнуть в>** - выполнить следующую команду программы обработки технологических команд с заходом в подпрограммы.
- **<Шагнуть через>** - выполнить следующую команду программы обработки технологических команд без захода в подпрограммы.
- **<Пауза>** - остановить 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 АО-“СПРУТ-Технология”. Адрес сервера: **http://www.sprut.ru**
  - **<Написать письмо в компанию СПРУТ-Технология>** - подготовка электронного письма для службы технической поддержки АО “СПРУТ-Технология”. E-Mail: **support@sprut.ru**
  - **<О программе>** - информация о генераторе постпроцессоров.

1.2.2 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.

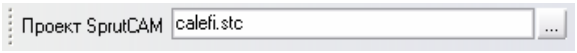
Панель установок

	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 подпрограммы.
	Выполнить следующую команду программы обработки технологических команд без захода в подпрограммы.
	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**.

1.2.3 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.

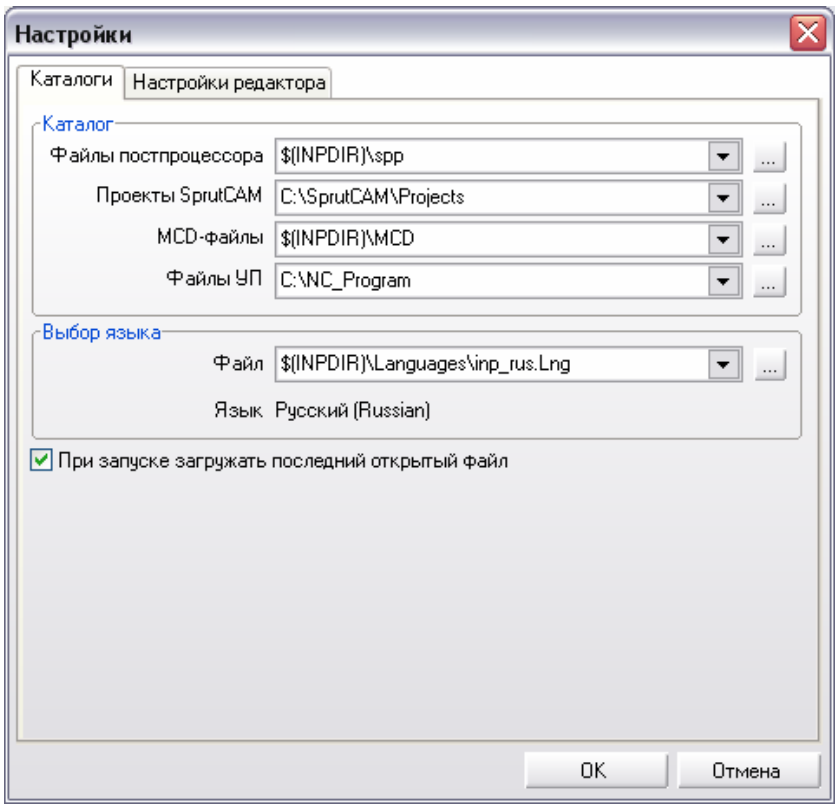
1.2.4 Системные установки

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 (***.SPP** файлы). Сохранение файлов постпроцессоров под новым именем по умолчанию производится в этом же каталоге.

Из каталога **Проекты SprutCAM** по умолчанию загружаются проекты системы **SprutCAM** (***.STC**). Связанные с открываемым проектом файлы технологических команд (***.MCD**) загружаются в соответствии с путями, указанными в этом проекте.

Из каталога **MCD-файлы** по умолчанию загружаются отдельные файлы технологических команд (***.MCD**).



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:

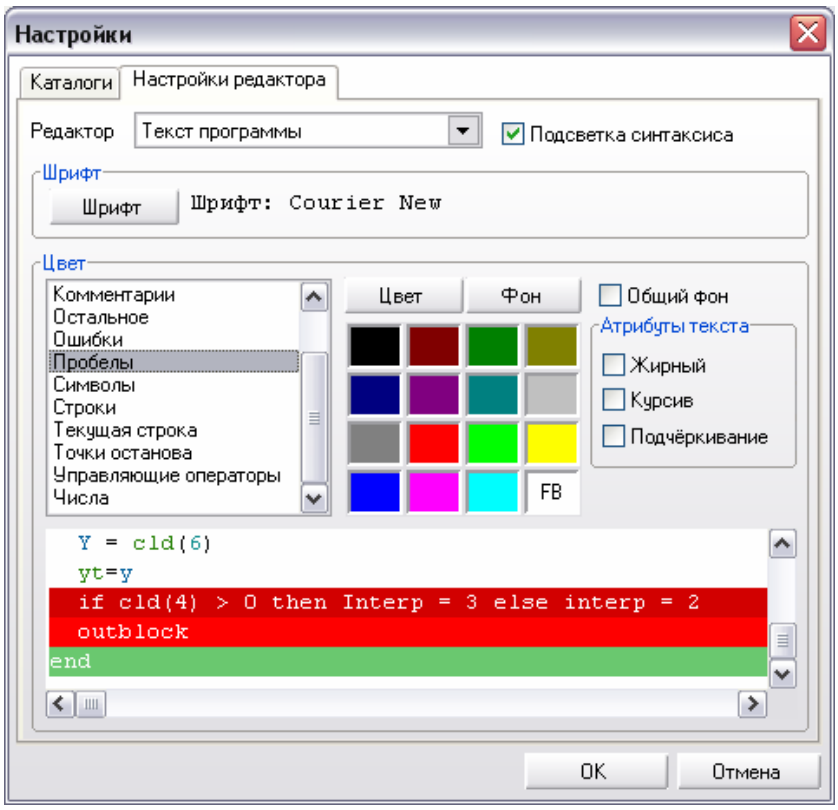
$(INPDIR) - каталог, из которого был запущен генератор постпроцессоров.

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.**

1.2.5 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.

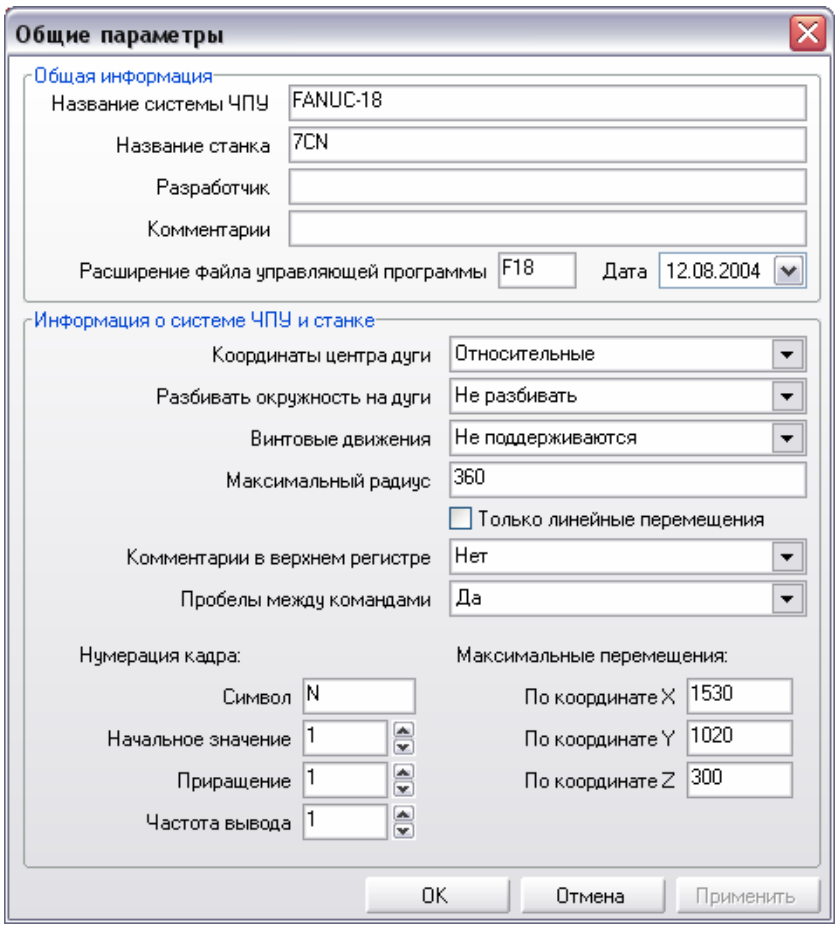
Для быстрого изменения цвета фона всех групп элементов существует флажок **Общий фон**. При установке флажка цвет фона всех групп заменяется на цвет фона текущего элемента.

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

1.2.6 Заполнение общих параметров

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

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



Общая информация

В верхней части страницы **Общая информация** вводятся название станка, название системы ЧПУ, 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

- 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**. Группа состоит из полей **Символ, Начальное значение, Приращение, Частота вывода.** В поле **Символ** вводится идентификатор регистра или символ, выводимый в кадр перед значением номера кадра. Кадры нумеруются, начиная с **начального значения**. После каждого вывода кадра к номеру кадра добавляется **приращение**. 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.

1.2.7 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:

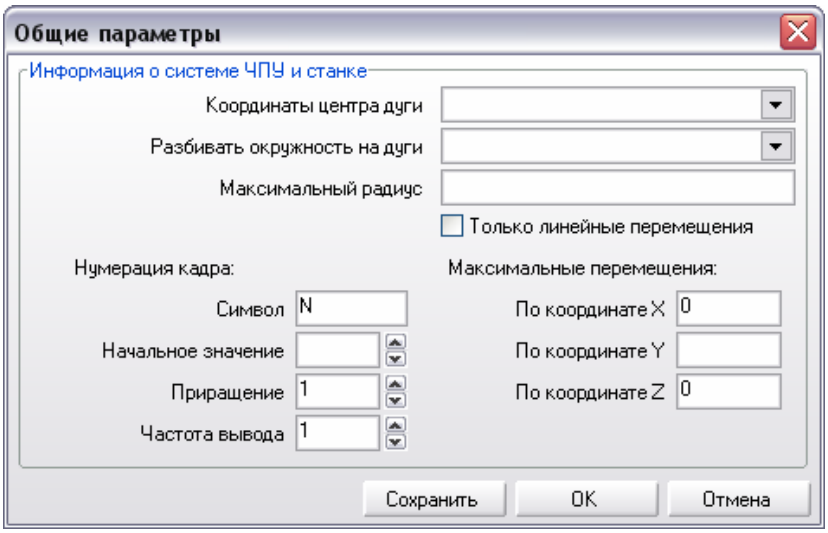
```
[Common definitions]
Spaces N           ! Y / N
CtrInc Y           ! Y / N
ArcDivision N      ! N / 90 / 180
Helical Y          ! Y / N
MaxRad 1000        !>= 0
LinearMovement N   ! Y / N
UpperCaseComment N ! Y / N
Sequence N 1 1 1   ! Symbol Frequence StepValue StartValue
Xmax 0             !>=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

где:

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.

- В первом случае в окно запроса информации добавится окно с выпадающим списком. Перед окном будет выведено "**Описание_параметра_или_запрос**". Выпадающий список будет состоять из значений: "**Описание_значения_1**", "**Описание_значения_2**", …, "**Описание_значения_N**". При выборе пользователем одного из пунктов соответствующее значение будет стартовым значением для переменной.
- Во втором случае в окно запроса информации добавится окно с полем для ввода данных. Перед окном будет выведено "**Описание_параметра_или_запрос**". Введённое значение будет стартовым значением для переменной.
- В третьем случае в окно запроса информации добавится окно с полем для ввода данных. Перед окном будет выведено "**Описание_параметра_или_запрос**". В поле для ввода данных будет занесено значение Значение_1. Введённое значение будет стартовым значением для переменной.

Значения строковых переменных должны быть заключены в кавычки.

Например, при подобном заполнении раздела:

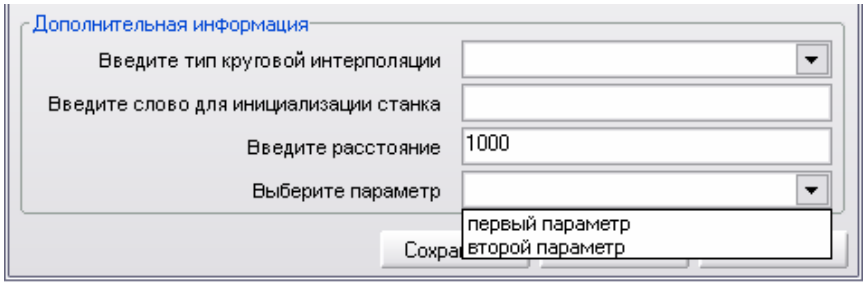
```
[Initial questions block]
Intrp  ?     "Введите тип круговой интерполяции"
       1     "IJ"
       2     "R"

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    "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.

1.2.8 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 **INTERPOL** 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 **INTERPOL** отслеживать текущий режим перемещений (позиционирование, линейная или круговая интерполяция) и в какой системе отсчета (абсолютной или относительной) задаются координаты.*

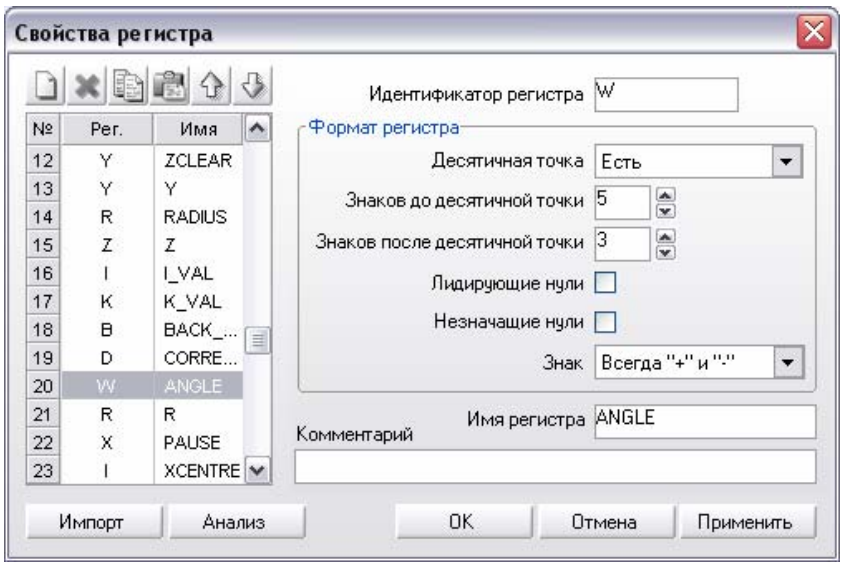
Список регистров отображается в левой части главного окна.

№	Рег.	Имя
1	N	PROGNUM
2	N	BLOCKN
3	G	PLANE
4	G	PIPEK
5	G	GFUNC
6	G	SYSTEM
7	U	SUBN
8	G	CYCLE
9	G	COMPFUN
10	G	APPROACH
11	X	X
12	Y	ZCLEAR
13	Y	Y
14	R	RADIUS

Определение свойств регистра

Окно редактирования свойств регистра открывается:

- Двойным нажатием левой клавиши мыши на строке регистра;
- при нажатии кнопки **<Свойства регистров>** на панели инструментов;
- выбором соответствующего пункта меню;



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 определяются свойства регистра:

- **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**;
  - **присутствует всегда**; выбор варианта “присутствует” значит, что точка будет стоять только тогда, когда значение регистра будет иметь дробную часть;
- **Знаков до десятичной точки** – максимальное количество знаков в целой части числа;
- **Знаков после десятичной точки** – количество знаков в дробной части числа;
- Наличие **лидирующих нулей** и наличие **незначащих нулей** – 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** ®.

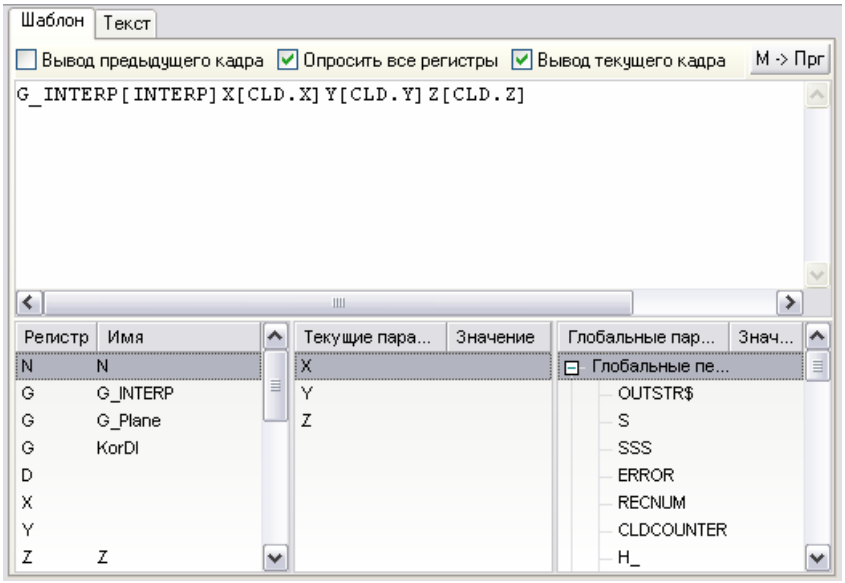
Для формирования списка регистров на основе управляющей программы предназначена кнопка **<Анализ>**. При этом анализируется текст указанной управляющей программы и все найденные адреса добавляются в список регистров.

При закрытии окна кнопкой <**Ok>** 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.

1.2.9 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 появиться в кадре и откуда должны быть подставлены значения, для этих адресов.

Редактор шаблонов расположен на закладке **Шаблоны**.

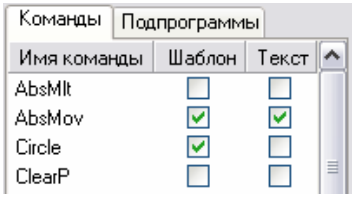


Для упрощения процесса создания шаблонов в нижней части окна находятся список регистров, список параметров текущей команды и список глобальных параметров процесса обработки.

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.

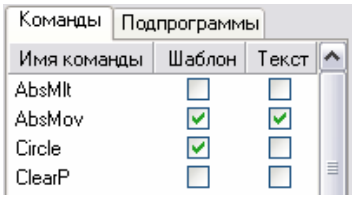


Правила формирования шаблонов подробно описаны в главе 2 .

1.2.10 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.



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 3.

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**.

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).

1.2.11 Подпрограммы

Подпрограммы пишутся на том же языке, что и программы, они вызываются из программ с помощью оператора “**CALL**”.

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.

Каждая подпрограмма начинается с заголовка, состоящего из слова “**SUB**” и имени программы, а заканчивается словом “**SUBEND**”.

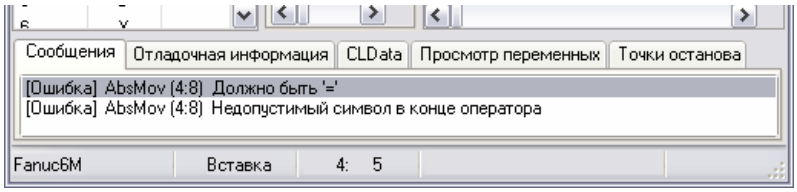
Имя подпрограммы может быть любым, не совпадающим с именами программ и уже имеющимися подпрограммами.

1.2.12 Трансляция программ обработки технологических команд

Трансляция программ обработки технологических команд производится нажатием кнопки на главной панели, выбором пункта главного меню <**Запуск>** <**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.

1.2.13 Работа с файлами технологических команд

Файлы технологических команд могут быть загружены из проекта системы **SprutCAM** (***.STC**), или как отдельные файлы операций (***.MCD**). Для загрузки необходимо нажать кнопку на главной

панели, или выбрать соответствующий пункт в главном или контекстном меню.

После загрузки поле **Проект SprutCAM** содержит имя проекта, из которого добавлены файлы операций.

В нижней 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.

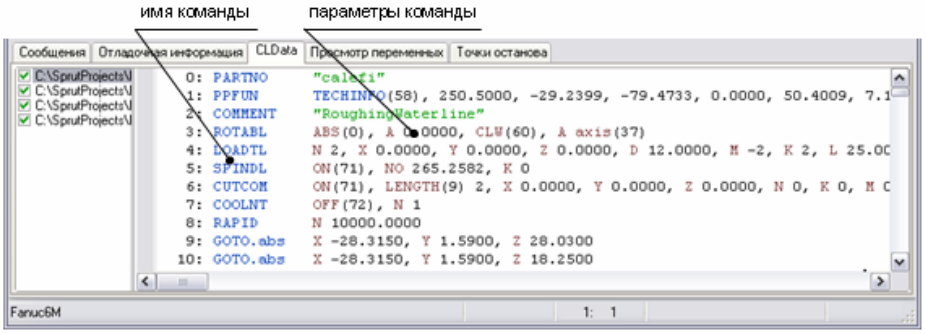
Список операций может быть отредактирован через контекстное меню, выпадающее при нажатии на правую кнопку мыши. Меню состоит из пунктов:

- **<Добавить>** - добавить файл технологических команд;
- **<Удалить>** - удалить выделенный файл.
- **<Удалить все>** - удалить все файлы.

В поле текста технологических команд загружается выбранная операция и представляется в терминах общепринятого единого промежуточного языка **CLDATA**.

Каждая технологическая команда в тексте **CLDATA** – отдельная строка, состоящая из имени команды и ее параметров.

Параметры доступны в шаблонах через список параметров текущей команды, а в программах ещё и через массив **CLD**.



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.

1.2.14 Тестовая генерация управляющих программ

Запуск тестовой генерации управляющей программы производится нажатием кнопки на главной панели, выбором пункта главного меню **<Запуск> <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.

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.

1.2.15 Отладка программ

Назначение встроенного отладчика 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.

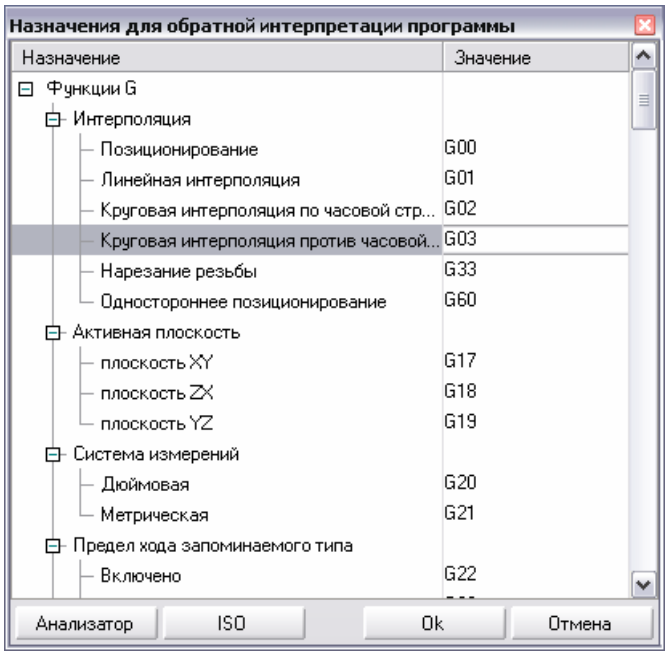
1.2.16 Обратная интерпретация программ

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 меню <**Вид**> <**Обратная интерпретация**>. Так же это окно можно вызвать с панели инструментов кнопкой **Назначения для обратной интерпретации команд**.

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.

2 ШАБЛОНЫ

2.1 ОСНОВНЫЕ ПОНЯТИЯ

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]
```

где **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.

2.2 СОСТАВ ШАБЛОНА

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

```
ЭЛЕМЕНТ1 ЭЛЕМЕНТ2 ... ЭЛЕМЕНТN
```

2.2.1 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:

1. все переменные подпрограммы **COMMON**;
2. 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;
3. предопределённые функции: FLAGIN, CROSS, NEXTTOOLNUM, CURDATE, CURTIME (подробнее см раздел “Предопределённые переменные”);
4. все параметры текущей технологической команды, передаваемые через массив CLD.

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

1. Десятичная точка. Может отсутствовать, присутствовать, в случае если число дробное или присутствовать всегда;

2. Максимальное количество знаков в целой части числа;

3. Количество знаков в дробной части числа;

4. Наличие лидирующих нулей и наличие незначащих нулей;

5. Знак значения регистра. Возможны варианты: без знака, только “-”, всегда “-” и “+”;

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

1. Лидирующие и незначащие нули отсутствуют;

2. Если число дробное, присутствует десятичная точка;

3. Если число отрицательное, присутствует знак;

4. Знаков до и после десятичной точки столько, сколько необходимо для полного отображения числа;

5. Идентификатор элемента выводится как текст.

2.2.2 Регистры в шаблонах

Для задания формата вывода значений, в соответствии с требованиями системы ЧПУ, используются регистры.

Понятие Регистра в постпроцессоре объединяет следующие свойства:

1. идентификатор регистра;

2. текущее значение;

3. предыдущее значение;

4. формат вывода значения в кадр;

5. имя регистра.

(Подробнее о регистрах см. раздел **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.

2.2.3 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.

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.

Шаблон: "G49G80M5M9 G_PLANE[17,Off]"

Кадр УП: "G49G80M5M9"

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

2.2.4 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 и вычитание
- умножение и 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.

2.2.5 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.

2.2.6 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 и 3 (X100.101 и 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:

Признак “Расставлять пробелы между командами“ отключен
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.

2.2.7 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$=”тест””
“G_INTERP[INTERP] X[XT] Y[YT] Z[ZT] ([CLDATA$])”
Кадр УП: “G0 X100.122 Y231.567 Z010.546 (тест)”

2.2.8 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_KORDL[43] X[XT] Y[YT]
G_KORDL[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_KORDL[43] X[XT] Y[YT]
G_KORDL[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_KORDL[43] X[XT] Y[YT]
G_KORDL[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.

2.3 УПРАВЛЕНИЕ ШАБЛОНОМ

2.3.1 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.

2.3.2 Преобразование шаблона в подпрограмму

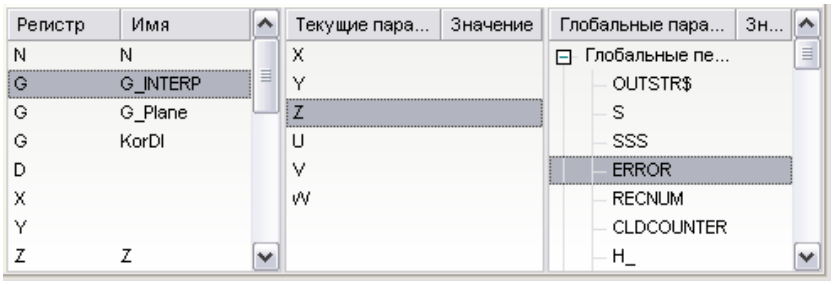
При редактировании шаблона может возникнуть ситуация, когда возможностей шаблона недостаточно. Для того чтобы использовать более мощный и гибкий механизм подпрограмм, без ручного переделывания шаблона, предназначена функция "**Преобразовать шаблон в подпрограмму**". Преобразование производится с помощью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

, которая находится на панели, рядом с переключателями шаблона.

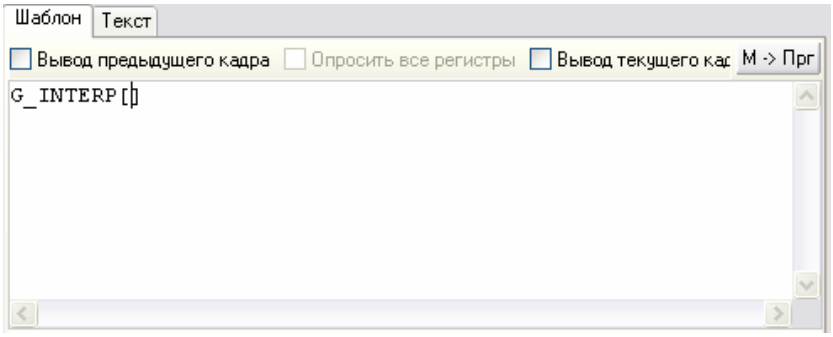
Для отмены преобразования необходимо перенести, используя буфер обмена, строку шаблона из редактора подпрограмм в редактор шаблонов и выставить соответствующие переключатели шаблона.

2.3.3 Визуальные средства формирования шаблона

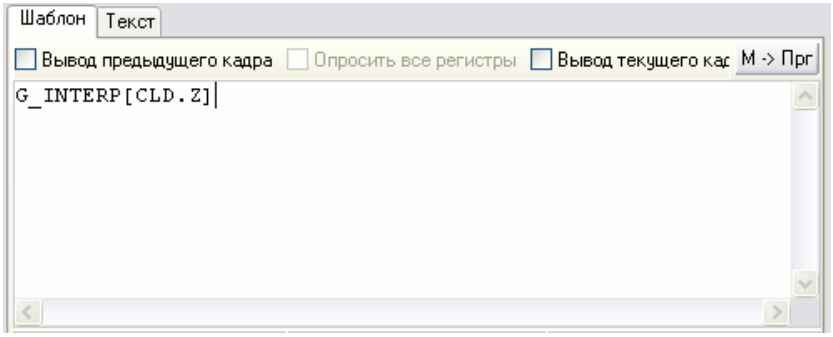
Для удобного формирования шаблона предназначены три списка, находящихся под редактором шаблонов: перечень регистров, перечень текущих параметров CLData и перечень глобальных переменных, предопределённых переменных и предопределённых функций.



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.



3 ОПИСАНИЕ ЯЗЫКА

3.1 ОСНОВНЫЕ ПОНЯТИЯ

3.1.1 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 |.

3.1.2 Программы обработки технологических команд, комментарии в программе

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**. Значения параметров для различных команд CLData описаны в главе 4.

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.

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.

3.1.3 Подпрограммы

Каждая подпрограмма начинается с заголовка, состоящего из слова " **SUB** " и имени программы, а заканчивается словом "**SUBEND**". Имя подпрограммы назначается пользователем, оно не должно совпадать с именами программ и существующих подпрограмм. Подпрограммы пишутся на таком же языке что и программы и так же могут содержать математические выражения и функции, операторы ввода/вывода, условные операторы, циклы, операторы перехода, вызовы других программ, операторы формирования кадров управляющей программы.

3.1.4 Понятие оператора языка

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

```
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.

3.1.5 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.

3.1.6 Переменные

Используемые в языке переменные делятся на три типа:

- **Целые**. Используется для представления целых чисел. Переменная целого типа имеет следующий диапазон значений: -2147483648 ÷ 2147483647.
- **Вещественные**. Предназначены для хранения чисел, имеющих дробную часть. Переменная вещественного типа имеет следующий диапазон значений: $\pm 2.9 \cdot 10^{-39} \div \pm 1.7 \cdot 10^{38}$ и имеет 11-12 значащих разрядов.
- **Строковые**. Представляют из себя последовательность символов длиной до $\sim 2^{31}$.

3.1.7 Массивы

Массив представляет собой структуру данных, позволяющую хранить под одним именем совокупность данных любого числового, но только одного какого-то типа. Массив характеризуется своим именем, типом хранимых элементов, размером (числом хранимых элементов) и нумерацией элементов. Массив одномерен.

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

Объявление переменной как массива имеет вид:

```
<имя массива>: array <размер> of <тип элемент>
```

Например:

```
V: array 10 of Real
V[5] = 10.67
i = 7
V[i] = 0.987
```

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.

Массив с заведомо известным количеством элементов называется **статический**. При объявлении размер массива можно опустить, такой массив называется **динамический**. Его размер определяется 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.

Например:

```
R: array of Integer
R[17] = 10
```

3.1.8 Математические выражения и функции

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 и вычитание
- умножение и деление
- возведение в степень

Необходимо помнить, что 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
- 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; где x, y - аргументы функции, представляющие собой число, переменную, математическую функцию или математическое выражение;
- 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;

Где,

n, m - числовые переменные или числа.

3.1.9 Предопределенные переменные

В языке используются следующие предопределенные переменные:

- **Регистры**. 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.
- Имеется предопределенный вещественный массив данных **CLD[n]**, где n - идентификатор элемента массива - целое положительное число. Идентификатор элемента массива n может сам являться переменной или выражением вещественного типа. При вызове программы обработки через массив CLD передаются параметры команды, считанные из файла технологических команд.
- Переменная **RecNum**, в которой хранится количество параметров в текущей технологической команде. То есть, при вызове программы обработки технологической команды в переменную заносится максимальный номер значащего элемента массива CLD. Использование переменной необходимо, прежде всего, в команде **PPFUN**, так как у нее не фиксировано количество записей.
- Переменная **CLDCounter** содержит номер текущей команды в файле технологических команд.

- 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$**. В переменной содержится текст текущего кадра управляющей программы после команд **FORMBLOCK** или **OUTBLOCK**. Тип - строковый.
- Переменные **Xt, Yt, Zt** - текущее положение инструмента. Тип – вещественный (переменные заполняются автоматически перед вызовом программ **CIRCLE, FROM, ABSMOV, INCMOV, ABSMLT, INCMLT, GOHOME**).
- Переменные **Xp, Yp, Zp** - предыдущее положение инструмента. Тип – вещественный (заполняются автоматически перед вызовом программ **CIRCLE, FROM, ABSMOV, INCMOV, ABSMLT, INCMLT, GOHOME**).
- Переменные **Xc, Yc, Zc** - координаты центра окружности (заполняются автоматически перед вызовом программы **CIRCLE**).
- Переменная **Interp** - текущая интерполяция, возможные значения: “0” – быстрый ход, “1” – линейная интерполяция, “2” – круговая интерполяция по часовой стрелке, “3” – круговая интерполяция против часовой стрелки; (значение заносится автоматически перед вызовом программ CIRCLE, FEDRAT, ABSMOV, INCMOV, ABSMLT, INCMLT, RAPID, GOHOME).
- Переменная **ToolRad** - радиус инструмента (значение заносится автоматически перед вызовом программы **LOADTL**).
- Переменная **Error** - целочисленная переменная. Если при выполнении программы произошли какие-либо ошибки, то номер ошибки содержится в переменой **ERROR**.
- Переменная **ArcPlane** - текущая плоскость круговой интерполяции, возможные значения: “33” – XY, “37” – YZ, “41” – XZ (значение заносится автоматически перед вызовом программы PLANE).
- Переменная **Feed** - текущее значение подачи (значение заносится автоматически перед вызовом программы FEDRAT).
- Переменные **TlComp, TrComp** - номер корректора на длину и на радиус соответственно (заполняется перед вызовом программы CUTCOM и LOADTL).
- Переменные **FromX, FromY, FromZ** - координаты исходной точки (заполняется перед вызовом программы FROM).
- Переменная **CurCode** – код текущей команды CLData (коды команд см. гл. 4 таблица 1).
- В строковой переменной **NCName$** содержится имя управляющей программы без расширения и без пути к файлу.

- Строковая переменная **NCPath$** содержит имя управляющей программы с расширением и полным путём к файлу.
- В строковой переменной **SPPName$** 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.
- Переменная **BlockStep** содержит значение приращения, используемого при формировании номера кадра.

3.1.10 Предопределённые функции

В языке используются следующие предопределенные функции:

- **NextCode** - возвращает код следующей технологической команды;
- **FlagIn** - возвращает значение “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

CurDate - возвращает строку, содержащую текущую дату;

CurTime - возвращает строку, содержащую текущее время.

GetCLDStr - возвращает строку текстового представления текущей команды CLData с параметрами.

3.2 ОПЕРАТОРЫ

3.2.1 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
```

3.2.2 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$.
```

3.2.3 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]
```

3.2.4 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. 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: ", Zt
INPUT "Введите название станка: ", NCNam$
INPUT "РАЗМЕРЫ ЗАГОТОВКИ", "Вдоль оси X", nx2, "Вдоль оси Y", ny2, "Высота по Z", nz2
```

3.2.5 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, т.е.

Формат:

```
<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, т.е.

Формат:

```
<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.

***Примечание**: Последовательность выполнения логических операций в операторе не может быть изменена указанием скобок. Для этого необходимо использовать последовательность из нескольких условных операторов. Таким образом, всегда можно заменить набор логических операций последовательностью логических операторов.*

***Примечание**: Когда условный оператор появляется внутри другого условного оператора, **ELSE**-часть соответствует ближайшему вверх по программе оператору **IF**. Как и все операторы, часть условного оператора может быть перенесена на другую строку.*

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"
```

3.2.6 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:

```
! Пример использования оператора case.
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
```

3.2.7 Оператор перехода на метку JUMP

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

Формат:

```
JUMP <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
```

3.2.8 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
```

3.2.9 Оператор цикла REPEAT

Структура repeat...until используется для организации циклического выполнения совокупности операторов, называемой телом цикла, до тех пор, пока не выполнится некоторое условие. Синтаксис управляющей структуры repeat...until:

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

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.

3.2.10 Оператор цикла WHILE

Структура while...do используется для организации циклического выполнения оператора, называемого телом цикла, пока выполняется некоторое условие. Синтаксис управляющей структуры while...do:

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

Структура работает следующим образом. Сначала вычисляется **<условие>**, 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.

3.2.11 Составной оператор

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:

```
! 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
```

3.2.12 Оператор вызова программы CALL

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

Формат:

```
CALL <имя программы>
```

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.

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

Пример:

```
CALL SUBTASK
```

3.2.13 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.

3.2.14 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.

Пример:

```
! пример подпрограммы
SUB EndCycle ! начало подпрограммы
  if Cld[1] = 163 then L = 81 else        ! Drill
  if Cld[1] = 168 then L = 84 else        ! Tap
  if Cld[1] = 209 then L = 85 else        ! Bore5
  if Cld[1] = 210 then begin              ! Bore6
    L = 86
    R4 = Spin
  end
  if Cld[1] = 213 then
   begin                                   ! Bore9
    if Cld[7] = 279.0 then R3 = Cld[8]
    L = 89
  end
SUBEND ! окончание подпрограммы
```

3.2.15 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. Пример работы с процедурой см. в разделе 3.2.16.

3.2.16 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
```

3.2.17 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
```

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). После вывода регистра в кадр вызывается указанная в регистре программа обработки, если таковая определена.

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

3.2.18 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**.

3.2.19 Оператор прямого вывода в кадр OUTPUT

Оператор выводит содержимое строковой переменной отдельным кадром управляющей программы.

Формат:

```
OUTPUT S$
```

Описание:

S$ - идентификатор строковой переменной.

Параметром оператора может служить также литерная константа или строковое выражение.

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
```

3.2.20 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, 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:

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

3.2.21 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:

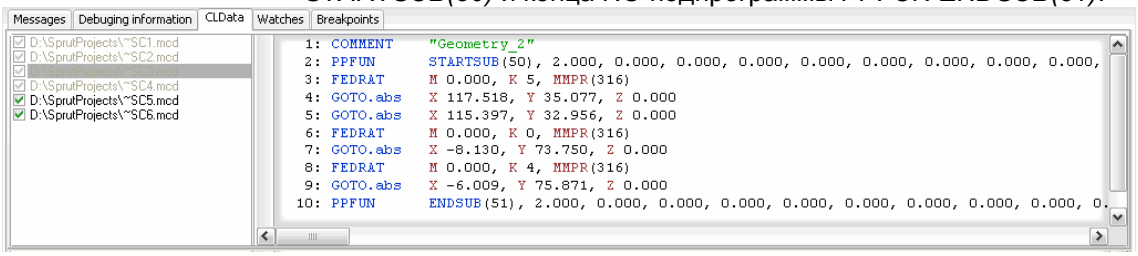
Оператор формирует кадр управляющей программы, в соответствии с правилами, описанными в строке шаблона. Опция OutBlock соответствует переключателю “Вывод текущего кадра”. А опция Poll соответственно переключателю “Опросить все регистры”. Подробнее о шаблонах см. гл. 2.

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
```

3.2.22 Операторы работы с NC-подпрограммами

NC-подпрограмма представляет собой последовательность технологических команд, оформленную в виде отдельного файла технологических команд ***.mcd**, 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 от обычных файлов ***.mcd** начинают обрабатываться автоматически, если файл помечен галочкой. 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.

3.2.23 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> } ) }
```

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.

Примеры:

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.

```
NCSub.Output          !Осуществляет трансляцию всех имеющихся
                      !NC-подпрограмм
NCSub.Output(3, 0)    !Осуществляет трансляцию NC-подпрограммы
                      !с номером 3 в режиме 0
NCSub.Output(CLD[2], 2)!Осуществляет трансляцию NC-подпрограммы
```

```
                        !с номером, извлекаемым из второго параметра
                        ! команды в режиме 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 <StartLabel>"
end
```

3.2.24 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.

Примеры:

```
St$: String
n1: Integer
St$ = "Example"
n1 = 3
NCSub.Name(n1)=St$+Str(n1) !Присвоение имени значения "Example3"
Output "o"+NCSub.Name(n1)  !Вывод в управляющую программу строки
                           !"oExample3"
```

3.2.25 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).

3.2.26 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).

3.2.27 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
    Output "o"+NCSub.Name(CLD[2])!Вывод строки, идентифицирующей
                                  !начало подпрограммы
                                  ! (здесь "oИмяПодпрограммы")
    NCSub.StartLabel(CLD[2]) = Str(BlockN) ! Присвоение начальной
                        !метке
  end                   ! текущего значения из регистра номера
                        !кадра
  else if CLD[1]=51 then begin ! Команда EndSub
    FormBlock             ! Вывод строки, идентифицирующей конец
    Output OutStr$ + " M99" ! подпрограммы (здесь "Nxx M99", где
                            ! Nxx - номер кадра
    NCSub.EndLabel(CLD[2]) = Str(BlockN) !Присвоение конечной
                            !метке
  end                       ! значения из регистра номера кадра

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

```
      NCSub.StartLabel(CLD[2])+ ! кадр, Nbb - кадр начала подпр.,
      ","+NCSub.EndLabel(CLD[2])! Nee - кадр конца подпрограммы
    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.

4 ПРИЛОЖЕНИЯ

4.1 СЛОВАРЬ СЛОВ CLDATA (ТЕХНОЛОГИЧЕСКИХ КОМАНД ПОДДЕРЖИВАЕМЫХ SPRUTCAM)

Таблица 1. Главные слова

N п/п	Слово	Код	Значение
1	CIRCLE	15000	Движение по дуге окружности
2	COEFF	2330	Коэффициенты сплайн-кривой
3	COMMENT	1065	Комментарий
4	COOLNT	1030	Охлаждение
5	CUTCOM	1007	Коррекция инструмента
6	CYCLE	1054	Цикл
7	DELAY	1010	Задержка
8	FEDRAT	1009	Подача
9	FINI	14000	Конечная запись
10	FROM	5003	Исходное положение
11	GOHOME	17	Возврат в исходное положение
12	GOTO	5004-5005	Перемещение инструмента
13	HEAD	1002	Номер шпиндельной головки
14	HEADER	3001	
15	INCR	66	Переход в абсолютную или относительную систему координат
16	INSERT	1046	Вывод непосредственно в кадр
17	LOADTL	1055	Загрузка инструмента
18	MOVNRB	3041	Начало сплайнового участка траектории
19	MULTGOTO	9004-9005	
20	OPSKIP	1012	Условный пропуск
21	OPSTOP	2003	Дополнительный останов
22	ORIGIN	1027	Начальные координаты
23	PALETA	1001	Смена палеты
24	PARTNO	1045	Номер заготовки
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	Безопасная позиция
31	SELCTL	1056	Выбор инструмента
32	SPINDL	1031	Шпиндель
33	STOP	2002	Останов
34	THREAD	1036	Нарезание резьбы
35	TRAILR	3004	

Таблица 2. Вспомогательные слова

N п/п	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

4.2 ФОРМАТ ТЕХНОЛОГИЧЕСКИХ КОМАНД

4.2.1 Номер детали

В программе обработки команды PARTNO формируется первый кадр управляющей программы. Программа обработки обязательно вызывается в начале генерации управляющей программы, даже в том случае если в файле технологических команд команда не содержится.

Команда:

```
PARTNO "........"
```

Переменная CLDATA$:

"........"

4.2.2 Конечная запись

Команда:

```
FINI
```

4.2.3 Постпроцессорная функция

Команда:

```
PPFUN PPFUN(500) | STARTSUB(50) | ENDSUB(51) | CALLSUB(52) |
REPSTART(53) | REPEND(54) | JUMP(55) | TECHINFO(58) {, a} {, b}
{, c} {, d} …
```

Параметры:

Параметр	CLD массив		Описание
PPFUN, или STARTSUB, или ENDSUB, или CALLSUB, или REPSTART, или REPEND, или JUMP или TECHINFO	CLD[1]	CLD.SubCode	идентификатор того, какая постпроцессорная функция вызывается
a, b, c, d …	CLD[2] – CLD[257]		список необязательных параметров. При вызове CALLSUB параметр "**a"** является обязательным и означает номер вызываемой подпрограммы

В случае, если CLD[1] = 500 (PPFUN), то CLD[2] принимает значение стойкости инструмента в минутах. Такая команда формируется, когда необходимо выполнить смену инструмента в связи с износом.

В случае, если CLD[1] = 58 (TECHINFO) необязательные параметры перечислены в таблице:

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	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 – inch)	
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	Инструмент
26	26	Номер инструмента	
27	27	Диаметр	
28	28	Длина рабочей части	
29	29	Угол рабочей части { угол конической части в радианах }	
30	30	Радиус торца { радиус скругления фрезы }	
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	

Массив CLD	Параметр	Описание переменной	Группа
35	35	Точка настройки (на кончик/на центр) 0 – Центральная точка 2 – Конечная точка	
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	Первая поворотная ось: 0 – отсутствует 1 – X, 2 – Y, 3 – Z, 4 – произвольная	Поворотные оси
48	48	Положение первой поворотной оси в текущей операции (в градусах)	
49	49	Вторая поворотная ось: 0 – отсутствует 1 – X, 2 – Y, 3 – Z, 4 – произвольная	
50	50	Угол второй поворотной оси в текущей операции (в градусах)	
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)	Охлаждение
60	60	Технологическая группа операции: 0 – не установлена, 1 - фрезерная, 2 - токарная.	Операция

4.2.4 Комментарий

Вывод комментария в управляющую программу.

Команда:

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

Переменная

```
CLDATA$:
".....”
```

4.2.5 Линейные перемещения

В зависимости от вида перемещения постпроцессор передает данные о перемещении либо в ABSMOV, либо в INCMOV, либо ABSMLT, либо INCMLT.

Команда:

```
GOTO.Abs X x, Y y, Z z
GOTO.Inc X x, Y y, Z z
MULTGOTO.Abs X x, Y y, Z z, V v, U u, W w
MULTGOTO.Inc X x, Y y, Z z, V v, U u, W w
```

Параметры:

x, y, z, u, v, w … – новые координаты инструмента (абсолютные или относительные)

Параметр	CLD массив		Описание
x, y, z, u, v, w	CLD[1] CLD[2] CLD[3] CLD[4] CLD[5] CLD[6]	CLD.X CLD.Y CLD.Z CLD.U CLD.V CLD.W	новые координаты инструмента (абсолютные или относительные)

4.2.6 Перемещение по окружности

Необходимо учитывать, в какой плоскости идет обработка, так как при разбиении окружности по четвертям или пополам постпроцессор берет необходимые данные из предопределенных переменных Xt, Yt, Zt, и кладет полученные координаты новой конечной точки в те координатные оси, в плоскости которых идет обработка. То есть, если обработка ведется в плоскости XZ, то необходимо обязательно отслеживать переменные Xt, Zt, а брать координаты центра в CLD(1) по X и CLD(3) по Z, а в CLD(5) и CLD(7) координаты конечной точки. Кроме того, в постпроцессоре радиус окружности, передаваемый в CLD(4), отрицательный в том случае, если это круговая интерполяция по часовой стрелке.

Команда:

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

Параметры:

Параметр	CLD массив		Описание
xc, yc	CLD[1] CLD[2]	CLD.Xc CLD.Yc	координаты X и Y центра окружности
z	CLD[3]	CLD.Zc	координата Z инструмента
r	CLD[4]	CLD.R	радиус окружности
xk, yk	CLD[5] CLD[6]	CLD.Xe CLD.Ye	координаты X и Y конечной точки дуги

Параметр	CLD массив		Описание
zk	CLD[7]	CLD.Ze	координата Z инструмента

4.2.7 Рабочая плоскость

В зависимости от рабочей плоскости процессор выводит перемещения в соответствующие координаты.

В постпроцессоре нужно иметь информацию о том, в какой плоскости идет обработка. Эти данные необходимы в модуле обработки окружности.

Команда:

```
PLANE XY(33)|YZ(37)|XZ(41)
```

Параметры:

Параметр	CLD массив		Описание
XY, или YZ или XZ	CLD[1]	CLD.Plane	код плоскости: 33 (XY), или 37 (YZ) или 41 (XZ)

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

4.2.8 Начальная точка

Задаются координаты исходной точки. Команда обычно встречается один раз и стоит в начале описания обработки детали. Если она отсутствует, то координаты точки исходного положения центра инструмента принимаются: X=0, Y=0, Z=0.

Если команда **FROM** присутствовала, то по команде GOHOME инструмент возвращается в заданную начальную точку.

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

4.2.9 Возврат в исходное положение

По этой команде инструмент перемещается в точку указанную оператором FROM или в точку с координатами X=0, Y=0, Z=0, но только на экране. В постпроцессор передается только код команды.

Команда:

```
GOHOME
```

4.2.10 Точка смены инструмента

Задаются координаты точки, в которую приводится центр инструмента для смены инструмента (команда LOADTL). По умолчанию смена производится в текущей точке.

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

4.2.11 Поворот стола

Команда:

```
ROTABL     ABS(0),     A     ab,     CCLW(59)|CLW(60),     PLANE
XY(33)|YZ(37)|XZ(41)
ROTABL     INCR(66),     B     ab,     CCLW(59)|CLW(60),     PLANE
XY(33)|YZ(37)|XZ(41)
```

Параметры:

Параметр	CLD массив		Описание
ABS или INCR	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 или clw	CLD[3]	CLD.CCLW	направление вращения против или по часовой стрелке
XY, или YZ или XZ	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	ISO B = – CLD[2]

Значение параметра CLD	Значение ISO
CLD[3] = 60	
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]

4.2.12 Компенсация 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 или OFF	CLD[1]	CLD.OnOff	71 – включение или 72 – выключение компенсации
LENGTH или R	CLD[2]	CLD.Length	9 – коррекция на длину или 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 или LEFT	CLD[10]	CLD.RGT	коррекция справа или слева

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

4.2.13 Загрузка инструмента

Используются данные о диаметре инструмента при обработке контура по эквидистанте. Для смены выводит инструмент в точку,

указанную ранее оператором SAFPOS. Если ранее оператора SAFPOS не было, то смена производится в текущих координатах.

Команда:

```
LOADTL N n, X x, Y y, Z z, D d, M m, K k, L l, P p, A a, R r,
PLANE XY(33)|YZ(37)|XZ(41), DURABILITY d2
```

Параметры:

Параметр	CLD массив		Описание
n	CLD[1]	CLD.N	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, k	CLD[6] CLD[7]	CLD.M CLD.K	номера корректоров
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
	CLD[12] CLD[13]		зарезервировано
XY, или YZ или XZ	CLD[14]		плоскость, которой перпендикулярна ось инструмента; 33 (XY), или 37 (YZ) или 41 (XZ)
	CLD[15] CLD[16]		зарезервировано
d2	CLD[17]	CLD.Dur	стойкость инструмента (в мин), 0 - не учитывать

4.2.14 Выбор инструмента

Команда:

```
SELCTL n
```

Параметры:

Параметр	CLD массив		Описание
N	CLD[1]	CLD.N	номер инструмента

4.2.15 Шпиндель

Используется частота вращения для проверки включен ли шпиндель в операторе **FEDRAT**, стандартных циклах, и в операторе выдержки для подсчета выдержки в оборотах.

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, или OFF, или	CLD[1]	CLD.OnOff	71 (ON) команда включения, 72 (OFF) выключения, 246

Параметр	CLD массив		Описание
ORIENT			(ORIENT) ориентированного останова.
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) или 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

4.2.16 Останов

Команда:

```
STOP
```

4.2.17 Дополнительный останов

Команда:

```
OPSTOP
```

4.2.18 Охлаждение

Команда:

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

Параметры:

Параметр	CLD массив		Описание
ON или OFF	CLD[1]	CLD.OnOff	включение или выключение охлаждения
n	CLD[2]	CLD.N	номер трубопровода

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

4.2.19 Подача

Сохраняется величина подачи до следующего оператора FEDRAT, или оператора перемещения с указанной подачей.

Команда:

```
FEDRAT N nm, K k, MMPM(315)
FEDRAT M nm, K k, MMPR(316)
FEDRAT NM nm, K k, p
```

Параметры:

Параметр	CLD массив		Описание
nm	CLD[1]	CLD.NM	подача в мм/мин, или в мм/об или других единицах
k	CLD[2]	CLD.K	диапазон подач
MMPM, MMPR, p	CLD[3]	CLD.MMPM	единицы измерения подачи (мм/мин или мм/об) или параметр, отвечающий за единицы задания подачи

4.2.20 Быстрый ход

Команда:

```
RAPID N n
```

Параметры:

Параметр	CLD массив		Описание
n	CLD[1]	CLD.N	величина ускоренной подачи

4.2.21 Выстой

Команда:

```
DELAY A a
```

Параметры:

Параметр	CLD массив		Описание
a	CLD[1]	CLD.A	время выстоя в секундах

4.2.22 Переход в абсолютную или относительную систему координат

После данного оператора перемещения по X, Y, Z выдаются в абсолютной или относительной системе координат.

Команда:

```
INCR ON(71)|OFF(72)
```

Параметры:

Параметр	CLD массив		Описание
ON или OFF	CLD[1]	CLD.OnOff	относительная или абсолютная система координат

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

Значение параметра CLD	Значение ISO
CLD[1] = 71	ISO.G = 91
CLD[1] = 72	ISO.G = 90

4.2.23 Начальные координаты

После определения смещения нулевой точки технологический процессор начинает пересчитывать все перемещения на величину смещения.

Команда:

```
ORIGIN X x, Y y, Z z, PPFUN f, N n
```

Параметры:

Параметр	CLD массив		Описание
X, y, z	CLD[1] CLD[2] CLD[3]	CLD.X CLD.Y CLD.Z	координаты смещения нулевой точки
F	CLD[4]	CLD.PPFun	номер постпроцессорной функции
n	CLD[5]	CLD.N	номер нулевой точки

4.2.24 Отмена и восстановление цикла

Все стандартные циклы G81-G92 попадают в запись CYCLE, поэтому необходимо в модуле произвести разбор вида цикла и описание их обработки.

Команда:

```
CYCLE ON (71) | OFF (72)
```

Параметры:

Параметр	CLD массив		Описание
Type	CLD[1]	CLD.Type	71 (ON) или 72 (OFF) - восстановление или отмена цикла

4.2.25 Цикл сверления типа G81

Команда:

```
CYCLE DRILL(163), A a, MMPM(315), N nm, F f, P p, T t
CYCLE DRILL(163), A a, MMPR(316), M nm, F f, P p, T t
```

Параметры:

Параметр	CLD массив		Описание
Type	CLD[1]	CLD.Type	тип цикла 163 (DRILL)
a	CLD[2]	CLD.A	глубина отверстия в мм
MMPM, MMPR	CLD[3]	CLD. MMPM	единицы измерения подачи (мм/мин или мм/об)
nm	CLD[4]	CLD.NM	подача в мм/мин или в мм/об

Параметр	CLD массив		Описание
f	CLD[5]	CLD.F	безопасный уровень в мм
	CLD[6] CLD[7]		зарезервировано
p	CLD[8]	CLD.P	уровень обратного хода в мм
	CLD[9] CLD[10]		зарезервировано
t	CLD[11]	CLD.Top	верхний уровень отверстия

4.2.26 Цикл сверления типа G82

Команда:

```
CYCLE FACE(81), A a, MMPM(315), N nm, F f, P p, DWELL(279) h, T
t
CYCLE FACE(81), A a, MMPR(316), M nm, F f, P p, DWELL(279) h, T
t
```

Параметры:

Параметр	CLD массив		Описание
Type	CLD[1]	CLD.Type	тип цикла 81 (FACE)
a	CLD[2]	CLD.A	глубина отверстия в мм
MMPM, MMPR	CLD[3]	CLD. MMPM	единицы измерения подачи (мм/мин или мм/об)
nm	CLD[4]	CLD.NM	подача в мм/мин или в мм/об
f	CLD[5]	CLD.F	безопасный уровень в мм
	CLD[6] CLD[7]		зарезервировано
p	CLD[8]	CLD.P	уровень обратного хода в мм
DWELL	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

4.2.27 Цикл сверления типа G84

Команда:

```
CYCLE TAP(168), A a, MMPM(315), N nm, F f, P p, T t, S s, PS ps,
SC sc, FR fr
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
Type	CLD[1]	CLD.Type	тип цикла 168 (TAP)
a	CLD[2]	CLD.A	глубина отверстия в мм
MMPM, MMPR	CLD[3]	CLD. MMPM	единицы измерения подачи (мм/мин или мм/об)
nm	CLD[4]	CLD.NM	подача в мм/мин или в мм/об
f	CLD[5]	CLD.F	безопасный уровень в мм
	CLD[6] CLD[7]		зарезервировано

Параметр	CLD массив		Описание
p	CLD[8]	CLD.P	уровень обратного хода в мм
	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	позиция шпинделя
sc	CLD[14]	CLD.Socket	тип патрона (0 - плавающий, 1 - жесткий)
fr	CLD[15]	CLD.FR	подача возврата

4.2.28 Цикл сверления типа G85

Команда:

```
CYCLE BORE5(209), A a, MMPM(315), N nm, F f, P p, T t
CYCLE BORE5(209), A a, MMPR(316), M nm, F f, P p, T t
```

Параметры:

Параметр	CLD массив		Описание
Type	CLD[1]	CLD.Type	тип цикла 209 (BORE5)
a	CLD[2]	CLD.A	глубина отверстия в мм
MMPM, MMPR	CLD[3]	CLD. MMPM	единицы измерения подачи (мм/мин или мм/об)
nm	CLD[4]	CLD.NM	подача в мм/мин или в мм/об
f	CLD[5]	CLD.F	безопасный уровень в мм
	CLD[6] CLD[7]		зарезервировано
p	CLD[8]	CLD.P	уровень обратного хода в мм
	CLD[9] CLD[10]		зарезервировано
T	CLD[11]	CLD.Top	верхний уровень отверстия

4.2.29 Цикл сверления типа G86

Команда:

```
CYCLE BORE6(210), A a, MMPM(315), N nm, F f, P p, T t
CYCLE BORE6(210), A a, MMPR(316), M nm, F f, P p, T t
```

Параметры:

Параметр	CLD массив		Описание
Type	CLD[1]	CLD.Type	тип цикла 210 (BORE6)
a	CLD[2]	CLD.A	глубина отверстия в мм
MMPM, MMPR	CLD[3]	CLD. MMPM	единицы измерения подачи (мм/мин или мм/об)
nm	CLD[4]	CLD.NM	подача в мм/мин или в мм/об
f	CLD[5]	CLD.F	безопасный уровень в мм
	CLD[6] CLD[7]		зарезервировано
p	CLD[8]	CLD.P	уровень обратного хода в мм
	CLD[9]		зарезервировано

Параметр	CLD массив		Описание
	CLD[10]		
t	CLD[11]	CLD.Top	верхний уровень отверстия

4.2.30 Цикл сверления типа G87

Команда:

```
CYCLE BORE7(211), A a, MMPM(315), N nm, F f, P p, T t
CYCLE BORE7(211), A a, MMPR(316), M nm, F f, P p, T t
```

Параметры:

Параметр	CLD массив		Описание
Type	CLD[1]	CLD.Type	тип цикла 211 (BORE7)
a	CLD[2]	CLD.A	глубина отверстия в мм
MMPM, MMPR	CLD[3]	CLD. MMPM	единицы измерения подачи (мм/мин или мм/об)
nm	CLD[4]	CLD.NM	подача в мм/мин или в мм/об
f	CLD[5]	CLD.F	безопасный уровень в мм
	CLD[6] CLD[7]		зарезервировано
p	CLD[8]	CLD.P	уровень обратного хода в мм
	CLD[9] CLD[10]		зарезервировано
t	CLD[11]	CLD.Top	верхний уровень отверстия

4.2.31 Цикл сверления типа G88

Команда:

```
CYCLE BORE8(212), A a, MMPM(315), N nm, F f, P p, DWELL(279) h,
T t
CYCLE BORE8(212), A a, MMPR(316), M nm, F f, P p, DWELL(279) h,
T t
```

Параметры:

Параметр	CLD массив		Описание
Type	CLD[1]	CLD.Type	тип цикла 212 (BORE8)
a	CLD[2]	CLD.A	глубина отверстия в мм
MMPM, MMPR	CLD[3]	CLD. MMPM	единицы измерения подачи (мм/мин или мм/об)
nm	CLD[4]	CLD.NM	подача в мм/мин или в мм/об
f	CLD[5]	CLD.F	безопасный уровень в мм
	CLD[6] CLD[7]		зарезервировано
p	CLD[8]	CLD.P	уровень обратного хода в мм
DWELL	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

4.2.32 Цикл сверления типа G89

Команда:

```
CYCLE BORE9(213), A a, MMPM(315), N nm, F f, P p, DWELL(279) h,
T t
CYCLE BORE9(213), A a, MMPR(316), M nm, F f, P p, DWELL(279) h,
T t
```

Параметры:

Параметр	CLD массив		Описание
Type	CLD[1]	CLD.Type	тип цикла 213 (BORE9)
a	CLD[2]	CLD.A	глубина отверстия в мм
MMPM, MMPR	CLD[3]	CLD. MMPM	единицы измерения подачи (мм/мин или мм/об)
nm	CLD[4]	CLD.NM	подача в мм/мин или в мм/об
f	CLD[5]	CLD.F	безопасный уровень в мм
	CLD[6] CLD[7]		зарезервировано
p	CLD[8]	CLD.P	уровень обратного хода в мм
DWELL	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

4.2.33 Глубокое сверление с полным выводом инструмента для удаления стружки

Команда:

```
CYCLE DEEP(153), A a, MMPM(315), N nm, F f, L l, I i, P p, T t
CYCLE DEEP(153), A a, MMPR(316), M nm, F f, L l, I i, P p, T t
```

Параметры:

Параметр	CLD массив		Описание
Type	CLD[1]	CLD.Type	тип цикла 153 (DEEP)
a	CLD[2]	CLD.A	глубина отверстия в мм
MMPM, MMPR	CLD[3]	CLD. MMPM	единицы измерения подачи (мм/мин или мм/об)
nm	CLD[4]	CLD.NM	подача в мм/мин или в мм/об
f	CLD[5]	CLD.F	безопасный уровень в мм
l	CLD[6]	CLD.L	глубина первого захода в мм
i	CLD[7]	CLD.I	переходная величина
p	CLD[8]	CLD.P	уровень обратного хода в мм
	CLD[9] CLD[10]		зарезервировано
t	CLD[11]	CLD.Top	верхний уровень отверстия

4.2.34 Глубокое сверление с отводами сверла для ломки стружки

Команда:

```
CYCLE BRKCHP(288), A a, MMPM(315), N nm, F f, L l, I i, P p,
DWELL(279) h, T t
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
Type	CLD[1]	CLD.Type	тип цикла 288 (BRKCHP)
a	CLD[2]	CLD.A	глубина отверстия в мм;
MMPM, MMPR	CLD[3]	CLD. MMPM	единицы измерения подачи (мм/мин или мм/об)
nm	CLD[4]	CLD.NM	подача в мм/мин или в мм/об
f	CLD[5]	CLD.F	безопасный уровень в мм
l	CLD[6]	CLD.L	глубина первого захода в мм
i	CLD[7]	CLD.I	переходная величина
p	CLD[8]	CLD.P	уровень обратного хода в мм
DWELL	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

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

Значение параметра CLD	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

4.2.35 Нарезание резьбы

Команда:

```
THREAD MMPR m, TPI t, MulTrd mt, Depth d, Cuts c, FinCut f, Ang
a, Oset1 o1, Oset2 o2
```

Параметры:

Параметр	CLD массив		Описание
m	CLD[1]	CLD. MMPR	WMmpr(316)/WPitch(1050)
t	CLD[2]	CLD.TPI	шаг резьбы/число ниток на дюйм
mt	CLD[3]	CLD. MulTrd	число заходов
d	CLD[4]	CLD. Depth	глубина
c	CLD[5]	CLD. Cuts	кол-во проходов (всего)
f	CLD[6]	CLD. FinCut	кол-во чистовых проходов
a	CLD[7]	CLD. Ang	угол врезания
o1, o2	CLD[8] CLD[9]	CLD. Oset1 CLD. Oset2	№№ корректоров, включаемых попеременно в каждом проходе

4.2.36 Смена палеты

Команда:

```
PALETA N n
```

Параметры:

Параметр	CLD массив		Описание
N	CLD[1]	CLD.N	номер палеты

4.2.37 Головка

Команда:

```
HEAD BOTH(83)|N n
```

Параметры:

Параметр	CLD массив		Описание
n или BOTH	CLD[1]	CLD.N	n - номер требуемой головки или 83 (BOTH) - одновременная работа двух головок

4.2.38 Коэффициенты сплайн-кривой

Промежуточные контрольные точки NURBS-кривой. Эти команды идут после инициализации NURBS-кривой (команда MovNRB), последняя точка кривой задаётся командой GOTO.

Команда:

```
COEFF X x, Y y, Z z, Knout k, Fedrat f, Spindl s, Denom d
```

Параметры:

Параметр	CLD массив		Описание
x, y, z	CLD[1] CLD[2] CLD[3]	CLD.X CLD.Y CLD.Z	координаты контрольной точки
k	CLD[4]	CLD.Knout	кнот (узел) NURBS кривой
f	CLD[5]	CLD.Fedrat	коэффициент подачи
s	CLD[6]	CLD.Spindl	коэффициент скорости вращения шпинделя
d	CLD[7]	CLD.Denom	коэффициент знаменателя NURBS кривой

4.2.39 Начало сплайнового участка траектории

Начало сплайнового участка траектории.

Команда:

```
MOVNRB Order
```

Параметры:

Параметр	CLD массив		Описание
Order	CLD[1]	CLD.Order	порядок NURBS'a (степень + 1)

4.2.40 Вставка

Текст попадает в переменную CLDATA$, после чего может быть выдан в УП.

Команда:

```
INSERT "........"
```

Переменная CLDATA$:

```
"........"
```

4.2.41 Условный пропуск

Команда:

```
OPSKIP ON(71)|OFF(72)
```

Параметры:

Параметр	CLD массив		Описание
ON или OFF	CLD[1]	CLD.OnOff	71 (ON) или 72 (OFF) включение или отключение условного пропуска

4.2.42 Печать постпроцессора

Текст попадает в переменную CLDATA$, после чего может быть выдан в окно оператора как комментарий, поясняющий действия постпроцессора.

Команда:

```
PPRINT "........"
```

Переменная CLDATA$:

```
"........"
```

4.2.43 Расширенный цикл

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).
- Цикл чернового точения (G71, G72).
- Цикл нарезания канавок (G74,G75).
- Цикл нарезания резьбы в несколько проходов (G76).
- Цикл сверления типа G81.
- Цикл сверления типа G82.
- Цикл нарезания резьбы осевым инструментом типа G84.

- Цикл глубокого сверления с отводами сверла для ломки стружки.
- Цикл глубокого сверления с полным выводом инструмента для удаления стружки.

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) {, a}
{, b} {, c} {, d} ...
```

Параметры:

Параметр	CLD массив		Описание
ON(71), CALL(52) или OFF(72)	CLD[1]	CLD. SubCmd	Модификатор включения стандартного цикла ON(71), вызова стандартного цикла CALL(52) и отмены стандартного цикла OFF(72).
LATHEFINISH (400), LATHEROUGH(401), LATHEGROOVE(402), LATHETHREAD(403), DRILL(163), FACE(81), TAP(168), BRKCHP(288), DEEP(153)	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:

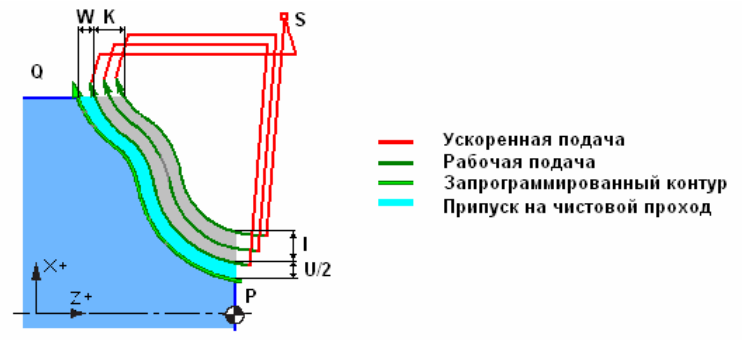
```
EXTCYCLE ON(71) ...
FEDRAT ...
GOTO ...
GOTO ...
EXTCYCLE CALL(52) ...
FEDRAT ...
GOTO ...
GOTO ...
EXTCYCLE CALL(52) ...
FEDRAT ...
GOTO ...
GOTO ...
EXTCYCLE CALL(52) ...
EXTCYCLE OFF(72) ...
```

В зависимости от конкретного типа цикла меняется набор последующих параметров.

LATHEFINISH(400) – Цикл чистового точения (G70, G73)

CLD массив		Описание
CLD[3]	CLD.CLParams(1)	Номер NC-подпрограммы, содержащей обрабатываемый контур PQ.

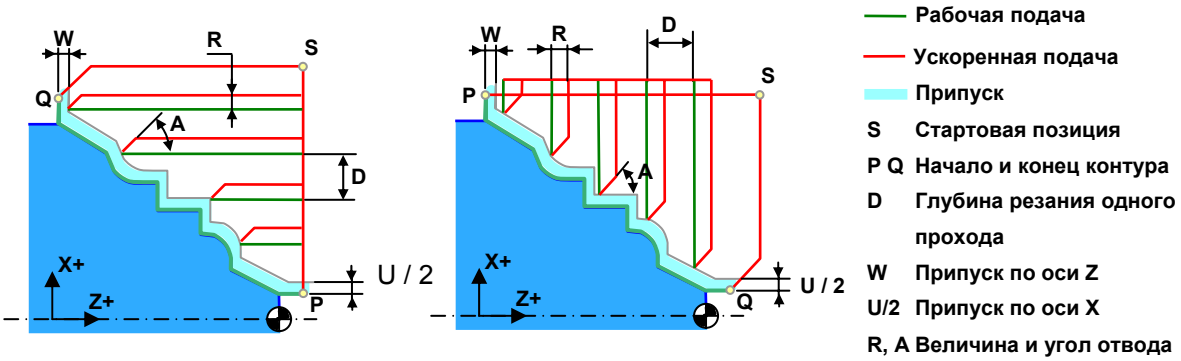
CLD массив		Описание
CLD[4]	CLD.CLParams(2)	Количество проходов (D). Если обработка производится за один проход D=1, имеет место цикл G70. Если D>1, то имеет место цикл G73.
CLD[5]	CLD.CLParams(3)	Знаковая толщина снимаемого слоя по оси Z (K). Для цикла G71 равна нулю.
CLD[6]	CLD.CLParams(4)	Знаковая 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.



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.

LATHEROUGH(401) – Цикл чернового точения (G71, G72)

CLD массив		Описание
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)

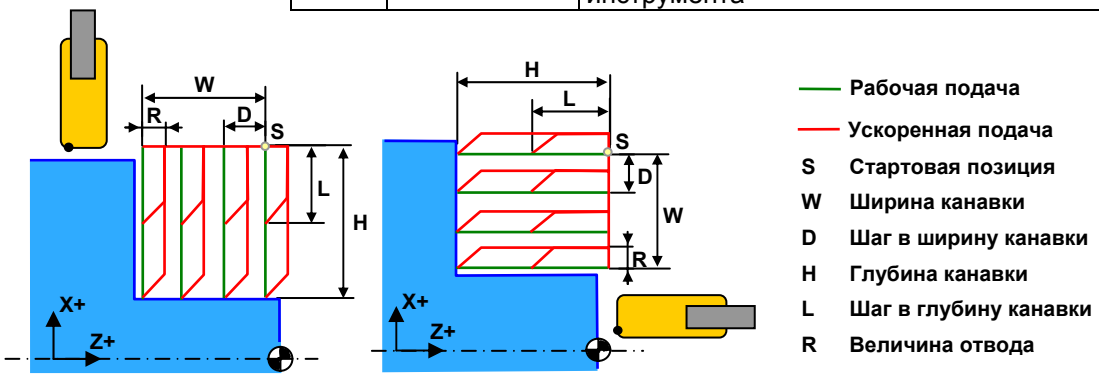


Приставка «знаковая» для величины означает, что если величина больше нуля, то она откладывается в положительном

направлении соответствующей оси, а если меньше нуля – в отрицательном направлении оси.

LATHEGROOVE(402) – Цикл нарезания канавок (G74,G75)

CLD массив		Описание
CLD[3]	CLD.CLParams(1)	Направление канавки: «0» - цилиндрическая (G75), «1» - торцевая (G74)
CLD[4]	CLD.CLParams(2)	Знаковая ширина канавки относительно стартовой позиции
CLD[5]	CLD.CLParams(3)	Знаковая глубина канавки относительно стартовой позиции
CLD[6]	CLD.CLParams(4)	Абсолютное значение шага в ширину
CLD[7]	CLD.CLParams(5)	Абсолютное значение шага в глубину (глубина для ломки стружки)
CLD[8]	CLD.CLParams(6)	Абсолютное значение отскока при выводе инструмента

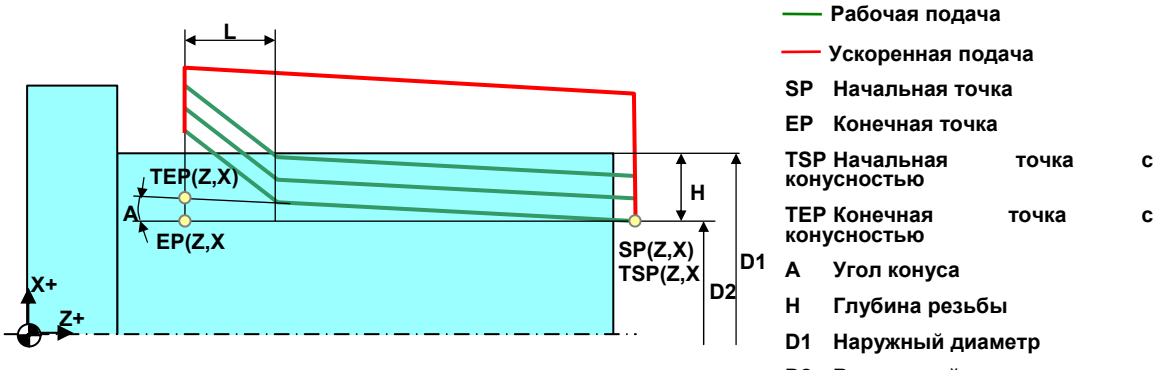


Приставка «знаковая» для величины означает, что если величина больше нуля, то она откладывается в положительном направлении соответствующей оси, а если меньше нуля – в отрицательном направлении оси.

LATHETHREAD(403) – Цикл нарезания резьбы в несколько проходов (G76)

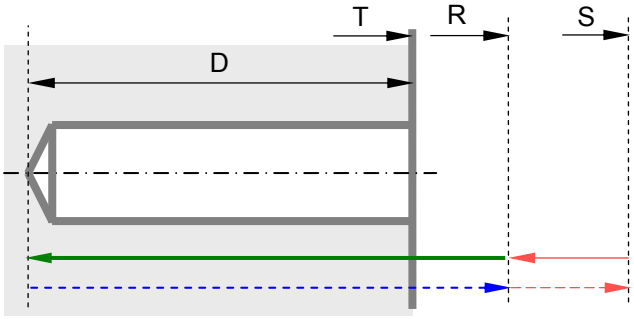
CLD массив		Описание
CLD[3]	CLD.CLParams(1)	Направление резьбы: «0» - левая, «1» - правая.
CLD[4]	CLD.CLParams(2)	Ориентация резьбы: «0» - наружная, «1» - внутренняя, «2» - торцевая.
CLD[5]	CLD.CLParams(3)	Угол конуса в градусах (для конической резьбы) (A).
CLD[6]	CLD.CLParams(4)	Координата Z начальной 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 массив		Описание
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[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.



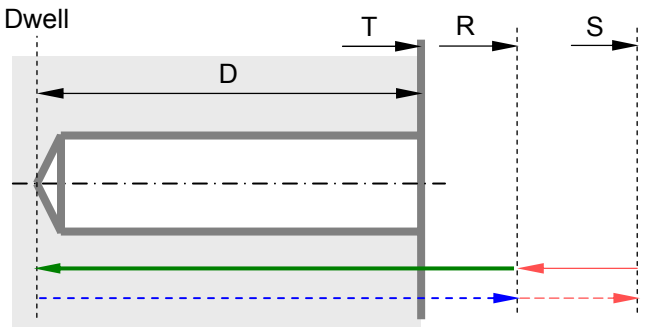
DRILL(163) – Цикл сверления типа G81

CLD массив		Описание
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, «1» - мм/мин.
CLD[9]	CLD.CLParams(7)	Рабочая подача.
CLD[10]	CLD.CLParams(8)	Подача возврата.



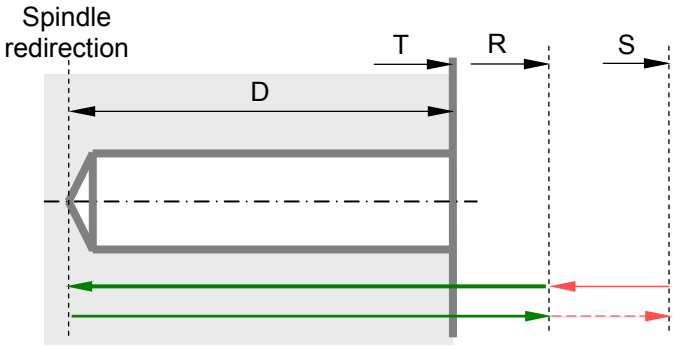
FACE(81) – Цикл сверления типа G82

CLD массив		Описание
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, «1» - мм/мин.
CLD[9]	CLD.CLParams(7)	Рабочая подача.
CLD[10]	CLD.CLParams(8)	Подача возврата.



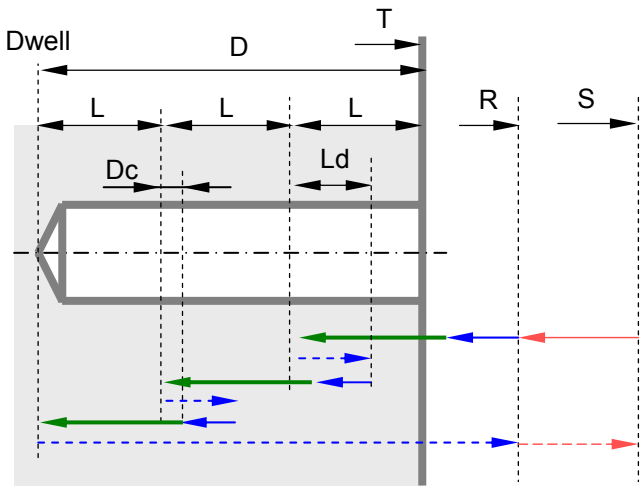
TAP(168) – Цикл нарезания резьбы осевым инструментом типа G84

CLD массив		Описание
CLD[3]	CLD.CLParams(1)	Верхний уровень отверстия.
CLD[4]	CLD.CLParams(2)	Глубина отверстия относительно верхнего уровня.
CLD[5]	CLD.CLParams(3)	Уровень возврата.
CLD[6]	CLD.CLParams(4)	Уровень безопасной плоскости.
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.



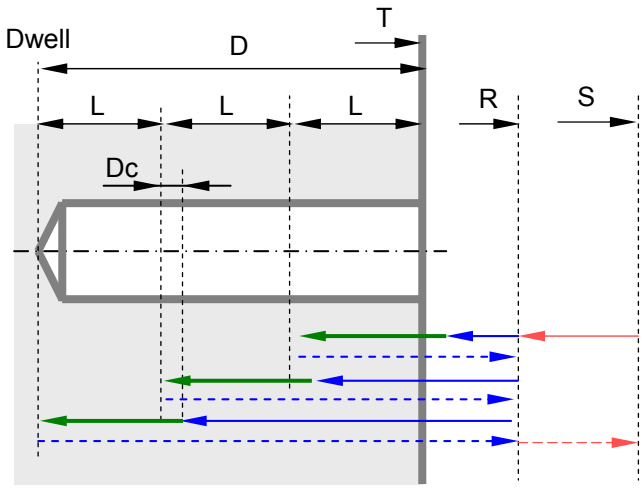
BRKCHP(288) – Цикл глубокого сверления с отводами сверла для ломки стружки

CLD массив		Описание
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, «1» - мм/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



DEEP(153) – Цикл глубокого сверления с полным выводом инструмента для удаления стружки

CLD массив		Описание
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, «1» - мм/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



4.2.44 Нарезание резьбы за один проход

Команда SINGLETHREAD реализует один проход при нарезании резьбы.

Команда:

```
SINGLETHREAD OD(0)|ID(1)|FACE(2), STEP(0)|COUNT(1), s, ANGLE a
```

Параметры:

Параметр	CLD массив		Описание
OD(0), ID(1) или FACE(2))	CLD[1]	CLD. Orientation	Ориентация резьбы: OD(0) - наружная, ID(1) - внутренняя, FACE(2) - торцевая.
STEP(0) или COUNT(1)	CLD[2]	CLD. IsCount	Способ задания шага резьбы: STEP(0) - расстоянием, COUNT(1) – количеством витков на единицу длины.
s	CLD[3]	CLD. Value	Значение шага резьбы.
a	CLD[4]	CLD. StartAngle	Начальное угловое положение шпинделя (используется для многозаходной резьбы).