Разработка постпроцессоров

Постпроцессоры преобразуют нейтральный поток команд CLData, сформированный CAM-системой, в управляющую программу для станка с ЧПУ. Руководство описывает две независимые системы постпроцессирования и общую модель входных данных.

Основные разделы

- SPPX-постпроцессоры — разработка на встроенном Pascal-подобном языке в Генераторе постпроцессоров.
- .NET-постпроцессоры — разработка .NET-сборок с помощью SDK постпроцессирования.
- CLData — справочник команд траектории и технологических данных, общих для обеих систем.
- Разработка с помощью ИИ — настройка инструментов, организация работы и проверка результатов.

Генератор постпроцессоров (sppx-постпроцессоры)

Генератор постпроцессоров — это подсистема построения постпроцессоров на встроенном Pascal-подобном языке — `.sppx`-постпроцессоров. В этом разделе описаны сам генератор, его модель шаблонов и регистров, а также язык. Внешние ссылки и примеры перечислены в разделе справочные материалы.

- Внешние справочные материалы и примеры
- Введение
  - Назначение генератора постпроцессоров
  - Набор файлов генератора постпроцессоров
- Общая организация работы
  - Принцип работы постпроцессора
  - Главное окно
    - Главная панель инструментов
    - Системные установки
    - Настройка редактора
    - Параметры станка
      - Определение данных о станке и системе ЧПУ
      - Транслитерация комментариев
    - Определение структуры и формата кадра (формирование списка регистров)
    - Шаблоны трансляции технологических команд
    - Программы обработки команд CLData
    - Подпрограммы
    - Трансляция программ обработки команд
    - Работа с файлами технологических команд
    - Тестовая генерация управляющих программ
    - Отладка программ
    - Запрос параметров постпроцессора при первом использовании
- Шаблоны постпроцессора
  - Структура шаблона
    - Элемент шаблона
    - Регистры в шаблонах
    - Модификаторы
    - Выражения
    - Вложенный шаблон
    - Разделители элементов шаблона
    - Присваивание переменных из шаблона
    - Отложенные шаблоны

        - Массив GMA в шаблонах
    - Управление шаблонами
        - Переключатели шаблонов
        - Преобразование шаблона в подпрограмму
        - Интерактивное создание шаблонов
- Описание языка
    - Основные определения
        - Условные обозначения
        - Программы обработки технологических команд, комментарии
        - Подпрограммы
        - Концепция оператора языка
        - Набор символов
        - Переменные
        - Массивы
        - Математические выражения и функции
        - Предопределённые переменные и функции
            - Стандартные математические функции
            - Строковые функции
            - Служебные функции и переменные
            - Функции и операторы доступа к CLData
        - Filter — выходной фильтр
        - Записи
        - Объекты
            - EulerConverter
            - Matrix
            - NCChannel
            - Point3D
            - Quaternion4d
            - StringHelper
    - Операторы
        - Оператор начала программы обработки PROGRAM
        - Оператор присваивания
        - Оператор вывода PRINT
        - Оператор ввода INPUT
            - Ввод из файла
            - Ввод с клавиатуры
        - Условный оператор IF
        - Оператор многовариантного выполнения CASE
        - Оператор JUMP
        - Оператор цикла FOR

        - Оператор CONTINUE
        - Оператор BREAK
    - Оператор цикла REPEAT
    - Оператор цикла WHILE
    - Составной оператор BEGIN...END
    - Оператор вызова подпрограммы CALL
    - Оператор начала подпрограммы SUB
    - Оператор конца подпрограммы SUBEND
    - Оператор начала процедуры PROC
    - Оператор возврата из процедуры RETURN
    - Оператор вывода кадра OUTBLOCK
    - Оператор формирования кадра FORMBLOCK
    - Оператор прямого вывода в кадр OUTPUT
    - Оператор замены подстроки в строке REPLACE
    - Оператор формирования кадра по шаблону MASK
    - Оператор смены текущего файла УП
    - Оператор вызова внешней задачи
    - Оператор доступа к CustomData CAM-системы
    - Операторы работы с подпрограммами УП
        - Оператор вывода подпрограммы УП NCSUB.OUTPUT
        - Оператор NCSUB.OUTPUTALL
        - Оператор задания начального маркера подпрограммы УП NCSUB.STARTLABEL
        - Оператор задания конечного маркера подпрограммы УП NCSUB.ENDLABEL
        - Оператор задания имени подпрограммы УП NCSUB.NAME
        - Техника работы с подпрограммами УП
    - Оператор ReplNCStr для замены строк в файле УП
- Разработка постпроцессоров с помощью ИИ
    - Как устроена разработка с помощью ИИ
    - Настройка инструментов
    - Организация рабочего пространства
    - Разработка через CAM Agent
    - Разработка в Visual Studio Code
    - Что дают расширения
    - Рабочий цикл SPPX
    - Рабочий цикл .NET
    - Проверка и подтверждение результата
    - Устранение проблем
    - Расширенная настройка
- Справочник CLData
    - Функции и операторы доступа к CLData

        - Именованные параметры CLData
        - Предопределённый массив CLD
        - Текущая команда (оператор Cmd)
        - Обход команд CLData
        - Файлы CLData (оператор CLDFile)
        - Информация о проекте (оператор Project)
        - Массив GMA
    - Информация о проекте в CLData
    - Описание технологических команд
        - AXESBRAKE - Управление тормозами осей станка
        - CIRCLE - Движение по окружности
        - CLAMP - Зажим приспособлений
        - COMMENT - Комментарии
        - COOLNT - Охлаждение
        - CUTCOM - Компенсация инструмента
        - CYCLE - Циклы обработки отверстий
            - CYCLE BORE5(209) - Цикл сверления G85
            - CYCLE BORE6(210) - Цикл сверления G86
            - CYCLE BORE7(211) - Цикл сверления G87
            - CYCLE BORE8(212) - Цикл сверления G88
            - CYCLE BORE9(213) - Цикл сверления G89
            - CYCLE BRKCHP(288) - Глубокое сверление с дроблением стружки
            - CYCLE DEEP(153) - Глубокое сверление с удалением стружки
            - CYCLE DRILL(163) - Цикл сверления G81
            - CYCLE FACE(81) - Цикл сверления G82
            - CYCLE TAP(168) - Цикл сверления G84
        - DELAY - Выстой
        - EDMMOVE - Электроэрозионное перемещение
        - EFFECTOR - Активация оконечного эффектора
        - EXTCYCLE - Расширенный цикл
            - EXTCYCLE BRKCHP - Токарный цикл сверления с дроблением стружки
            - EXTCYCLE DEEP - Токарный цикл сверления с удалением стружки
            - EXTCYCLE DRILL - Токарный цикл сверления (G81)
            - EXTCYCLE FACE - Токарный цикл сверления (G82)
            - EXTCYCLE LATHEFINISH - Токарный чистовой цикл (G70, G73)
            - EXTCYCLE LATHEGROOVE - Токарный цикл прорезки канавок (G74, G75)
            - EXTCYCLE LATHEROUGH - Токарный черновой цикл (G71, G72)
            - EXTCYCLE LATHETHREAD - Токарный цикл нарезания резьбы (G76)
            - EXTCYCLE LATHETHREADG92(404) - Токарный однопроходный цикл нарезания резьбы (G92)

        - EXTCYCLE TAP - Токарный цикл нарезания резьбы метчиком (G84)
        - EXTCYCLE W5DBore5 - Цикл сверления (G85)
        - EXTCYCLE W5DBore6 - Цикл сверления (G86)
        - EXTCYCLE W5DBore7 - Цикл сверления (G87)
        - EXTCYCLE W5DBore8 - Цикл сверления (G88)
        - EXTCYCLE W5DBore9 - Цикл сверления (G89)
        - EXTCYCLE W5DChipBreaking - Цикл сверления с дроблением стружки (G73)
        - EXTCYCLE W5DChipRemoving - Цикл сверления с удалением стружки (G83)
        - EXTCYCLE W5DDrill - Цикл сверления (G81)
        - EXTCYCLE W5DFace - Цикл сверления (G82)
        - EXTCYCLE W5DHolePocketing - Цикл выборки отверстия
        - EXTCYCLE W5DTap - Цикл нарезания резьбы метчиком (G84)
        - EXTCYCLE W5DThreadMill - Цикл фрезерования резьбы
        - EXTCYCLE WProbing - Измерительный цикл
            - Дополнительные параметры
            - Параметры измерения бобышки
            - Параметры измерения наружного угла (две стенки)
            - Параметры измерения внутреннего угла (две стенки)
            - Параметры измерения наружного прямоугольника
            - Параметры измерения паза
            - Параметры измерения паза с зоной защиты
            - Параметры измерения отверстия
            - Параметры измерения отверстия с зоной защиты
            - Параметры измерения внутреннего прямоугольника
            - Параметры измерения внутреннего прямоугольника с зоной защиты
            - Параметры измерения поверхности
            - Параметры измерения наружного угла (три стенки)
            - Параметры измерения внутреннего угла (три стенки)
            - Параметры измерения выступа (три точки)
            - Параметры измерения выступа
    - FEDRAT - Подача
    - Filter - Выходной фильтр (глобальный обработчик)
    - FINI - Конечная запись
    - FROM - Начальная точка
    - GOHOME - Возврат в исходную позицию
    - GOTO - Линейное перемещение
    - HEAD - Номер шпиндельной головки
    - INSERT - Вставка
    - INTERPOLATION - Режим интерполяции
    - LOADTL - Загрузка инструмента

- MULTIARC - Многоосевое движение по окружности
- MULTIGOTO - Многокоординатное перемещение
- OPSKIP - Условный пропуск
- OPSTOP - Дополнительный останов
- ORIGIN - Исходные координаты
- PALETA - Смена палеты
- PARTNO - Номер детали
- PHYSICGOTO - Перемещение по физическим осям станка
- PLANE - Рабочая плоскость
- POWER - Задание мощности
- PPFUN - Постпроцессорная функция
  - PPFUN CALLSUB(52) - Команда вызова подпрограммы УП
  - PPFUN ENDSUB(51) - Команда конца подпрограммы УП
  - PPFUN ENDTECHINFO(59) - Конец технологической информации операции
  - PPFUN PPFUN(500) - Команда контроля стойкости инструмента
  - PPFUN STARTSUB(50) - Команда начала подпрограммы УП
  - PPFUN TECHINFO(58) - Команда технологических параметров
  - PPFUN WEDMConditions(56) - Команда задания условий электроэрозионной обработки
  - PPFUN WEDMPassItem(57) - Свойства прохода проволочной электроэрозионной обработки
- PPRINT - Печать постпроцессора
- RAPID - Ускоренное перемещение
- ROTABL - Поворот стола
- SAFPOS - Точка смены инструмента
- SELWORKPIECE - Выбор активной державки заготовки
- SINGLETHREAD - Однопроходная резьба
- SPINDL - Шпиндель
- STOP - Останов
- STRUCTURE - Структура траектории
- SYNCAXES - Синхронизация движения осей
- TAKEOVER - Передача заготовки
- TLCONTACT - Контакт инструмента
- WAIT - Ожидание точки синхронизации

Внешние ссылки и примеры (Генератор постпроцессоров / sppx)

Внешние ресурсы для разработки `.sppx`-постпроцессоров в Генераторе постпроцессоров.

Документация по продукту

Ссылка	URL
Портал СПРУТКАМ	https://sprut.ru/
Руководство «Генератор постпроцессоров»	Открыть руководство
Учебник по разработке постпроцессоров	https://docs.sprut.ru/tutorials/lnp/17/index.html

Разработка с помощью ИИ

Адреса загрузки инструментов, описанных в разделе Разработка постпроцессоров с помощью ИИ.

Ссылка	URL
Инструменты постпроцессирования для VS Code — выпуски	https://gitverse.ru/spruttechnology/postprocessor-tools/releases
Центр документации СПРУТКАМ	https://docs.sprut.ru/portal/

Сопутствующие ссылки

- Постпроцессоры — .NET — другая подсистема постпроцессирования, в виде .NET-сборок.
- CLData — общая входная модель, читаемая обеими подсистемами.

Введение

Сейчас CAM-система имеет две независимые системы постпроцессирования.

1. Генератор постпроцессоров. Это интегрированная среда разработки постпроцессоров, которая имеет свой собственный формат, язык программирования и отладчик.
2. Новая открытая система постпроцессирования, базирующаяся на платформе .NET, использующая C# как язык программирования и Visual Studio Code как среду разработки и отладки кода.

Обе системы актуальны и могут быть использованы на ваше усмотрение.

Здесь вы найдёте документацию только для Генератора постпроцессоров. Документация для второй системы (основанной на .NET) доступна здесь.

Назначение генератора постпроцессоров

Генератор постпроцессоров является приложением для операционных систем семейства Windows.

Назначение генератора постпроцессоров — разработка файлов настройки постпроцессора на различные системы ЧПУ. Эти файлы используются исполняющей системой постпроцессора для формирования конкретной 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 файл настройки постпроцессора.

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

Кроме работы с данными о станке и системе ЧПУ, в среде генератора постпроцессоров также возможны просмотр файлов технологических команд и тестовая генерация управляющих программ.

Набор файлов генератора постпроцессоров

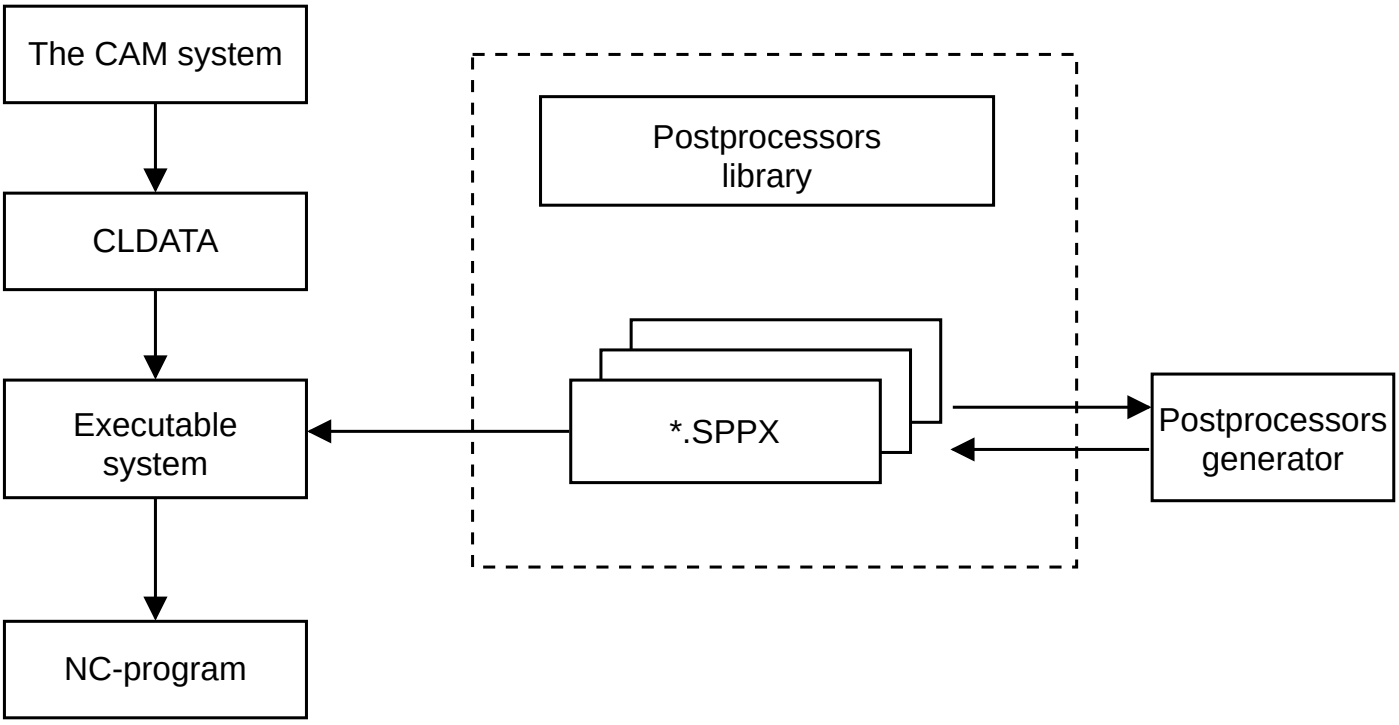
- Inp.exe – исполняемый модуль генератора постпроцессоров.
- InpD.dll – исполняющая система постпроцессора.
- CLDImport.dll – библиотека импорта проектов CAM-системы.
- *.sppx – файл с данными о конкретной системе ЧПУ и с исходными текстами программ обработки технологических команд.
- *.stcx – XML-файлы проектов CAM-системы.
- *.stcp – файлы проектов CAM-системы.
- *.mcd – временные файлы технологических команд, связанные с файлами проектов CAM-системы.
- *.spp, *.inp – файлы настройки постпроцессора для предыдущей версии генератора постпроцессоров.

Общая организация работы

На этой странице нет содержимого. См. вложенные разделы.

Принцип работы постпроцессора

С помощью генератора постпроцессоров разрабатываются файлы настройки постпроцессора для различных систем ЧПУ (файлы *.SPPX). Также возможен импорт файлов предыдущей версии постпроцессора (файлы *.SPP, *.INP). Файл настройки постпроцессора содержит описание всех особенностей формирования управляющей программы для указанной системы ЧПУ. Это описание используется исполняющей системой постпроцессора при формировании управляющих программ из файлов технологических команд (*.MCD, *.STCP, *.STCX), которые, в свою очередь, могут быть сгенерированы, например, CAM-системой.



Для разработки файла настройки постпроцессора необходимо заполнить данные о станке и системе ЧПУ, описать структуру и формат кадра, а также заполнить шаблоны 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. Идентификаторы и значения регистров выводятся в кадр управляющей программы в том же порядке, в котором регистры расположены в списке.

Шаблон содержит перечень регистров в требуемом порядке. Эти регистры выводятся в кадр управляющей программы для соответствующей технологической команды.

Программы обработки технологических команд пишутся на специальном проблемно-ориентированном языке. Этот язык допускает математические выражения и функции, операторы ввода/вывода, условные операторы, циклы, оператор перехода, вызовы подпрограмм, операторы формирования кадров управляющей программы и операторы работы с файлом технологических команд.

Заполнение данных о станке и системе ЧПУ, описание структуры и формата кадра, написание программ обработки технологических команд производятся в среде генератора постпроцессоров. В этой среде также возможны просмотр файлов технологических команд и тестовая генерация управляющей программы.

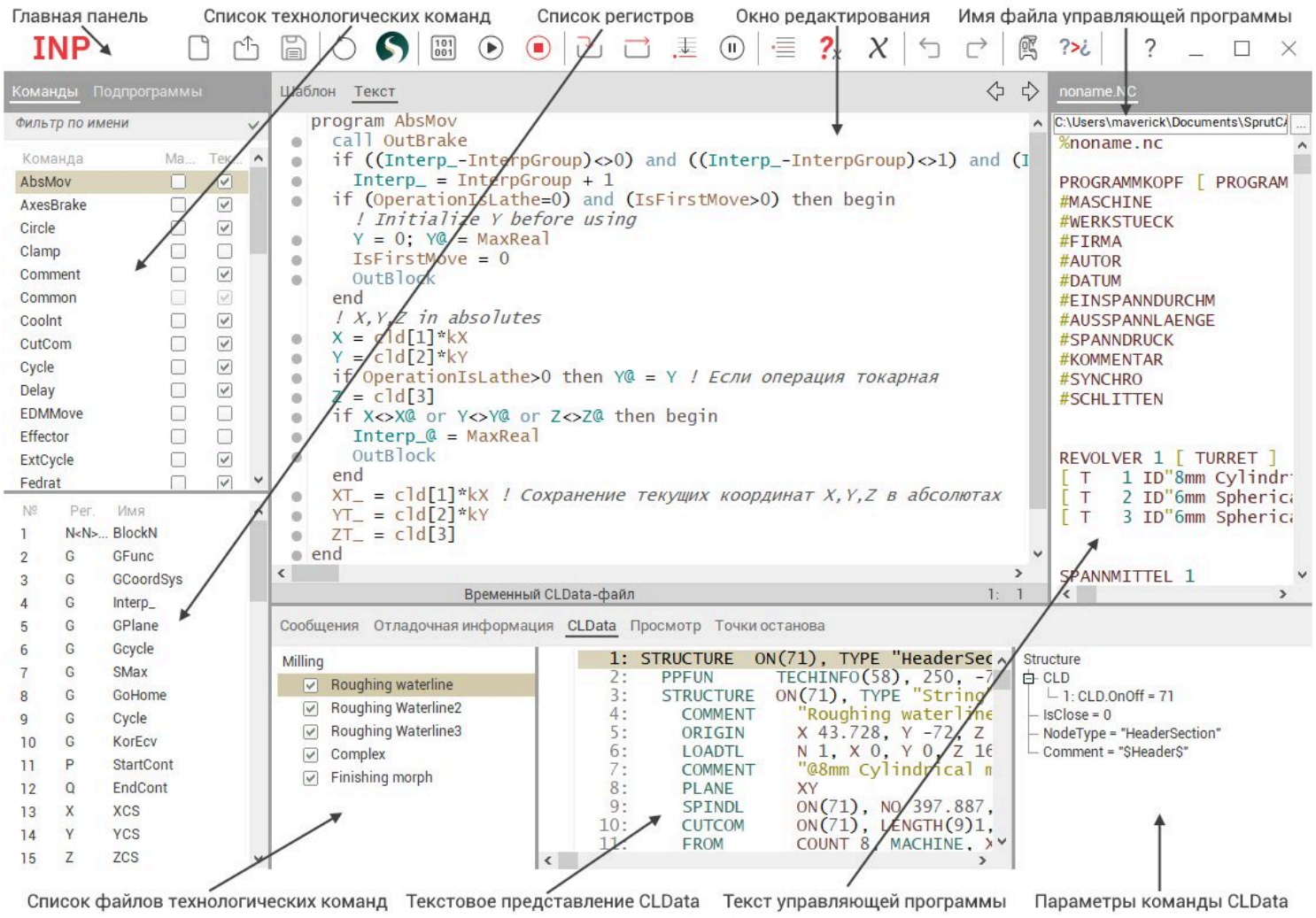
Исходные тексты программ обработки технологических команд, данные о станке и системе ЧПУ, шаблоны, список и формат регистров сохраняются в файле с именем конкретного постпроцессора и расширением *.sppx. Данные из этого файла используются исполняющей системой постпроцессора для формирования управляющей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

Исполняющая система постпроцессора считывает данные о процессе обработки из файла технологических команд, анализирует код команды и активирует программу, которая обрабатывает эту команду (имя шаблона и программы обработки совпадает с именем соответствующей команды). Затем, если использовался шаблон, он формирует строку управляющей программы по соответствующему шаблону. Параметры технологической команды передаются через предопределённый массив CLD и специальный оператор CMD. Вызванная программа может изменять значения регистров и внутренних переменных, а также формировать кадр управляющей программы.

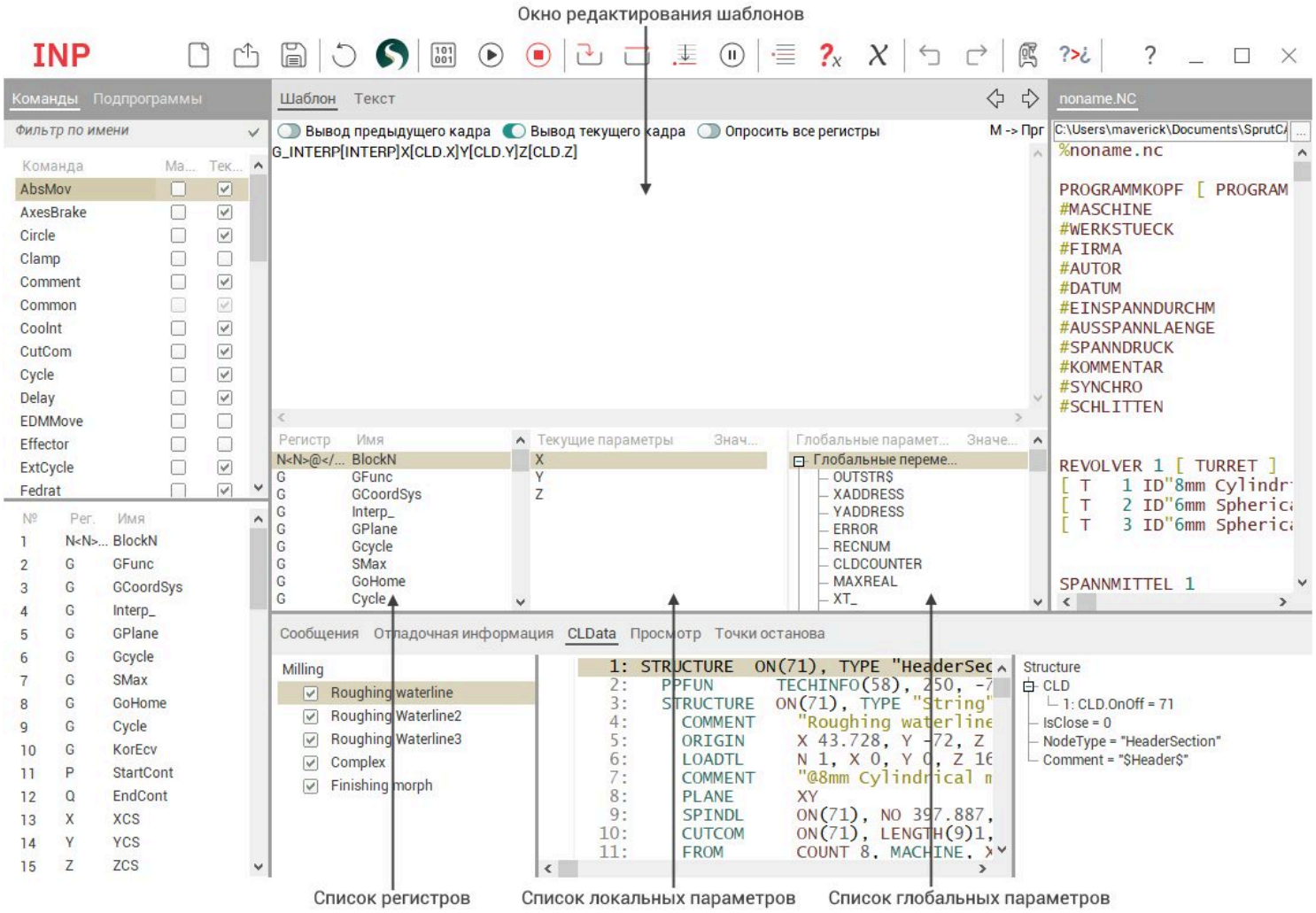
Соответствующие операторы в программах обработки команд формируют кадр управляющей программы, причём в текущий кадр выводятся только идентификаторы и значения тех регистров, которые изменились с момента формирования предыдущего кадра.

Главное окно

Главное окно:



В верхней части окна расположена Главная панель инструментов. Слева расположены список программ обработки команд и список регистров. В центре находятся переключающиеся страницы с редактором шаблонов и редактором программ обработки команд. Кроме редактора, на вкладке Шаблон расположены списки регистров, локальных и глобальных параметров. Внизу расположены переключающиеся окна системных сообщений, отладочной информации, список файлов, содержащих траекторию движения инструмента, окно текстового представления этих файлов, список контролируемых переменных и список точек останова.



Сопутствующие ссылки

[Главная панель инструментов]

[Системные установки]

[Настройка редактора]

[Определение данных о станке и системе ЧПУ]

[Транслитерация комментариев]

[Определение структуры и формата кадра]

[Шаблоны трансляции технологических команд]

[Программы обработки команд CLData]

[Подпрограммы]

[Трансляция программ обработки команд]

Работа с файлами технологических команд

Тестовая генерация управляющих программ

Отладка программ

Главная панель инструментов

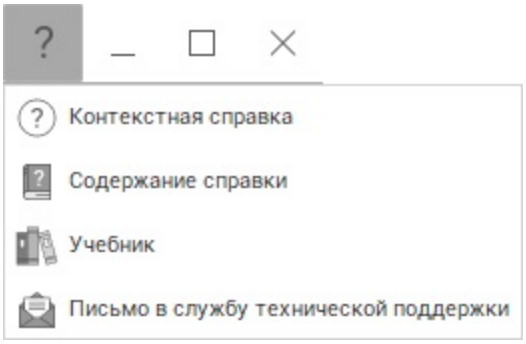


- – Создание нового файла настройки постпроцессора. Перед созданием система сбрасывает своё состояние.
- – Открытие ранее сохранённого файла настройки постпроцессора. Перед загрузкой система сбрасывает своё состояние.
- – Сохранение файла настройки постпроцессора под текущим именем.
- – Запрос новых файлов CLData из CAM-системы.
- – Открытие файла технологических команд, сгенерированного CAM-системой.
- – Запуск трансляции программ обработки команд.
- – Запуск формирования управляющей программы из файлов траектории движения инструмента. Если программы обработки команд не оттранслированы, сначала будет запущена трансляция.
- – Прервать формирование управляющей программы.
- – Формировать управляющую программу с остановкой в подпрограммах.
- – Формировать управляющую программу без остановки в подпрограммах.
- – Формировать управляющую программу до текущей позиции в CLData.
- – Приостановить формирование управляющей программы.
- – Добавить точку останова в обработчике технологических команд или файлах CLData.
- – Добавить новую переменную в список наблюдаемых.
- – Вычислить значение переменной или выражения. Вычисление возможно только в режиме отладки.
- – Отменить последнее изменение, сделанное в тексте программы обработки технологической команды или шаблона.
- – Вернуть последнее изменение, сделанное в тексте программы обработки технологической команды или шаблона.
- – Открытие окна данных о станке и системе ЧПУ.
- – Открытие диалога свойств транслитерации. Панель инструментов приложения расположена в верхней левой части главного окна. С двумя кнопками связаны выпадающие меню.



- – Открытие окна системных установок.
- INP – Информация о генераторе постпроцессоров.

Меню помощи состоит из действий, которые помогают получить ответы на возникающие вопросы.



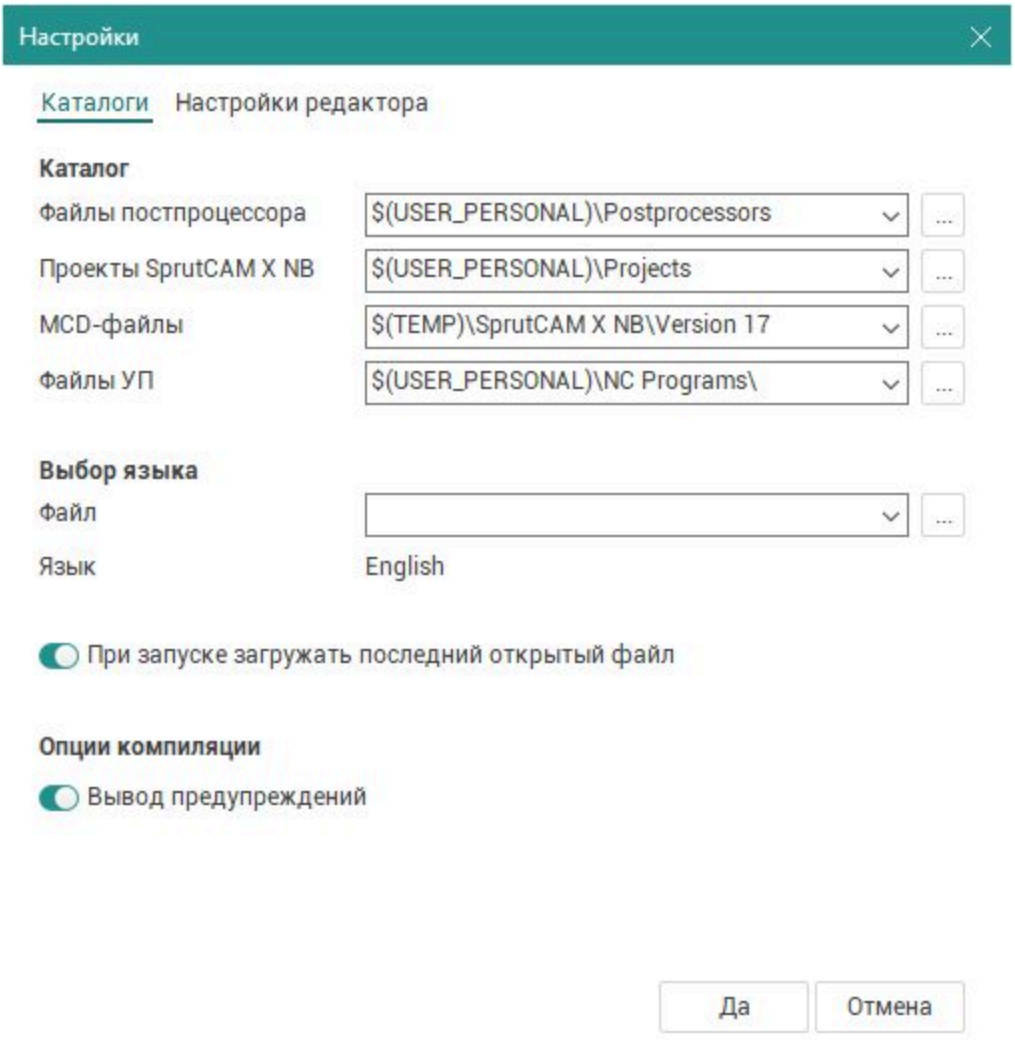
- – Запуск справки генератора постпроцессоров.
- – Подготовка сообщения для службы технической поддержки поставщика CAM-системы.

Сопутствующие ссылки

Главное окно

Системные установки

Окно системных установок вызывается из главного меню. Здесь задаются пути по умолчанию для системных файлов и языков поддержки.



Файлы настройки постпроцессора для различных систем ЧПУ (файлы *.sppx) загружаются из каталога Файлы постпроцессора. Новые файлы настройки постпроцессора сохраняются в этот же каталог.

Проекты CAM-системы загружаются по умолчанию из каталога Проекты CAM-системы. Файлы технологических команд (*.mcd), связанные с открытым проектом, загружаются по соответствующим путям, указанным в файле проекта.

Отдельные файлы технологических команд (*.mcd) загружаются по умолчанию из каталога MCD-файлы.

Сгенерированные управляющие программы сохраняются по умолчанию в каталог Файлы УП.

Для смены языка программы используется панель Выбор языка.

Пути можно корректировать вручную или при помощи диалогов, которые вызываются нажатием кнопки ... .

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

При определении реальных имён каталогов в процессе работы указанная переменная заменяется соответствующим полным путём на момент запуска системы или на момент последнего редактирования системных установок.

Для загрузки последнего открытого файла *.SPPX при запуске генератора постпроцессоров имеется флажок При запуске загружать последний открытый файл.

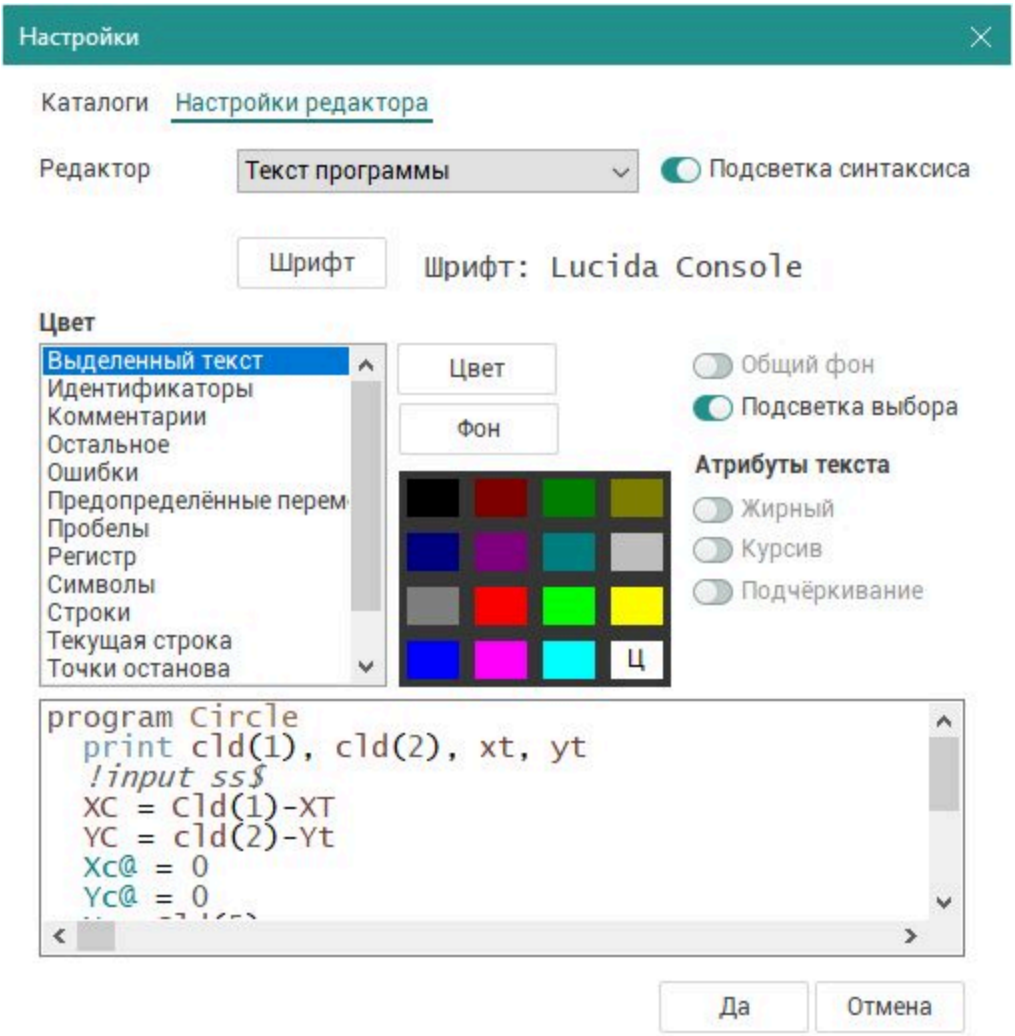
Сопутствующие ссылки

Главное окно

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, а также цвет фона.

Окно настройки редактора вызывается из главного меню. В окне настраиваются шрифт и подсветка синтаксиса для редактора программ обработки технологических команд, окна текстового представления 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 задаются название и размер шрифта.

На панели Цвет для каждой группы элементов можно назначить цвет текста, цвет фона текста и 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.

Функция Подсветка выбора подсвечивает выбранное слово по всему редактору.

Кнопка OK закрывает окно и сохраняет все изменения. Кнопка Отмена закрывает окно и отменяет все изменения.

Сопутствующие ссылки

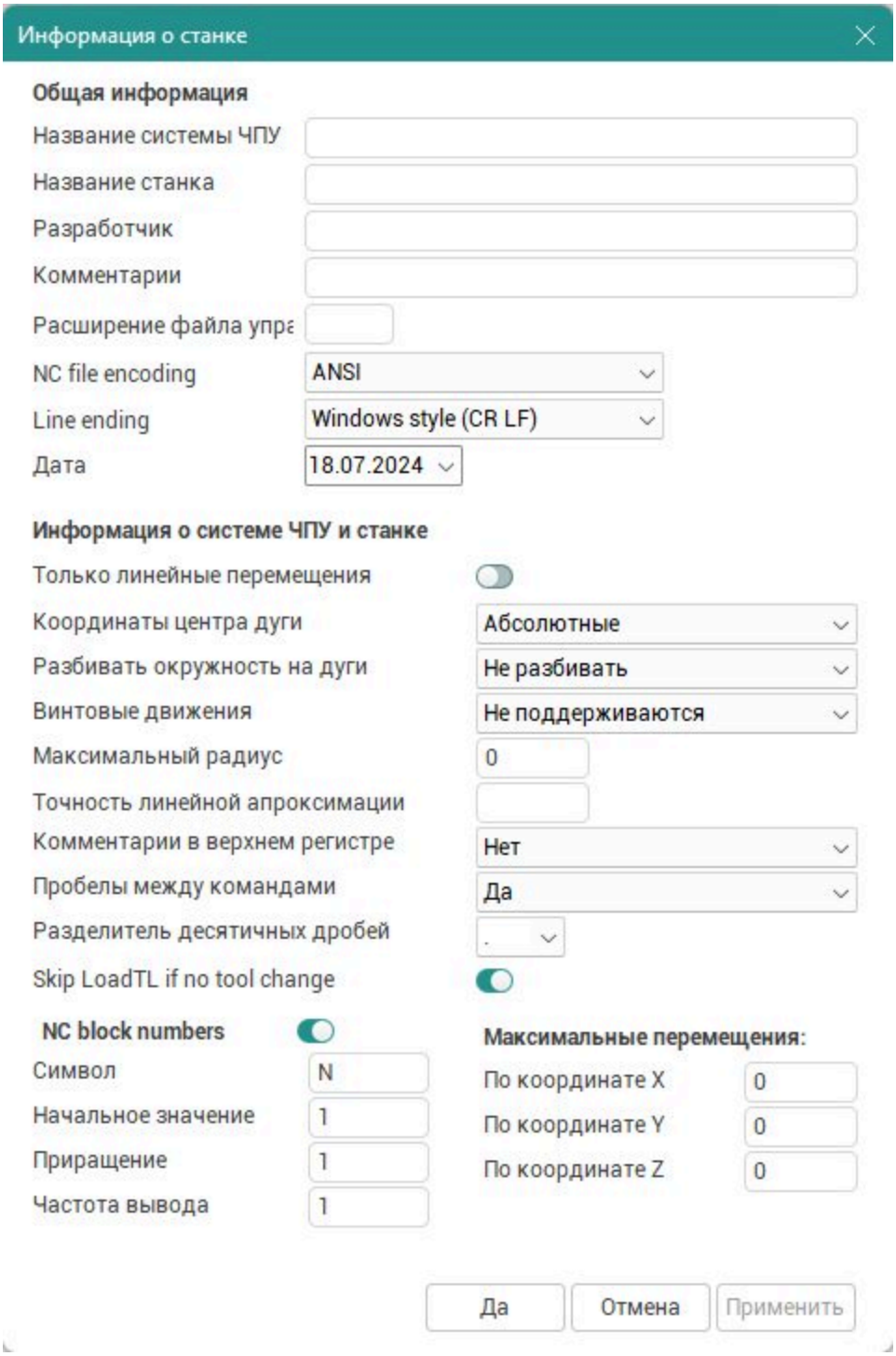
Главное окно

Параметры станка

На этой странице нет содержимого. См. вложенные разделы.

Определение данных о станке и системе ЧПУ

Окно редактирования данных о станке и системе ЧПУ открывается нажатием кнопки на главной панели инструментов.



Название станка, название системы ЧПУ и расширение файлов управляющей программы отображаются на панели Общая информация. Название станка и название системы ЧПУ являются только информационными параметрами. Все управляющие программы,

сгенерированные с использованием редактируемого файла настройки, сохраняются с указанным расширением.

На панели Информация о системе ЧПУ и станке задаются соответствующие параметры:

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, 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, при 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 отрезками и формирует команды линейных перемещений.

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

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

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

Для формирования номера кадра имеется группа полей Нумерация кадра. Если соответствующий флажок на панели не установлен, то функция автоматической нумерации кадров выключена. При установке этого флажка становятся доступными несколько полей настройки автоматической нумерации. Группа состоит из полей Символ, Начальное значение, Приращение, Частота вывода. В поле Символ вводится идентификатор регистра или произвольный символ, выводимый в кадр перед значением номера кадра. Если в этом поле указать идентификатор регистра, то появляется возможность дополнительного контроля над номером кадра за счёт настроек формата регистра (например, если номер кадра выйдет за диапазон возможных значений регистра, то нумерация начнётся заново). В случае автоматической нумерации кадры нумеруются, начиная с Начального значения. После каждого вывода кадра к номеру кадра добавляется Приращение. Величина приращения

доступна не только из окна Общие параметры, но и программно через предопределённую переменную BlockStep.

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

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

Разделитель десятичных дробей — символ, который будет выводиться в кадры УП в качестве разделителя десятичных дробей в числах.

Пропускать LoadTL если нет смены инструмента — пропускает вызов обработчика команды LoadTL, если он не приводит к реальной смене инструмента относительно предыдущей операции. Если эта опция включена и свойство ToolChanged команды LOADTL имеет значение 0 (false), то процедура LoadTL для данной команды вызвана не будет.

Кнопка OK закрывает окно и сохраняет все изменения. Кнопка Отмена закрывает окно и отменяет все изменения.

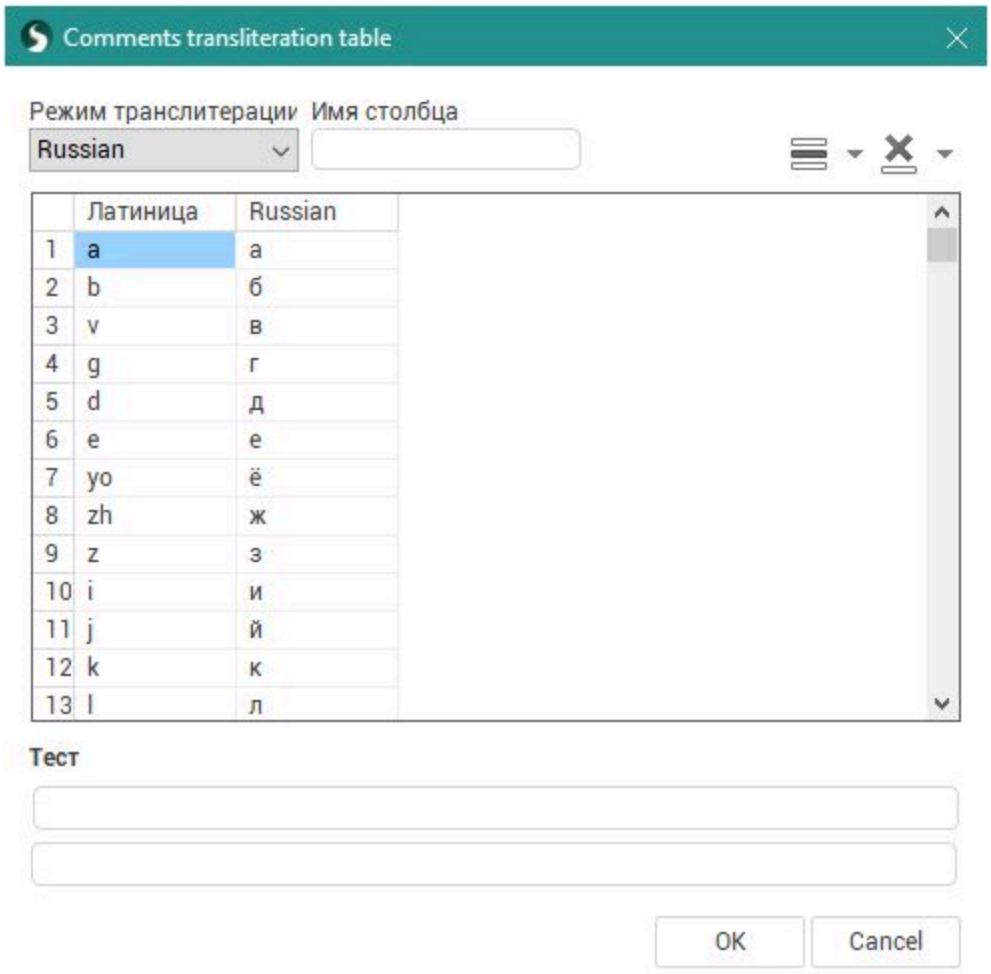
Сопутствующие ссылки

Главное окно

Транслитерация комментариев

Транслитерация комментариев используется постпроцессором для преобразования комментариев в управляющей программе (передаваемых через технологическую команду COMMENT), которые содержат символы, не поддерживаемые системой ЧПУ (например, символы, отличные от ASCII), в строки, содержащие только символы, допустимые для системы ЧПУ.

Параметры транслитерации задаются нажатием кнопки ?>¿ на главной панели инструментов.



Поле Режим транслитерации задаёт исходный язык комментариев (активный столбец в таблице транслитерации). По умолчанию установлено значение Отключено.

Каждый язык, который может быть транслитерирован, представлен столбцом в таблице транслитерации. Первый столбец Латиница является ключевым, его символы используются для замены символов из других языков. Строки таблицы определяют, какие символы в активном языке будут транслитерированы (строки активного столбца), и задают транслитерируемые символы или их сочетания (столбец Латиница).

Таблицу транслитерации можно изменять в любой момент: добавлять, удалять или переименовывать столбцы, а также добавлять, удалять или изменять строки и ячейки любого столбца. Для изменения имени столбца используется поле Имя столбца. Для добавления и удаления строк и столбцов таблицы используются следующие кнопки:

-  Новая строка – добавляет новую строку после выделенной;
-  Новый столбец – добавляет новый столбец после выделенного;
-  Удалить строку – удаляет выделенную строку;
-  Удалить столбец – удаляет выделенный столбец.

Панель Тест предназначена для проверки настроек транслитерации в реальном времени. При вводе текста в верхнее поле система выводит транслитерированный текст в нижнее поле.

Сопутствующие ссылки

Главное окно

Определение структуры и формата кадра (формирование списка регистров)

Для определения структуры и формата кадра 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 определённом формате. С каждым адресом в постпроцессоре связан определённый 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).

Кадры управляющей программы формируются автоматически при выполнении операторов OUTBLOCK и FORMBLOCK. Кадр управляющей программы формируется системой по следующему алгоритму. Система последовательно перебирает регистры; если текущее значение регистра отличается от его предыдущего значения, то регистр записывается в кадр, а его текущее значение присваивается предыдущему, иначе регистр в кадр не записывается.

При записи регистра в кадр управляющей программы сначала записывается идентификатор регистра, а затем его значение. Значение записывается в кадр в соответствии с описанным для этого регистра форматом (длина и точность значения регистра, наличие десятичной точки, знака, лидирующих и незначащих нулей).

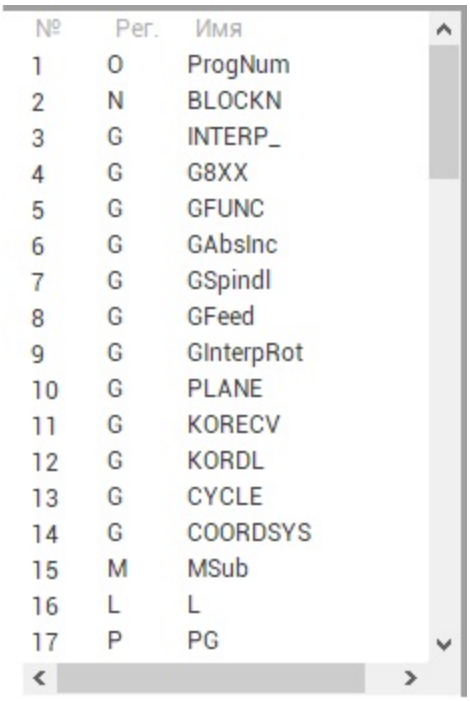
Таким образом, для того чтобы определить структуру и формат кадра, необходимо заполнить список регистров и задать их свойства. Регистры должны быть расположены в списке в том порядке, в котором они и их значения должны появляться в кадрах управляющей программы. Данное правило действительно для программ обработки. 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 (символы, которые записываются в кадр перед значением регистра). Это позволяет создавать отдельный регистр для каждой группы функций одного типа.

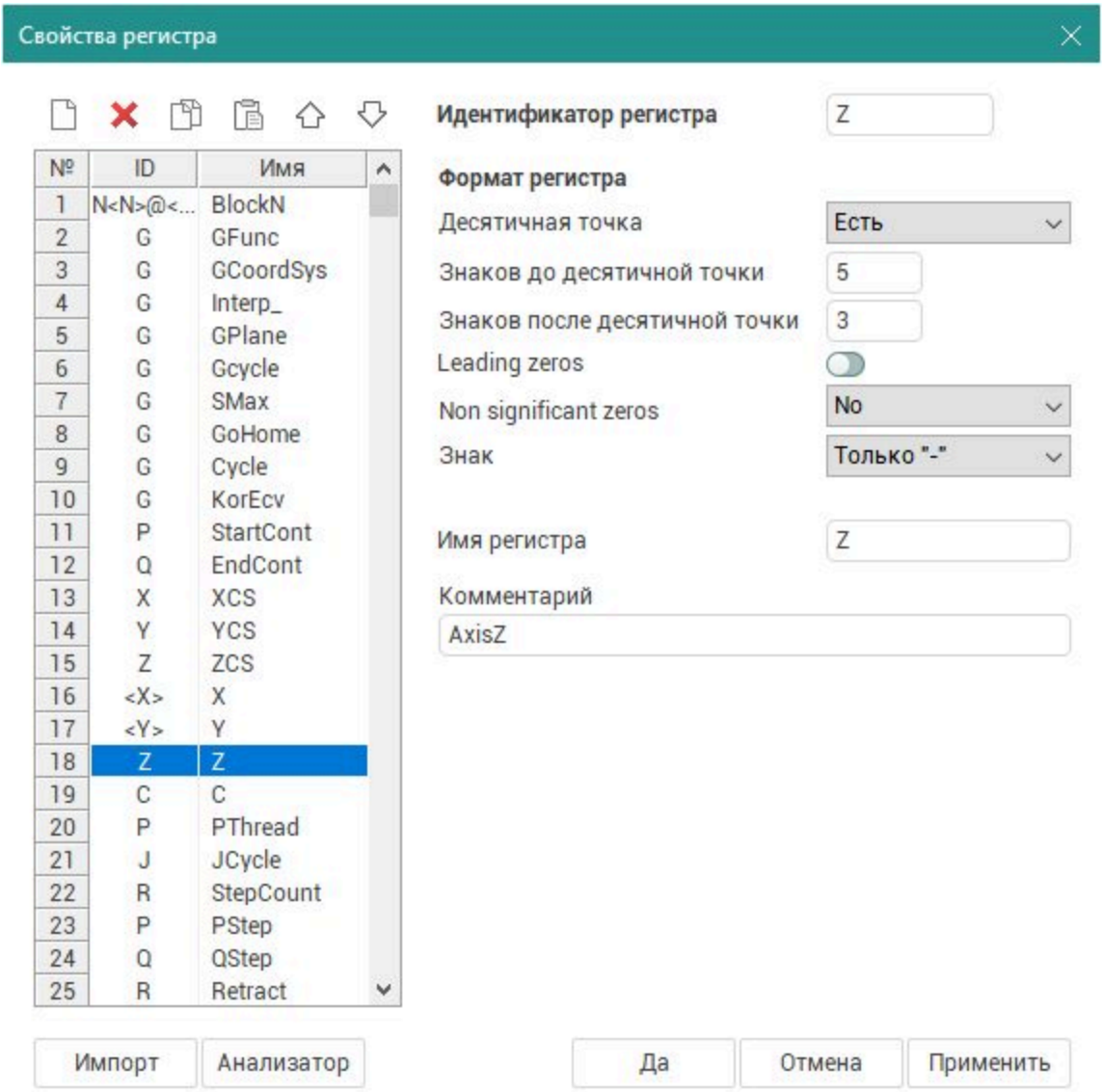
Например, создадим регистр с именем ABSOLUTE и идентификатором G для подготовительной функции переключения между абсолютной/относительной системами координат.

Предположим также, что мы создаём регистр с именем INTERP и таким же идентификатором G для подготовительной функции позиционирования, линейной интерполяции, круговой интерполяции с направлением. Это позволяет нам: во-первых, при необходимости, вывести обе эти команды в один кадр управляющей программы (например, `N190 G91 G1 X50 Y30`); во-вторых, отслеживать текущий режим перемещений (позиционирование, линейная или круговая интерполяция с направлением), определять текущую систему координат (абсолютная или относительная) по текущим значениям регистров ABSOLUTE и INTERP.

Список регистров отображается в левой части главного окна.



Окно редактирования свойств регистра открывается при добавлении регистра, при выборе пункта Свойства в контекстном меню или двойным нажатием левой клавиши мыши на регистре в списке.



Список регистров отображается в левой части окна. Кнопки редактирования списка расположены ниже:

- – вставляет новый регистр после выделенного, окно редактирования свойств нового регистра открывается автоматически;
- – удаляет выделенный регистр;
- – копирует выделенный регистр в буфер обмена;
- – вставляет регистр из буфера обмена после выделенного. Окно редактирования свойств вставленного регистра открывается автоматически;
- – переместить текущий регистр вверх по списку;
- – переместить текущий регистр вниз по списку.

Чтобы изменить положение регистра в списке, необходимо нажать левую клавишу мыши на регистре и перетащить его в нужную позицию, удерживая левую клавишу нажатой.

Если у вас многоканальный станок, вы можете управлять регистрами определённого канала, выбрав его в списке каналов.



Также можно вручную добавить каналы , скопировать существующий канал в новый или удалить текущий канал .

В этом окне можно задать следующие поля:

- Идентификатор регистра – символы, которые записываются в кадр управляющей программы перед значением регистра.
- Десятичная точка – это поле может иметь следующие значения:
  - Отсутствует;
  - Присутствует – при выборе этого варианта десятичная точка выводится, если значение регистра имеет дробную часть;
  - Присутствует всегда.
- Знаков до десятичной точки – максимальное количество знаков в целой части числа.
- Знаков после десятичной точки – максимальное количество знаков в дробной части числа.
- Лидирующие нули и Незначащие нули – определяют режим вывода нулей перед и после значения регистра.
- Знак – определяет режим вывода знака значения регистра. Доступны следующие варианты:
  - Без знака;
  - Только "-";
  - Всегда "+" и "-";
- Имя регистра – имя регистра, которое используется программами обработки команд.
- Комментарий – комментарий к регистру.

Кнопка Импорт предназначена для импорта списка регистров из постпроцессоров форматов SPPX, SPP и PPP (формат старой версии), а также 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.

Кнопка OK закрывает окно и сохраняет все изменения. Кнопка Отмена закрывает окно и отменяет все изменения. Кнопка Применить сохраняет все изменения, окно регистров не закрывается.

Текущее значение любого регистра доступно из программного кода по имени регистра как обычная переменная. Чтобы получить предыдущее значение регистра, необходимо записать имя регистра со знаком @. Например:

```
program AbsMov
  Len1: Real
  Len1 = sqr((X-X@)^2 + (Y-Y@)^2 + (Z-Z@)^2) ! Здесь X, Y и Z - текущие значения
регистров, X@, Y@ и Z@ - предыдущие значения регистров
end
```

Также можно получить доступ к регистрам из программного кода с помощью оператора RGS в форме Rgs[Индекс в списке] или Rgs["Имя регистра"]. Например, для конкретного списка регистров, показанного на рисунке выше:

```
program AbsMov
  Len1: Real
  Len1 = sqr((Rgs[3]-Rgs[3]@)^2 + (Rgs[4]-Rgs[4]@)^2 + (Rgs[5]-Rgs[5]@)^2) ! Здесь
Rgs[3] - текущее значение регистра X, Rgs[3]@ - предыдущее значение регистра X
end
```

Полный синтаксис, который может быть применён к оператору RGS, приведён ниже. Внутри квадратных скобок можно указывать либо индекс регистра в списке (целочисленная константа или переменная), либо имя регистра (строковая константа или переменная).

- `RGS.ItemCount` - возвращает общее количество регистров в списке.
- `Rgs[<Index or Name>].STR` - строковый идентификатор (ID) регистра, который выводится в управляющую программу.
- `Rgs[<Index or Name>].NAME` - строковое имя регистра, используемое для доступа к нему из программного кода постпроцессора.
- `Rgs[<Index or Name>]` или эквивалентно `Rgs[<Index or Name>].DATA` - текущее значение регистра.
- `Rgs[<Index or Name>]@` или эквивалентно `Rgs[<Index or Name>].OLDDATA` - предыдущее значение регистра.
- `Rgs[<Index or Name>].NUM` - возвращает индекс регистра в списке регистров.

Пример, демонстрирующий преимущества использования оператора RGS вместо имени регистра, приведён ниже.

```
program AbsMov
  ToolAxisIndex: Integer
  case PlaneGNumber of ! Запоминаем индекс регистра осевого перемещения инструмента
в зависимости от текущей плоскости
    17: ToolAxisIndex = Rgs["Z"].Num
    18: ToolAxisIndex = Rgs["Y"].Num
    19: ToolAxisIndex = Rgs["X"].Num
  end
  X = CLD.X ! Заполняем текущие значения регистров X, Y и Z из CLDATA
  Y = CLD.Y
  Z = CLD.Z
  if Interp=0 then begin ! Если мы имеем случай ускоренного перемещения и хотим
реализовать безопасное опускание
    if Rgs[ToolAxisIndex] < Rgs[ToolAxisIndex]@ then begin ! Если инструмент
перемещается вниз
      Rgs[ToolAxisIndex]@ = Rgs[ToolAxisIndex] ! То мы предотвращаем вывод
перемещения вдоль оси инструмента
      OutBlock ! Таким образом, здесь мы выведем перемещение только в
плоскости инструмента
      Rgs[ToolAxisIndex]@ = 999999 ! Теперь мы должны присвоить любое несовпадающее
число старому значению регистра,
    end ! чтобы обеспечить гарантированный вывод перемещения вдоль оси инструмента
при следующем OutBlock
  end
  OutBlock ! Вывод перемещения в УП
end
```

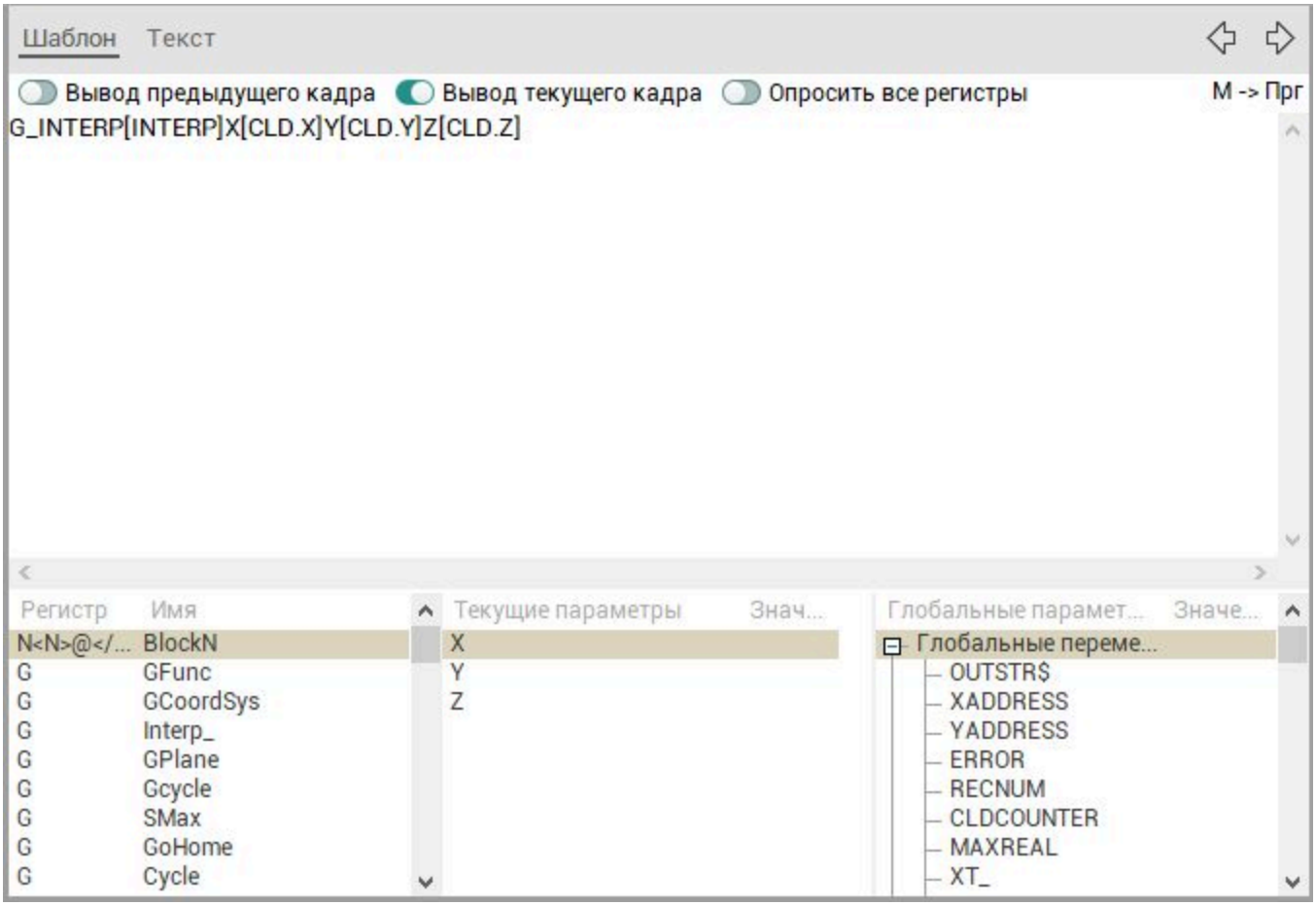
Сопутствующие ссылки

Главное окно

Шаблоны трансляции технологических команд

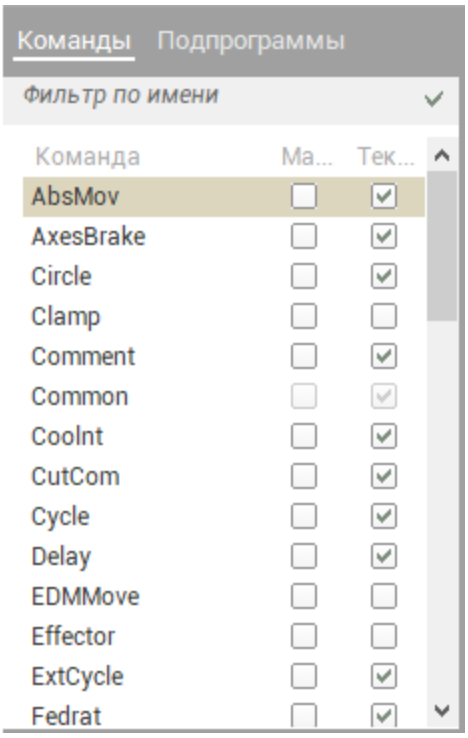
Использование шаблонов позволяет быстро и просто определить способ преобразования команд обработки CLData в строки NC-кода. Любой команде CLData соответствует свой шаблон, в котором определяются слова и значения для этих слов.

Редактор шаблонов расположен на странице Шаблон.



Для упрощения процесса создания шаблонов в нижней части окна находятся список регистров, список параметров текущей команды и список глобальных переменных. Двойной щелчок по любому элементу этих списков вставляет элемент в текст шаблона. Над полем текста шаблона расположена панель флажков, управляющая процессом обработки шаблона.

Активизация шаблона выполняется установкой флажка в списке команд CLData.



Правила формирования шаблонов подробно описаны в главе Шаблоны постпроцессора.

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

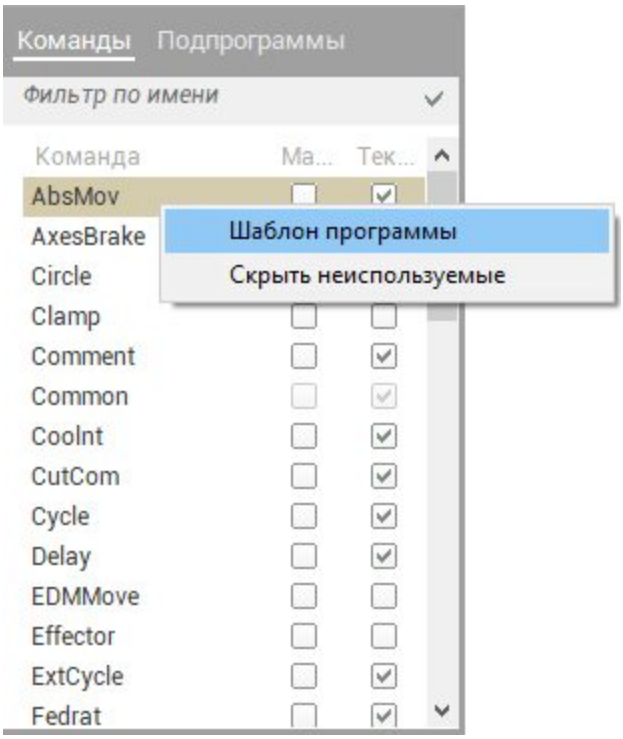
Сопутствующие ссылки

Главное окно

Программы обработки команд CLData

Как и шаблоны, программы служат для определения процесса преобразования команды CLData в строку NC-кода. Программы могут расширять шаблон или использоваться вместо шаблонов. И шаблон, и программы имеют свои достоинства и недостатки. Программы — гибкий и мощный инструмент для реализации самых сложных преобразований. Однако овладение этим инструментом требует опыта программирования.

Любой команде CLData соответствует своя программа обработки.



Эти программы пишутся на специальном проблемно-ориентированном языке и могут содержать математические выражения и функции, операторы ввода/вывода, условные операторы, циклы, оператор перехода, вызовы подпрограмм, операторы формирования кадров управляющей программы и операторы работы с файлом технологических команд. Этот язык подробно описан в главе Описание языка.

Каждая программа начинается с заголовка, состоящего из ключевого слова PROGRAM и имени программы, и заканчивается ключевым словом END. Имя программы совпадает с именем команды, которую она обрабатывает. При вызове программы параметры передаются в неё через предопределённый массив CLD и специальный оператор Cmd.

Активизация программы выполняется установкой флажка в столбце кода списка команд CLData. Если программа не активна, то она не транслируется и не выполняется, даже если её текст существует. Исключение составляет программа COMMON. Она транслируется и выполняется всегда и первой. Эта программа предназначена для определения глобальных переменных, т. е. переменных, доступных из любой подпрограммы.

Пустая программа создаётся двойным щелчком в списке технологических команд.

При нажатии правой кнопки мыши в списке программ появляется контекстное меню. Оно содержит следующие пункты:

- Шаблон программы — создаётся пустой шаблон программы, включающий заголовок и оператор конца;
- Скрыть неиспользуемые — неактивированные программы будут скрыты.

Программы могут использовать подпрограммы, которые организованы в отдельной вкладке. При нажатии правой кнопки мыши в списке подпрограмм появляется контекстное меню. Оно содержит следующие пункты:

- Добавить – добавляет новую программу обработки технологической команды. Откроется окно ввода имени программы.
- Удалить – удаляет выделенную программу.
- Переименовать – переименовывает текущую программу. Откроется окно ввода имени программы.

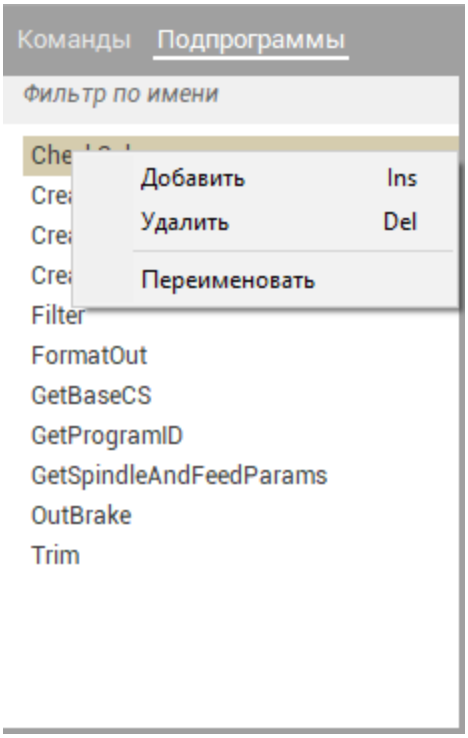
Редактор выбранной программы расположен в центре главного окна. При выборе другой программы в списке её текст сразу отображается в окне редактирования.

Сопутствующие ссылки

Главное окно

Подпрограммы

Вкладка Подпрограммы содержит список определяемых пользователем подпрограмм и подпрограмму Filter. Для добавления, удаления или переименования подпрограмм используйте контекстное меню. Код подпрограммы можно просматривать или редактировать в редакторе кода. Редактор кода отображает код выбранной подпрограммы.



Имя подпрограммы может быть любым допустимым строковым идентификатором, если он не совпадает с именем технологической программы и не используется другой подпрограммой.

Принципы определения и синтаксис подпрограмм описаны в главе «Описание языка», раздел «Подпрограммы».

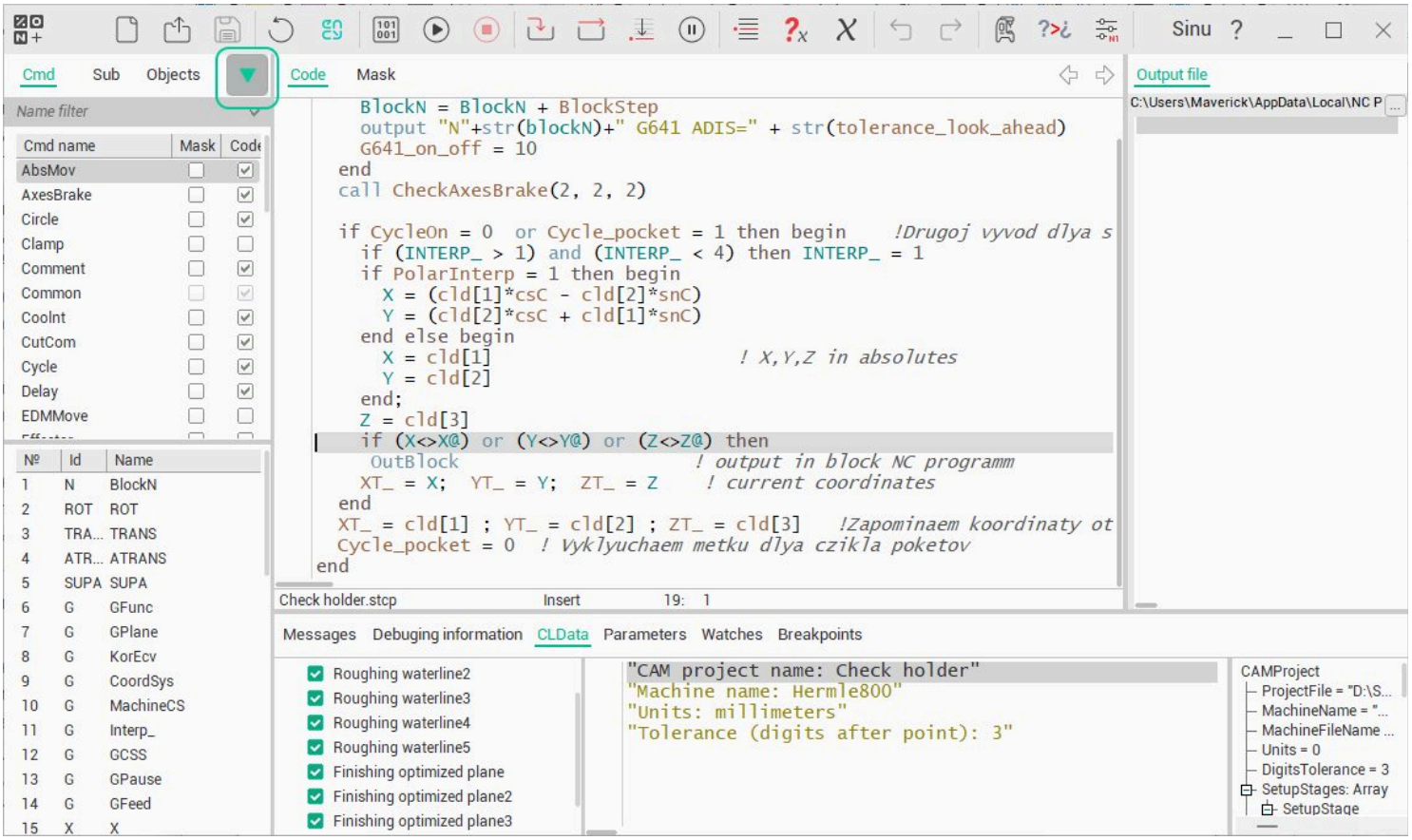
Сопутствующие ссылки

Главное окно

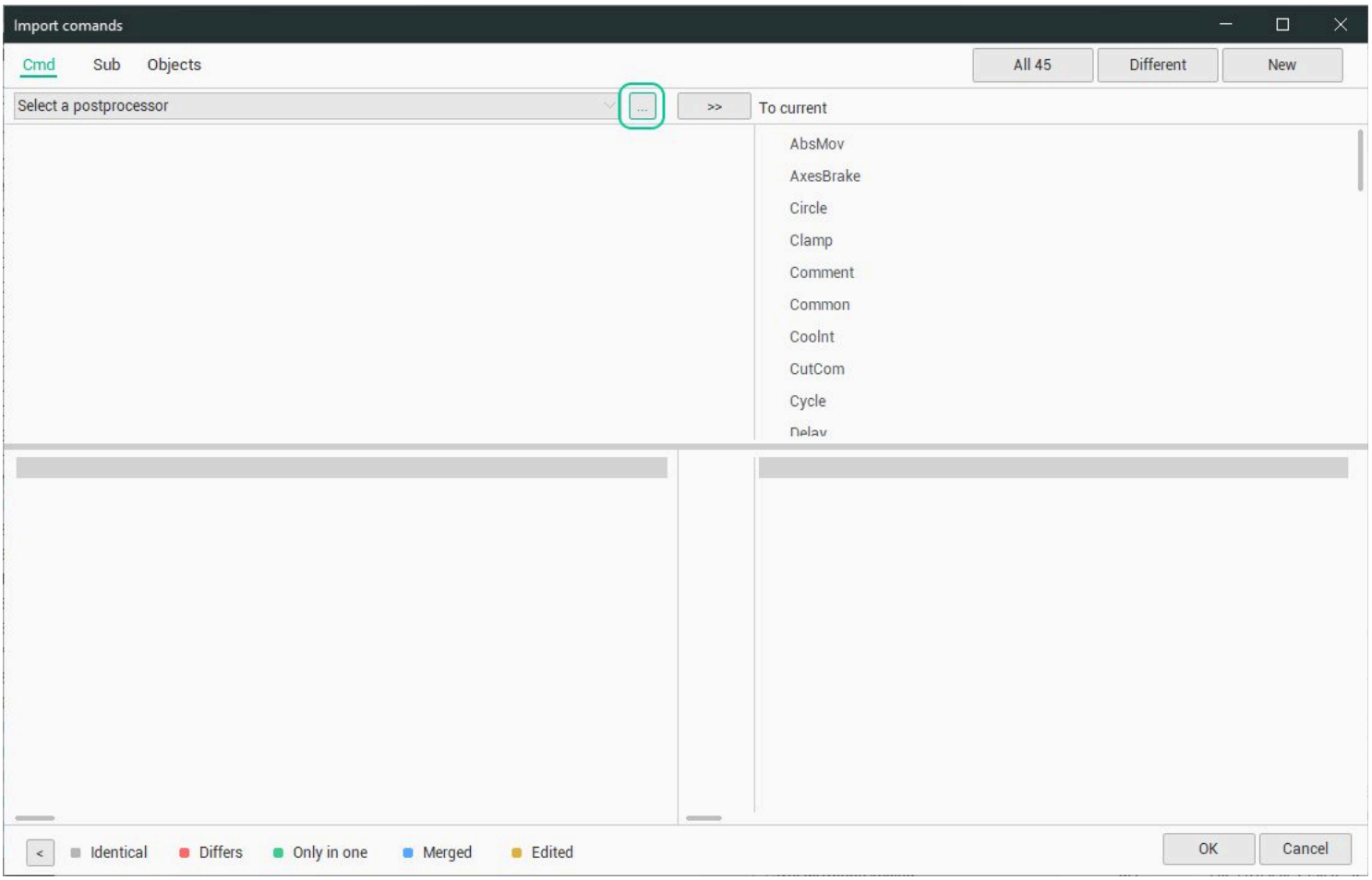
Импорт команд, подпрограмм и объектов

Окно импорта позволяет сравнивать текущий постпроцессор с другим постпроцессором и переносить из него команды, подпрограммы и объекты. Можно импортировать весь элемент или только отдельные отличающиеся фрагменты его кода.

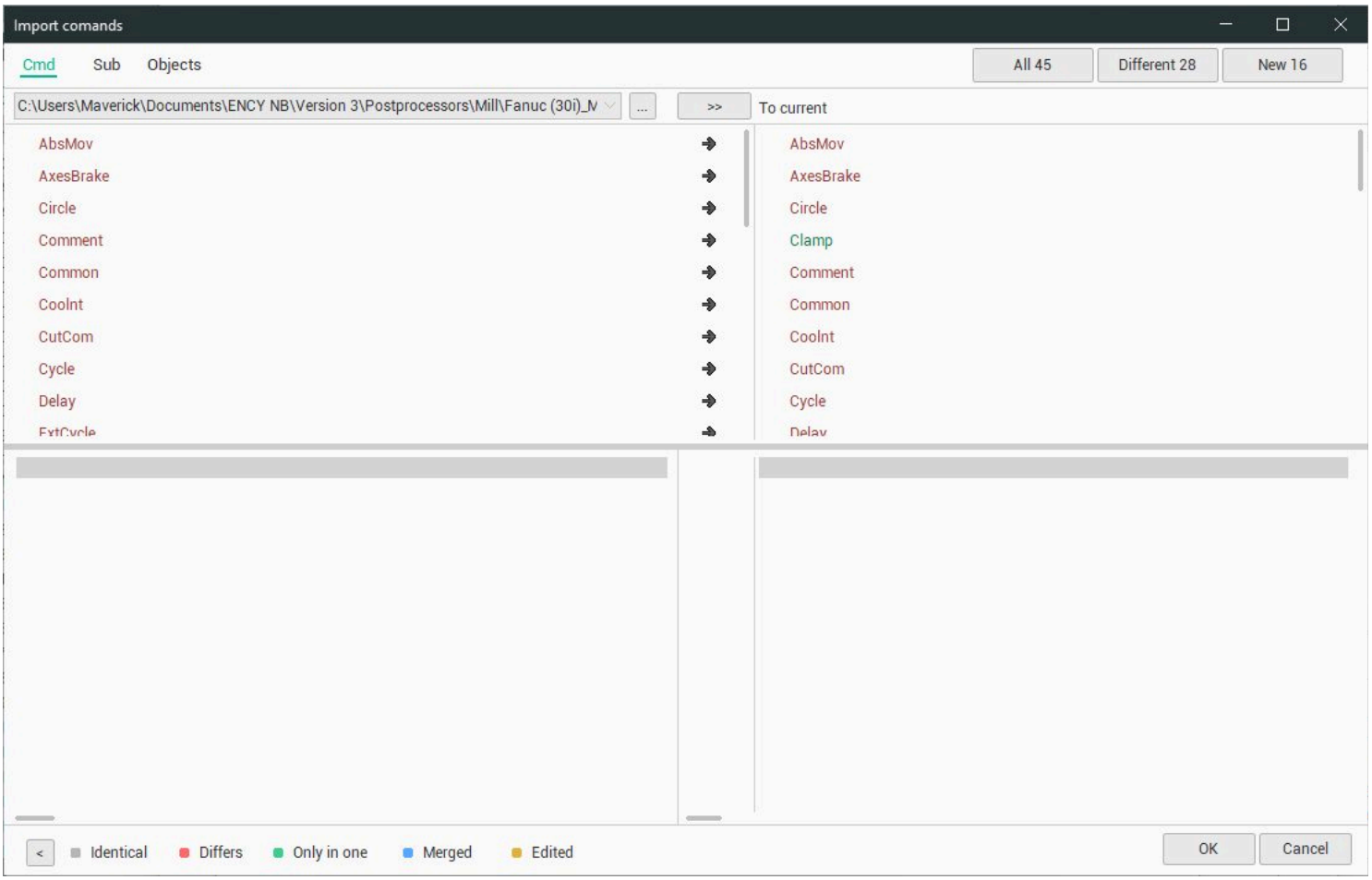
Откройте окно импорта, нажав кнопку со стрелкой в переключателе Команды, Подпрограммы или Объекты.



Нажмите кнопку выбора постпроцессора и укажите постпроцессор, из которого нужно импортировать элементы.



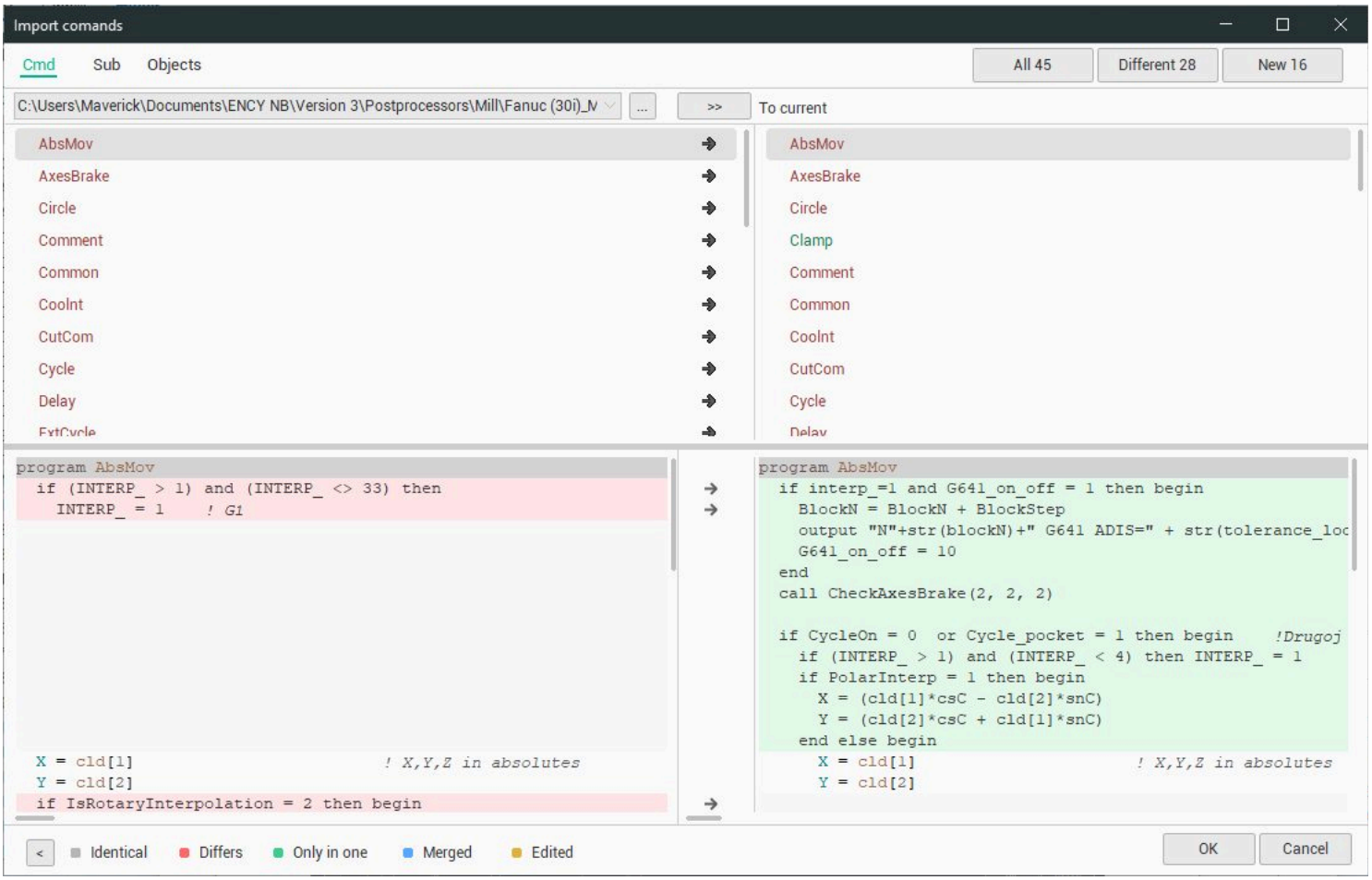
После загрузки постпроцессора в левой части окна отображаются его элементы, а в правой -- элементы текущего постпроцессора.



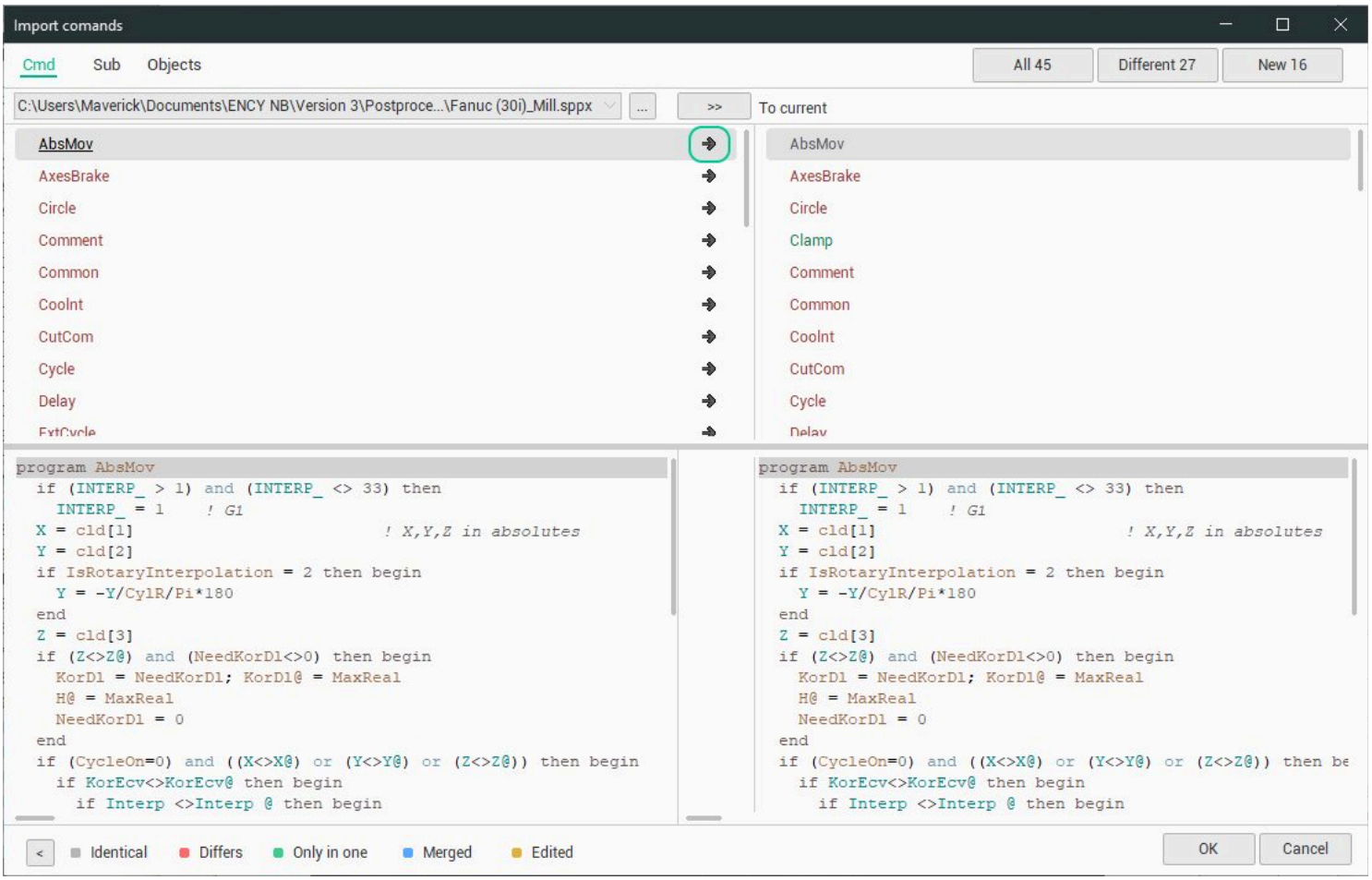
Цвет элемента показывает результат сравнения:

- красный -- элемент есть в обоих постпроцессорах, но его содержимое отличается;
- зеленый -- элементы идентичны;
- черный -- элемент отсутствует в другом постпроцессоре.

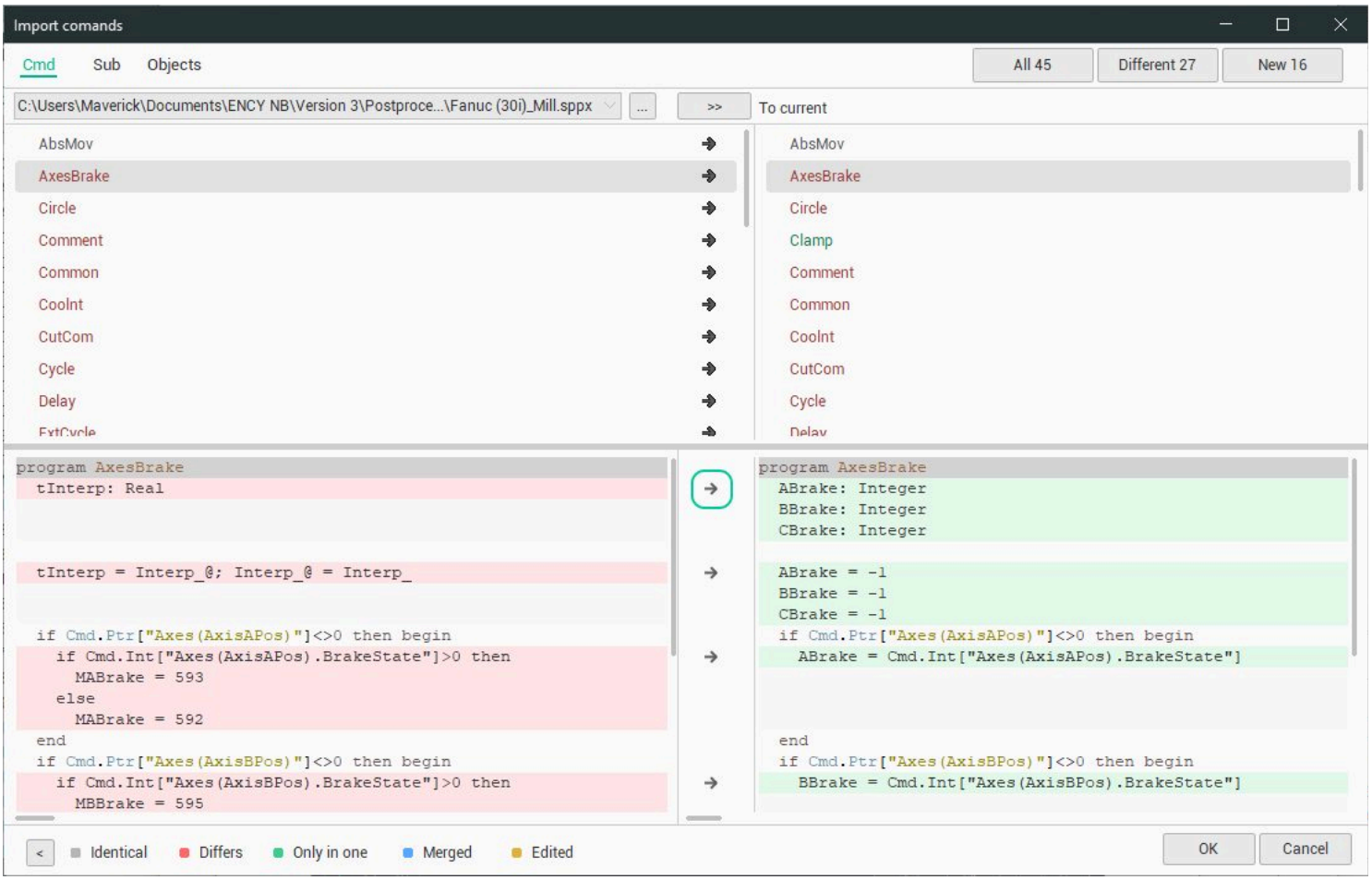
Выберите элемент, выделенный красным, чтобы просмотреть его код в обоих постпроцессорах.



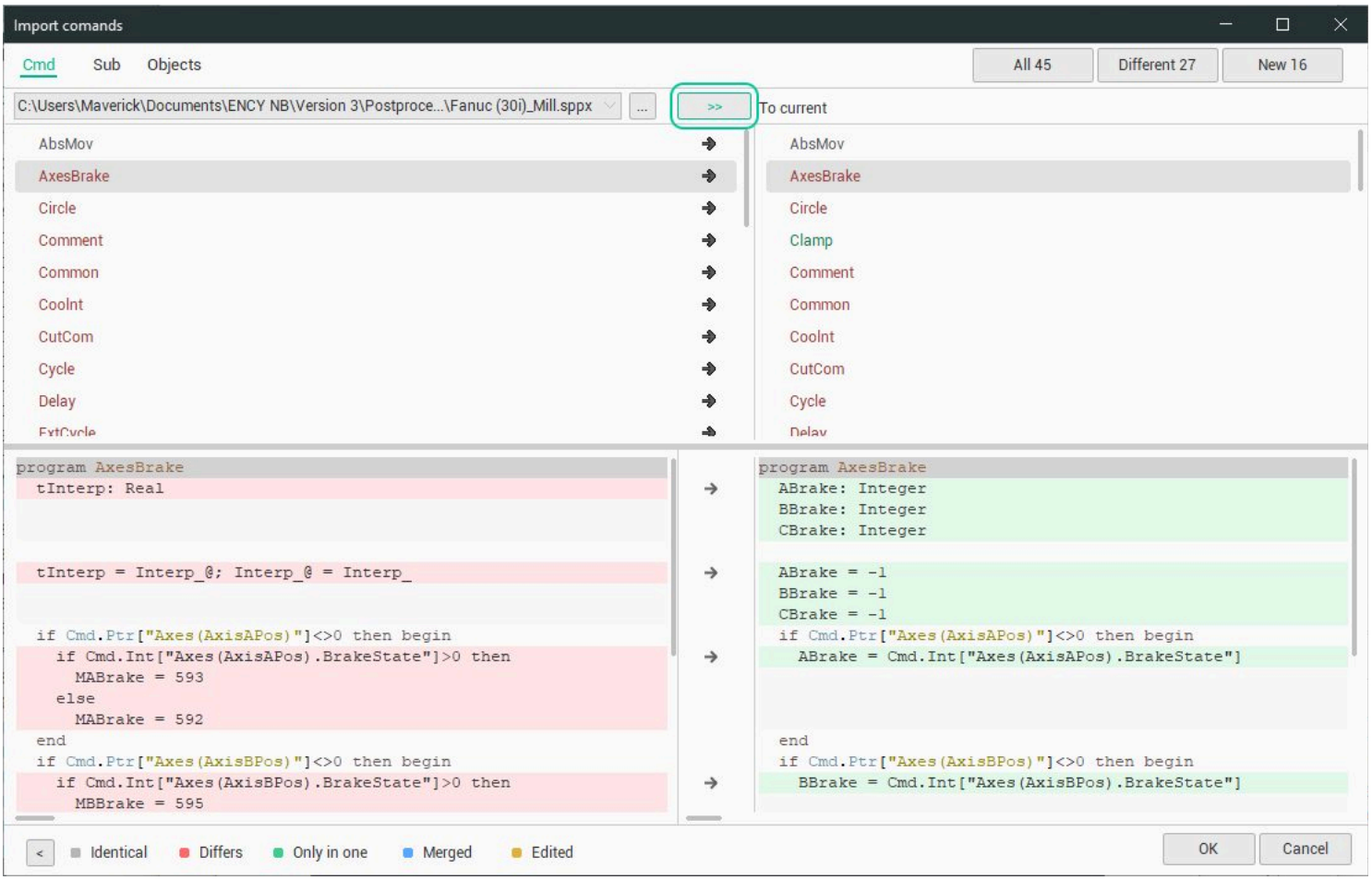
Чтобы полностью заменить элемент в текущем постпроцессоре версией из импортируемого постпроцессора, нажмите стрелку рядом с именем элемента.



Для переноса только одного отличающегося фрагмента нажмите стрелку рядом с этим фрагментом в области просмотра кода текущего постпроцессора.



Чтобы заменить все элементы текущего постпроцессора версиями из импортируемого, нажмите Merge all to current.



При необходимости можно загрузить другой постпроцессор, сравнить его элементы, а затем вернуться к ранее загруженному постпроцессору. Выберите нужный постпроцессор в раскрывающемся списке рядом с кнопкой загрузки.

Сопутствующие ссылки

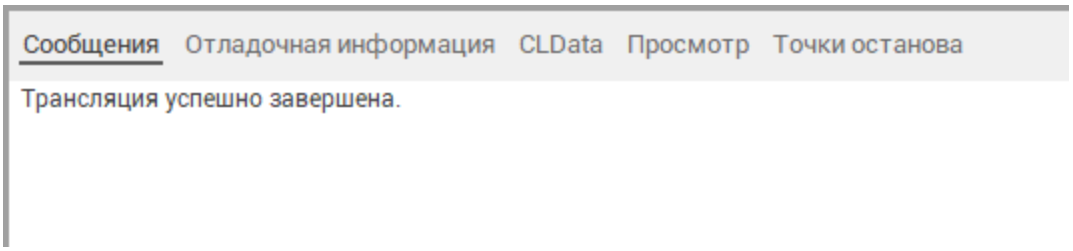
Главное окно

Трансляция программ обработки команд

Трансляцию можно запустить нажатием кнопки  на главной панели инструментов или нажатием комбинации [Ctrl-F9].

Примечание: Транслируются только активизированные программы.

Системные сообщения о процессе трансляции выводятся на странице сообщений в нижней части главного окна.



Ошибки, возникающие в результате трансляции программ, заносятся в список системных сообщений с префиксом [Ошибка]. Двойное нажатие левой клавишей мыши на такое сообщение загружает исходный текст программы и устанавливает курсор в позицию ошибки.

Сопутствующие ссылки

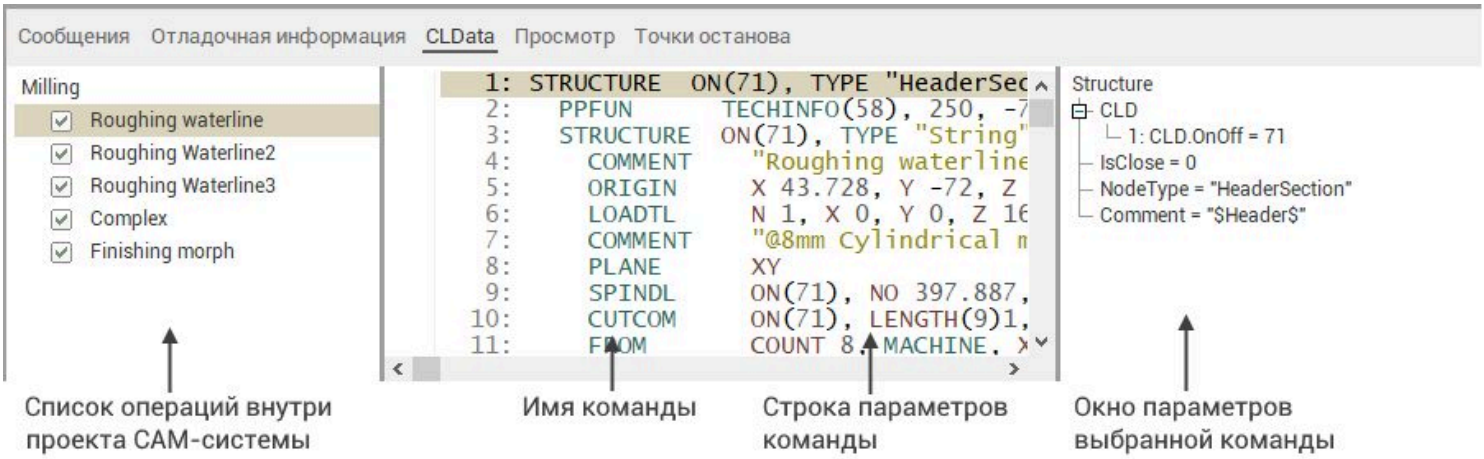
Главное окно

Работа с файлами технологических команд

Файлы технологических команд могут быть загружены из проекта CAM-системы (*.stcp, *.stcx). Нажатие кнопки или выбор соответствующего пункта в главном или контекстном меню загружает файл.

После загрузки файлов технологических команд из проекта CAM-системы имя проекта, содержащего эти файлы, отображается на главной панели инструментов.

В нижней части главного окна на странице CLData расположен список файлов технологических команд. Каждая строка списка соответствует операции CAM-системы. В формировании NC-кода участвуют только отмеченные флажком операции.



Список операций можно редактировать через контекстное меню. Нажмите правую кнопку мыши в поле списка, чтобы открыть контекстное меню. Меню содержит следующие пункты:

- Добавить – добавляет файл технологических команд;
- Удалить – удаляет выделенный файл;
- Удалить все – удаляет все файлы.
- Обновить CLData – запрашивает новые файлы CLData из проекта, открытого в CAM-системе.

В центре вкладки CLData расположено текстовое представление технологических команд (команд CLData). В нём отображается список команд для файла, выделенного в списке слева. Каждая технологическая команда в тексте CLDATA — это отдельная строка, состоящая из имени команды и её параметров. При двойном нажатии левой клавиши мыши на имени команды в редактор шаблонов загружается шаблон, соответствующий команде, а в редактор программ — программа, соответствующая команде.

Справа от списка команд расположено окно параметров технологической команды. Оно содержит список имён параметров и их значений для выделенной команды. Чтобы быстро получить значение параметра в программном коде, необходимо выделить соответствующий

параметр и выбрать пункт Добавить параметр в код в контекстном меню (открывается нажатием правой кнопки мыши) или сделать двойной щелчок левой кнопкой мыши. Параметры доступны через массив CLD или оператор Cmd в программах и через список параметров текущей команды в шаблонах.

Сопутствующие ссылки

Главное окно

Тестовая генерация управляющих программ

Тестовую генерацию управляющей программы можно выполнить нажатием кнопки  на главной панели инструментов или нажатием клавиши [F9]. Если в этот момент программы обработки технологических команд не оттранслированы, то трансляция будет запущена автоматически. При возникновении ошибок трансляции тестовый запуск не выполняется.

Исходными данными для генерации управляющей программы являются загруженные файлы технологических команд. Управляющая программа формируется на основании тех технологических файлов, которые отмечены флажками в списке.

Алгоритм генерации описан ниже:

1. Система считывает очередную команду из файла технологических команд.
2. Система анализирует код команды и в соответствии с параметрами команды заполняет предопределённый массив CLD и предопределённые переменные.
3. Если для команды активизирована соответствующая программа, то программа выполняется.
4. Если для команды активизирован соответствующий шаблон, то по шаблону формируется строка NC-кода.

Примечание: Таким образом, для любой команды могут быть активизированы и программа, и шаблон. В этом случае сначала выполняется программа, а затем шаблон. Если и шаблон, и программа не активизированы, то команда игнорируется.

5. Если в результате предыдущих действий сформирована строка NC-кода, то выполняется подпрограмма Filter. Обычно эта подпрограмма используется для замены некоторых элементов в кадре УП перед окончательным выводом в управляющую программу.

Соответствующие операторы формируют кадр УП из шаблона или из программы обработки. Если используется программа обработки, то в кадр выводятся только идентификаторы и значения тех регистров, которые изменились с момента вывода предыдущего кадра.

Управляющая программа сохраняется в окне текста управляющей программы (оно находится в правой части главного окна) и в файле с именем, заданным в поле имени управляющей программы. Если это поле пустое, то программа в файл не сохраняется. Если управляющая программа содержит подпрограммы, то они также могут отображаться в отдельных файлах; в этом случае появятся дополнительные вкладки, переключившись на которые можно просмотреть сгенерированные тексты подпрограмм.

Отладочная информация, формируемая в программах обработки, записывается в окно отладки.

Если в процессе генерации возникли ошибки, они записываются в окно системных сообщений.

После безошибочной генерации NC-кода устанавливается связь между NC-кодом и командами CLData. При двойном нажатии на имени команды в окне команд CLData выделяется соответствующая строка NC-кода. И наоборот, при двойном нажатии на строке кода в окне управляющей программы выделяется соответствующая команда CLData.

Сопутствующие ссылки

Главное окно

Отладка программ

Интегрированный отладчик позволяет контролировать выполнение программы. Программу можно выполнять по шагам, проверяя код программы и результат её выполнения. В процессе отладки можно входить в подпрограммы или выполнять подпрограмму за один шаг, контролировать значения переменных и просматривать отладочные сообщения. Отладчик кардинально сокращает время разработки постпроцессоров и облегчает поиск и устранение ошибок.

Функции отладчика:

- Запуск программы – [F9];
- Выполнить программу до курсора – [F4];
- Шаг с обходом (выполнить текущий оператор без входа в подпрограмму) – [F8];
- Установка/снятие точки останова – [Ctrl+F8];
- Шаг с заходом (выполнить текущий оператор с входом в подпрограмму) – [F7];
- Прервать отладку – [Ctrl+F2];
- Добавить новую переменную в список наблюдения – [Ctrl+F7];
- Вычислить выражение – [Ctrl+F4].

Перед запуском программы компилируются все изменённые подпрограммы. Если в процессе компиляции ошибок не найдено, то программа запускается.

Запустить выполнение программы можно любой из команд: [F4], [F7], [F8] или [F9].

[F4] выполняет программу до оператора в позиции курсора. Если текущая строка не содержит оператора, то программа выполняется до конца. Строка, на которой прервалось выполнение, является курсором отладчика и выделяется цветом.

Если отладка запущена, то [F7] и [F8] выполняют оператор программы, на котором находится курсор отладчика, иначе курсор отладчика устанавливается на первом операторе первой программы. Команда [F7], в отличие от [F8], позволяет войти в подпрограмму, если текущий оператор — CALL.

[Ctrl+F8] позволяет установить/снять точку останова. Когда выполнение программы достигает точки останова, оно приостанавливается с активацией среды отладки.

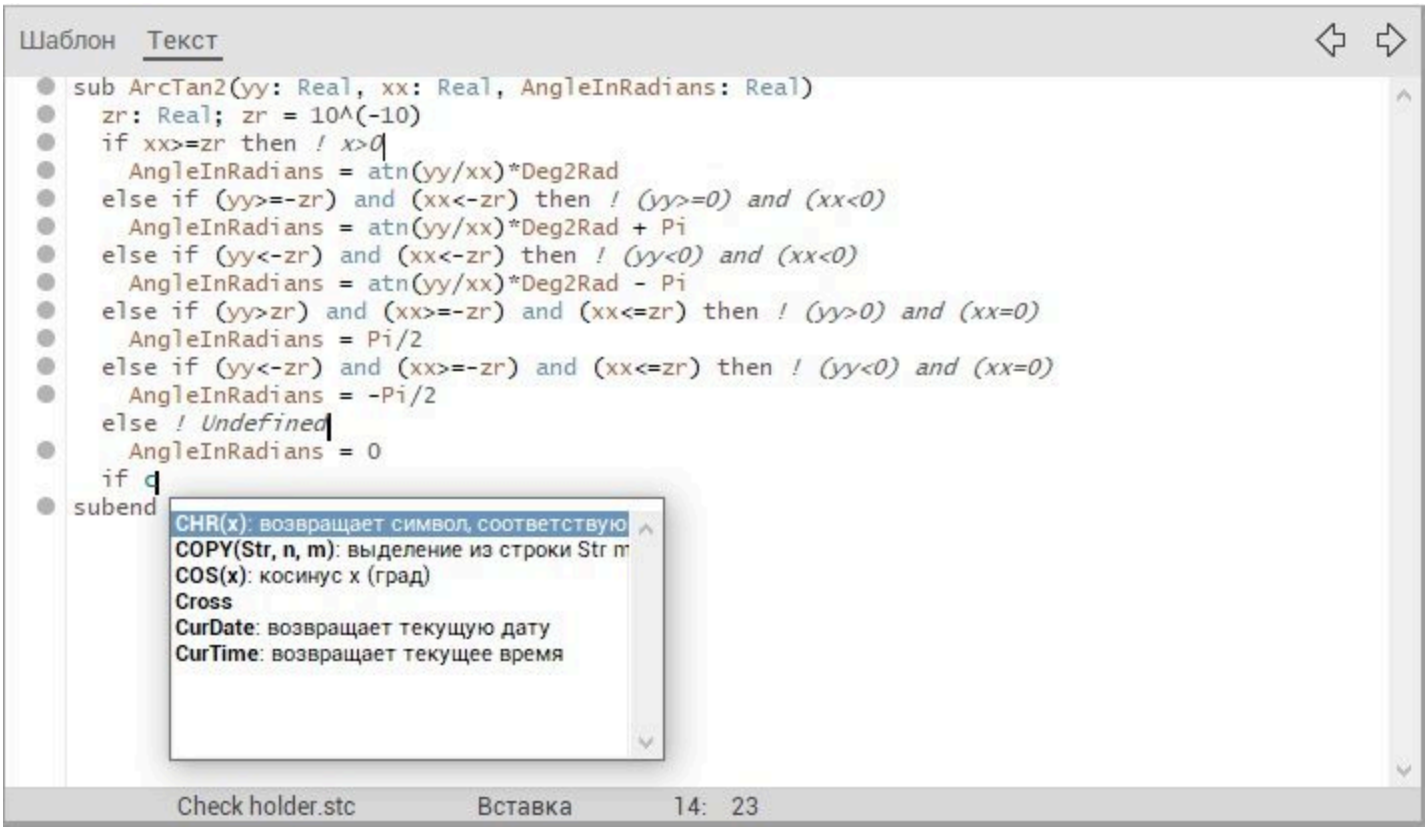
В любой момент процесса отладки можно прервать выполнение командой [Ctrl+F2] или продолжить выполнение без отладки командой [F9].

Изменение количества строк в отлаживаемой программе приводит к неправильной индикации текущего курсора отладчика.

[Ctrl+F7] позволяет добавить переменную или выражение в список наблюдения. Если [Ctrl+F7] нажата в режиме редактирования, то имя переменной берётся из текущей позиции курсора. В режиме отладки при наведении указателя мыши на переменную её значение отображается во всплывающей подсказке.

Для поиска слова дважды щёлкните по этому слову, и появится новая вкладка со списком всех упоминаний данного слова во всех командах.

При редактировании кода программ для ввода имён предопределённых функций и переменных удобно пользоваться всплывающей подсказкой. Её можно вызвать в любой момент редактирования нажатием комбинации клавиш [Ctrl+пробел]. Появившееся окно содержит список стандартных функций, совпадающих с уже введённой в редакторе частью слова, а также показывает краткое описание этих функций. При двойном щелчке мышью по одной из строк списка выделенная в ней функция добавляется в текущую позицию редактора.



Сопутствующие ссылки

Главное окно

Запрос параметров постпроцессора при первом использовании

Иногда дилер не знает некоторую специфическую информацию о станке пользователя. Можно организовать запрос данных. В этом случае у пользователя будет запрошена информация о системе ЧПУ, и введённые данные сохранятся в файле постпроцессора.

Файл постпроцессора *.sppx — это обычный XML-файл. Поэтому для организации запроса данных его можно отредактировать в простом текстовом редакторе (например, в Windows Notepad, а лучше в Notepad++, чтобы получить подсветку синтаксиса). В файле есть раздел CommonDefinitions (или узел в терминологии XML). Он показан ниже. Это текстовое представление информации, которую вы задаёте в окне информации о станке.

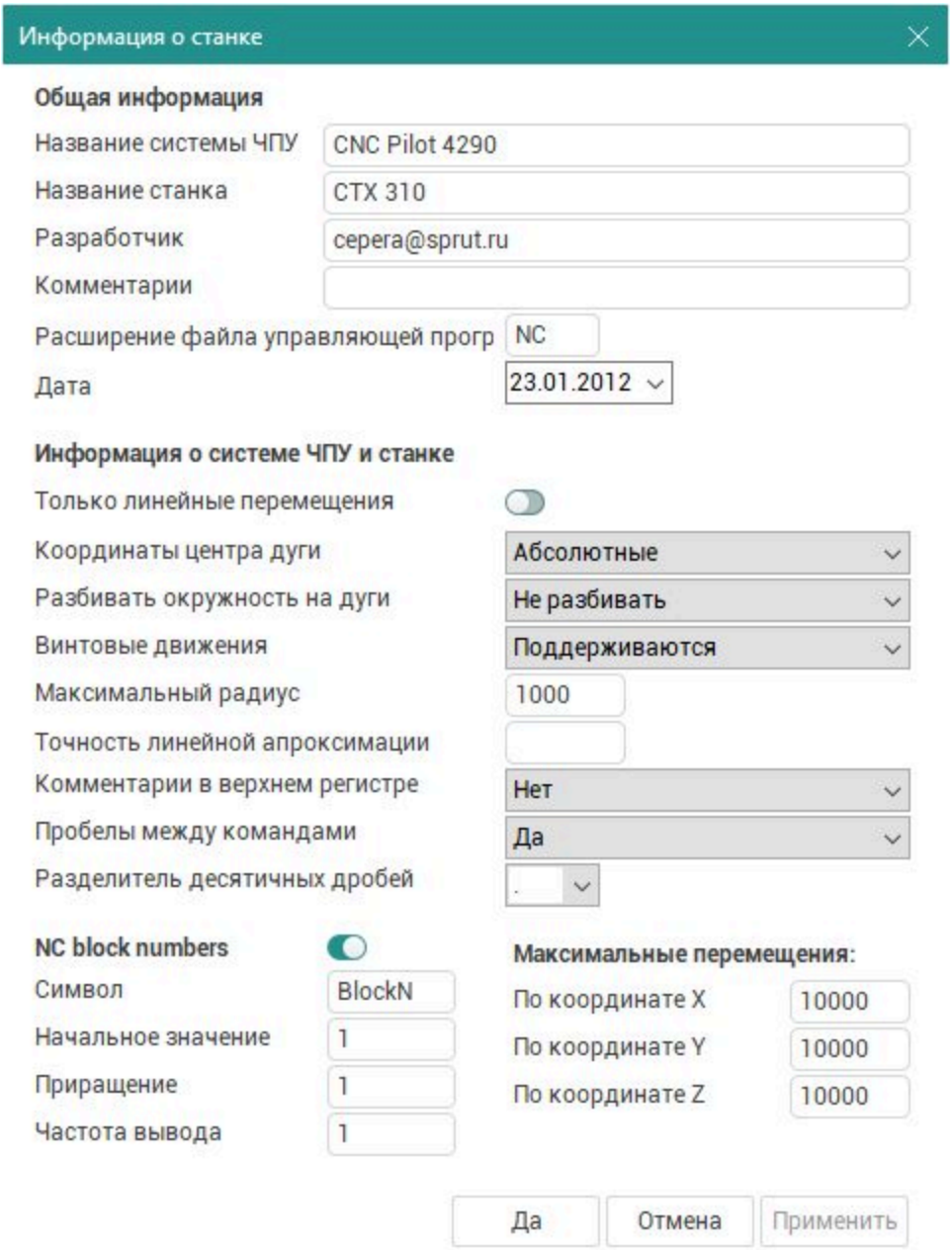
```
<CommonDefinitions>
    <LinearApproxPrecision IsSet="False">0.01</LinearApproxPrecision>
    <Circles>
        <Support>True</Support>
        <HelixSupport>True</HelixSupport>
        <Center>Absolute</Center>
        <Division>DoNotBreak</Division>
        <MaxRadius>10000</MaxRadius>
    </Circles>
    <XMax>100000</XMax>
    <YMax>100000</YMax>
    <ZMax>100000</ZMax>
    <UpperCaseComments>True</UpperCaseComments>
    <Spaces>False</Spaces>
    <DecimalSeparator>Point</DecimalSeparator>
    <BlocksNumbering>
        <Address>N</Address>
        <StartValue>100</StartValue>
        <StepValue>2</StepValue>
        <Frequence>0</Frequence>
    </BlocksNumbering>
</CommonDefinitions>
```

Можно добавить специальный атрибут `MakeQuery="True"` к любому из узлов. Тогда значение для этого конкретного параметра будет запрошено у пользователя сразу после открытия файла постпроцессора.

Например, если раздел содержит следующий текст:

```
    <CommonDefinitions>
        <LinearApproxPrecision IsSet="False"
MakeQuery="True">0.01</LinearApproxPrecision>
        <Circles>
            <Support MakeQuery="True">True</Support>
            <HelixSupport MakeQuery="True">True</HelixSupport>
            <Center MakeQuery="True">Absolute</Center>
            <Division MakeQuery="True">DoNotBreak</Division>
            <MaxRadius MakeQuery="True">10000</MaxRadius>
        </Circles>
        <XMax MakeQuery="True">100000</XMax>
        <YMax MakeQuery="True">100000</YMax>
        <ZMax MakeQuery="True">100000</ZMax>
        <UpperCaseComments MakeQuery="True">True</UpperCaseComments>
        <Spaces MakeQuery="True">False</Spaces>
        <DecimalSeparator MakeQuery="True">Point</DecimalSeparator>
        <BlocksNumbering>
            <Address MakeQuery="True">N</Address>
            <StartValue MakeQuery="True">100</StartValue>
            <StepValue MakeQuery="True">2</StepValue>
            <Frequence MakeQuery="True">0</Frequence>
        </BlocksNumbering>
    </CommonDefinitions>
```

Тогда сразу после открытия файла постпроцессора (включая случай запуска генерации G-кода напрямую из CAM-системы) появится следующий диалог:



При нажатии Сохранить данные сохранятся в файле постпроцессора, и в следующий раз диалог не появится. При нажатии OK управляющая программа будет сгенерирована с введёнными данными, но диалог появится в следующий раз снова. При нажатии Отмена файл постпроцессора не будет открыт (G-код не будет сгенерирован).

Ввод дополнительной информации

Иногда полезно, чтобы система запрашивала список дополнительных параметров, и пользователь должен указать значение каждого из этих параметров перед работой с постпроцессором. Это возможно. Список появится в диалоге информации о станке. Если файл постпроцессора содержит специально сформированный узел `InitialQuestions`, то появляется панель «Дополнительная информация» с выпадающими списками или полями ввода. После того как пользователь укажет значения в этих полях, глобальная переменная, соответствующая каждому параметру, инициализируется введённым значением.

Формат раздела **InitialQuestions** показан ниже:

```
    <InitialQuestions>
        <!-- Multiple <Question> nodes possible  -->
        <Question>
            <QuestionSentence>User visible text for the question</QuestionSentence>
            <VarID>Name_of_variable</VarID>
            <VarType>Type_of_variable</VarType> <!-- vtString, vtReal, vtInteger
types available -->
            <VarValue>
                <![CDATA[Default value of the variable]]>
            </VarValue>
            <EnumValues>
                <!-- None or multiple <EnumValue> nodes are possible  -->
                <EnumValue>
                    <Caption>User visible caption of first possible value</Caption>
                    <Value>
                        <![CDATA[Variable first possible value]]>
                    </Value>
                </EnumValue>
                <EnumValue>
                    <Caption>User visible caption of second possible value</Caption>
                    <Value>
                        <![CDATA[Variable second possible value]]>
                    </Value>
                </EnumValue>
            </EnumValues>
        </Question>
    </InitialQuestions>
```

Где

- "User visible text for the question" – сообщение для пользователя рядом с полем ввода.
- "Name_of_variable" – имя любой глобальной переменной, доступной в постпроцессоре (включая пользовательские переменные, определённые в обработчике Common).
- "Type_of_variable" – тип переменной. Может принимать одно из трёх значений: vtString для типа String, vtReal для типа Real и vtInteger для типа Integer.
- "Default value of the variable" – начальное значение, которое будет присвоено переменной.
- Если переменная должна принимать только ограниченный набор значений (выпадающий список), то определите непустой раздел "EnumValues".
- "User visible caption of first possible value" – понятное пользователю описание значения.
- "Variable first possible value" – любое значение, которое должно иметь тот же тип, что и глобальная переменная.
- Текст между `<!--` и `-->` — это просто комментарий в корректном для XML синтаксисе.

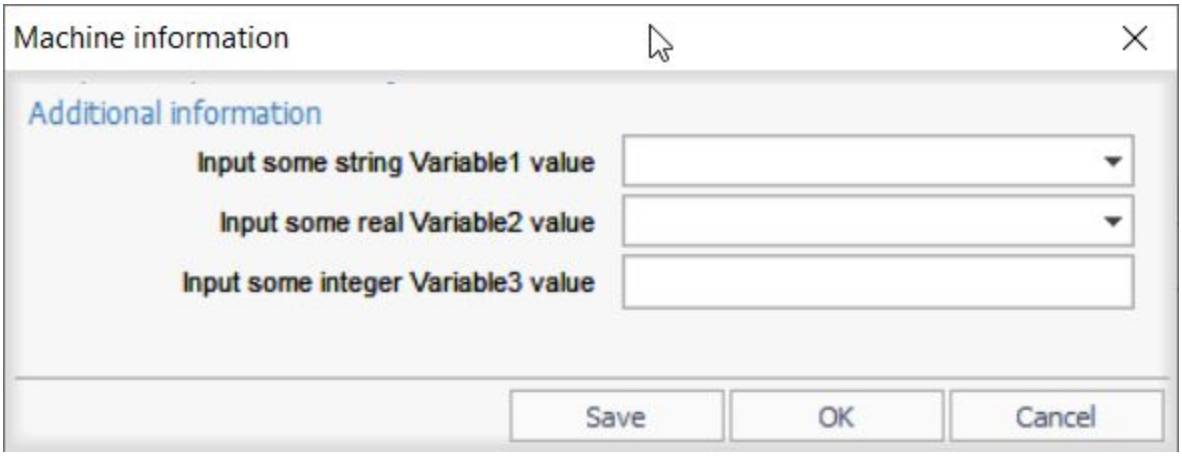
- Синтаксическая конструкция, начинающаяся с `<![CDATA[` и заканчивающаяся `]]>`, необходима для того, чтобы можно было вводить любые наборы символов без нарушения корректности XML.

Пример раздела InitialQuestions показан ниже:

```
<InitialQuestions>
    <Question>
        <QuestionSentence>Input some string Variable1 value</QuestionSentence>
        <VarID>Variable1</VarID>
        <VarType>vtString</VarType>
        <VarValue>
            <![CDATA[Default variable 1 value 1]]>
        </VarValue>
        <EnumValues>
            <EnumValue>
                <Caption>Value 1</Caption>
                <Value>
                    <![CDATA[Default variable 1 value 1]]>
                </Value>
            </EnumValue>
            <EnumValue>
                <Caption>Value 2</Caption>
                <Value>
                    <![CDATA[Default variable 1 value 2]]>
                </Value>
            </EnumValue>
            <EnumValue>
                <Caption>Value 3</Caption>
                <Value>
                    <![CDATA[Default variable 1 value 3]]>
                </Value>
            </EnumValue>
        </EnumValues>
    </Question>
    <Question>
        <QuestionSentence>Input some real Variable2 value</QuestionSentence>
        <VarID>Variable2</VarID>
        <VarType>vtReal</VarType> <!--vtString, vtReal, vtInteger-->
        <VarValue>
            <![CDATA[10.321]]>
        </VarValue>
        <EnumValues>
            <EnumValue>
                <Value>
```

```
                    <![CDATA[9.98]]>
                </Value>
            </EnumValue>
            <EnumValue>
                <Value>
                    <![CDATA[10.321]]>
                </Value>
            </EnumValue>
            <EnumValue>
                <Value>
                    <![CDATA[4.782]]>
                </Value>
            </EnumValue>
        </EnumValues>
    </Question>
    <Question>
        <QuestionSentence>Input some integer Variable3 value</QuestionSentence>
        <VarID>Variable3</VarID>
        <VarType>vtInteger</VarType>
        <VarValue>
            <![CDATA[20]]>
        </VarValue>
    </Question>
</InitialQuestions>
```

Стартовый диалог ввода будет следующим:



После нажатия кнопки сохранения раздел примет следующий вид:

```
<InitialQuestions>
    <Question MakeQuery="False">
        <QuestionSentence>Input some string Variable1 value</QuestionSentence>
        <VarID>Variable1</VarID>
        <VarType>vtString</VarType>
```

```
        <VarValue>
            <![CDATA[Default variable 1 value 2]]>
        </VarValue>
        <EnumValues>
            <EnumValue>
                <Caption>Value 1</Caption>
                <Value>
                    <![CDATA[Default variable 1 value 1]]>
                </Value>
            </EnumValue>
            <EnumValue>
                <Caption>Value 2</Caption>
                <Value>
                    <![CDATA[Default variable 1 value 2]]>
                </Value>
            </EnumValue>
            <EnumValue>
                <Caption>Value 3</Caption>
                <Value>
                    <![CDATA[Default variable 1 value 2]]>
                </Value>
            </EnumValue>
        </EnumValues>
    </Question>
    <Question MakeQuery="False">
        <QuestionSentence>Input some real Variable2 value</QuestionSentence>
        <VarID>Variable2</VarID>
        <VarType>vtReal</VarType>
        <VarValue>
            <![CDATA[9.98]]>
        </VarValue>
        <EnumValues>
            <EnumValue>
                <Caption>9.98</Caption>
                <Value>
                    <![CDATA[9.98]]>
                </Value>
            </EnumValue>
            <EnumValue>
                <Caption>10.321</Caption>
                <Value>
                    <![CDATA[10.321]]>
                </Value>
            </EnumValue>
            <EnumValue>
                <Caption>4.782</Caption>
```

```
                <Value>
                    <![CDATA[4.782]]>
                </Value>
            </EnumValue>
        </EnumValues>
    </Question>
    <Question MakeQuery="False">
        <QuestionSentence>Input some integer Variable3 value</QuestionSentence>
        <VarID>Variable3</VarID>
        <VarType>vtInteger</VarType>
        <VarValue>
            <![CDATA[12345]]>
        </VarValue>
        <EnumValues/>
    </Question>
</InitialQuestions>
```

И во время выполнения переменные будут иметь следующие значения:

- Variable1 = "Default variable 1 value 2";
- Variable2 = 9.98;
- Variable3 = 12345;

Сопутствующие ссылки

Главное окно

Шаблоны постпроцессора

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

Каждой команде CLData соответствует свой шаблон. Шаблоны используют специальный язык, благодаря которому они близки по виду к кадрам управляющей программы.

Например, пусть команда CLData такая:

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

и пусть для обработки этой команды нужно сформировать следующий кадр:

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

Шаблон, формирующий нужный кадр:

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

где:

- XT, YT, ZT — текущие координаты инструмента;
- 1 — тип интерполяции (позиционирование G0, линейная G1 или дуговая G2, G3 (для систем ЧПУ, отличных от G-кода, может быть иначе));
- FT — значение подачи.

В шаблонах можно использовать переменные и значения регистров, встроенный массив GMA, встроенные функции и выражения, дающие числовой результат. Также можно использовать специальные слова (переключатели и модификаторы), управляющие выводом кода в управляющую программу.

Сопутствующие ссылки

Состав шаблона

Переключатели шаблона

Модификаторы шаблона

Преобразование шаблона в подпрограмму

Визуальные средства формирования шаблона

Состав шаблона

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

```
элемент1 элемент2... элементN.
```

Сопутствующие ссылки

Шаблоны постпроцессора

Элемент шаблона

Регистры в шаблонах

Модификаторы

Выражения

Вложенный шаблон

Разделители элементов шаблона

Занесение значений в переменные из шаблона

Отложенные шаблоны

Массив GMA в шаблонах

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 заключён в квадратные скобки "[" и "]", то вместо него подставляется значение или переменная, стоящая в скобках. Например:

- Шаблон:

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

- Кадр УП:

```
100.12456 200
```

В примере значение переменной XT равно 100.12456.

В квадратных скобках допустимо использовать:

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

В управляющей программе числа могут иметь различные представления. Способ вывода числа определяется следующими параметрами:

- Десятичная точка — это поле может принимать значения:

    - Отсутствует,
    - Присутствует, если число имеет дробную часть,
    - Присутствует всегда;
- Ширина целой части — максимальное количество знаков для представления целой части числа;
- Ширина дробной части — количество знаков для представления дробной части числа;
- Лидирующие нули и Незначащие нули — определяют режим вывода нулей до и после числа;
- Знак — определяет режим вывода знака числа. Доступны следующие варианты:
    - Нет,
    - Только "-",
    - Только "+",
    - Всегда "+" и "-".

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

- Лидирующие нули и Незначащие нули — отсутствуют;
- Десятичная точка — присутствует, если число имеет дробную часть;
- Знак — присутствует, если число отрицательное;
- Ширина целой части и Ширина дробной части — позволяют вывести число без округления;
- Идентификатор элемента выводится как текст.

Сопутствующие ссылки

Состав шаблона

Регистры в шаблонах

Регистры используются для задания формата вывода чисел в соответствии с требованиями систем ЧПУ. Регистр в постпроцессоре объединяет следующие свойства:

- Идентификатор регистра;
- Текущее значение регистра;
- Предыдущее значение;
- Формат вывода значения;
- Имя регистра.

Подробнее см. раздел Определение структуры и формата кадра (формирование списка регистров).

Если перед открывающей квадратной скобкой указать имя или идентификатор регистра, то при выводе значения будет использован формат вывода числа этого регистра. 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
```

При разборе шаблона предполагается, что слово, стоящее перед открывающей квадратной скобкой, определяет регистр. Исполняющая система ищет регистр сначала по имени, а если регистр не найден — по идентификатору. Регистры перебираются в порядке, определённом в списке регистров. Поэтому если в списке есть несколько регистров с одинаковым идентификатором, будет использован первый.

Если регистр не найден, то значение выводится в формате по умолчанию, а перед числом 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 регистр найден, то старому значению регистра присваивается текущее значение, а текущему — новое значение. После этого исполняющая система сравнивает старое и текущее значения, и если они не равны, то идентификатор регистра и значение выводятся в кадр управляющей программы в соответствии с заданным форматом вывода числа.

Например, пусть текущее 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
```

Старое и новое значения регистра равны, поэтому значение регистра F в кадр управляющей программы не выводится.

Сопутствующие ссылки

Состав шаблона

Модификаторы

Иногда требуется вывести новое значение регистра независимо от того, изменилось оно или нет, и наоборот — изменить значение регистра, но не выводить его в текущий кадр. Для этого предназначены модификаторы On и Off.

Модификатор записывается в квадратных скобках после значения через запятую.

Модификатор On. Если модификатор указан, то значение регистра выводится в кадр управляющей программы в любом случае.

Например, пусть текущее значение регистра `G_PLANE` равно 17:

- Шаблон:

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

- Кадр УП:

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

Можно вывести текущее значение регистра без занесения нового значения. Для этого нужно записать On без значения. Например:

- Шаблон:

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

- Кадр УП:

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

Модификатор Off. Если модификатор указан, то новое значение заносится в регистр, но в кадр управляющей программы регистр не выводится. Например:

- Шаблон:

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

- Кадр УП:

```
G49G80M5M9
```

Модификатор Off можно использовать без значения. Это нужно, чтобы исключить вывод текущего значения регистра в кадр управляющей программы без занесения нового значения.

Сопутствующие ссылки

Состав шаблона

Выражения

В шаблонах можно записывать выражения. Выражение — это математическая формула. Результатом выражения является число.

Синтаксически математическое выражение представляет собой комбинацию чисел, числовых переменных и числовых функций, разделённых знаками математических операций и круглыми скобками. Простейшие примеры математического выражения — это число и числовая переменная.

В языке допустимы следующие математические операции:

- `+` — сложение;
- `-` — вычитание;
- `*` — умножение;
- `/` — деление;
- `^` — возведение в степень.

Следует помнить, что два знака операций не могут следовать один за другим.

Допустимы следующие математические функции:

- `SIN(x)` — синус угла x (в градусах);
- `COS(x)` — косинус угла x (в градусах);
- `TAN(x)` — тангенс угла x (в градусах);
- `ATN(x)` — арктангенс угла x (в градусах);
- `ASIN(x)` — арксинус угла x (в градусах);
- `ACOS(x)` — арккосинус угла x (в градусах);
- `SQR(x)` — квадратный корень из x;
- `ABS(x)` — абсолютное значение x;
- `SGN(x)` — знак числа x;
- `ROUND(x, y)` — округление x до y знаков после запятой;
- `LOG(x)` — десятичный логарифм x;
- `LN(x)` — натуральный логарифм x.

В выражении доступны предопределённые переменные и функции, а также все параметры текущей команды CLDATA, 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 регистр X был найден в списке регистров, и формат вывода числа был взят из свойств регистра. В выражении второго элемента присутствует параметр, переданный через массив CLD, — `CLD.Y` (равный 0.617), умноженный на 2. В третьем элементе используется предопределённая переменная FEED. Эта переменная содержит текущее значение подачи.

Сопутствующие ссылки

Состав шаблона

Вложенный шаблон

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

- Шаблон:

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

- Кадр УП:

```
G43 Z0.123
```

Значение регистра Z 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 порядке, соответствующем списку регистров. Например:

- Шаблон:

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

- Кадр УП:

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

Изменённый регистр G в соответствии со списком регистров помещён в начало кадра.

Сопутствующие ссылки

Состав шаблона

Разделители элементов шаблона

В общем случае строка шаблона состоит из нескольких элементов. Между элементами можно вставлять пробелы. Эти пробелы не влияют на строку управляющей программы.

Для управления пробелами между словами в строке можно использовать параметр Расставлять пробелы между командами. Если параметр включён, 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 строке пробел добавлен между первым и вторым элементами. Количество пробелов перед первым элементом, а также между вторым и третьим, третьим и четвёртым элементами не изменилось.

Если параметр Расставлять пробелы между командами выключен, то все пробелы между элементами удаляются независимо от их количества. Например:

- Шаблон:

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

- Кадр УП:

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

Из строки управляющей программы удалены все пробелы.

Сопутствующие ссылки

Состав шаблона

Занесение значений в переменные из шаблона

При необходимости из шаблона можно присвоить значения любым предопределённым переменным или переменным COMMON. Значения можно присваивать числовым и строковым переменным. Строковой переменной можно присвоить только константу. Числовой переменной можно присвоить любое выражение, в том числе с функциями и другими переменными.

Значения переменных можно присваивать в любой строке шаблона. Для такой строки шаблона кадр управляющей программы не формируется. Разделителем между переменными служит знак `;`. Например:

- Шаблон:

```
INTERP=0;XP=XT;YP=YT-Y1+2;CLDATA$="текст"
G_INTERP[INTERP] X[XT] Y[YT] Z[ZT] ([CLDATA$])
```

- Кадр УП:

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

Сопутствующие ссылки

Состав шаблона

Отложенные шаблоны

Иногда необходимо использовать текущие параметры текущей команды при обработке другой команды. Есть два способа решить эту задачу. Первый способ — сохранить параметры во временные переменные и использовать их позже в нужной команде. Второй, более удачный способ — использование отложенных шаблонов. Текущие параметры таких шаблонов сохраняются, а выполнение шаблона откладывается до обработки требуемой команды. При выполнении отложенного шаблона текущие параметры заменяются сохранёнными.

Отложенный шаблон должен быть заключён в фигурные скобки `{` и `}`. Внутри фигурных скобок обязательно указывается имя одной из команд CLDATA. Шаблон выполняется при появлении этой команды. Строки, расположенные до этой команды, выполняются перед выполнением команды, а строки, расположенные после указанной команды, — после выполнения команды. Например:

- Шаблон команды SafPos:

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

- Шаблон команды LoadTL:

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

- Шаблон команды AbsMov:

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

- Команды CLData:

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

- Кадр УП:

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

В этом случае в момент выполнения команды SafPos создаётся отложенный шаблон. Значения `CLD.X`, `CLD.Y`, `CLD.Z` команды SafPos сохраняются, и выполнение шаблона откладывается до обработки команды LoadTL. Значения XT, YT, ZT не сохраняются, так как 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:

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

В завершение выполняется:

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

Если перед закрывающей фигурной скобкой указать идентификатор Keep, то отложенный шаблон будет выполняться каждый раз при обработке требуемой команды (в предыдущем примере это LoadTL). Например:

- Шаблон команды SafPos:

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

В этом примере отложенный шаблон будет выполняться каждый раз при обработке команды LoadTL.

Сопутствующие ссылки

Состав шаблона

Массив GMA в шаблонах

Предопределённый массив GMA предназначен для удобного программирования постпроцессора команды MULTIGOTO как в виде программы технологической команды, так и в виде шаблона. Массив GMA, как и массив CLD, инициализируется перед обработкой команды. Каждый элемент массива GMA соответствует оси в кинематической схеме станка.

Синтаксис использования массива GMA в шаблонах отличается от кода программы постпроцессора: при использовании в шаблоне ключевое слово "GMA" и скобки опускаются. Для доступа к свойству элемента массива GMA при программировании шаблона команды MULTIGOTO укажите имя оси в кавычках и имя свойства без кавычек, разделив их точкой. Например, чтобы вывести в управляющую программу текущее значение координаты с именем AxisXPos, следует использовать шаблон `X["AxisXPos".Vn]`.

Ниже приведены примеры обращения к элементам массива GMA:

- `"<AxisName>".OutFlag` — наличие координаты в текущей команде MULTIGOTO.
- `"<AxisName>".Vn` — текущее значение координаты.
- `"<AxisName>".Vp` — предыдущее значение координаты.
- `"<AxisName>".Axis` — имя координаты в кинематической схеме станка.
- `"<AxisName>".Reg` — имя регистра, связанного с координатой.
- `"<AxisName>".TurnCount` — количество полных оборотов (360 градусов) при текущем перемещении поворотной оси.
- `"<AxisName>".Dir` — направление вращения.

AxisName — имя координаты станка.

В следующем примере в управляющую программу записываются значения координат X, Y и Z, переданные командой MULTIGOTO, для осей AxisXPos, AxisYPos и AxisZPos соответственно.

```
X["AxisXPos".Vn] Y["AxisYPos".Vn] Z["AxisZPos".Vn]
```

Сопутствующие ссылки

Состав шаблона

Управление шаблонами

На этой странице нет содержимого. См. вложенные разделы

Переключатели шаблона

С каждым шаблоном связаны три переключателя:

1. Вывод предыдущего кадра. Если переключатель включён, то до разбора шаблона формируется кадр управляющей программы. В него выводятся все регистры, старые и текущие значения которых не равны.
2. Опросить все регистры. Если переключатель включён, то помимо регистров, перечисленных в шаблоне, в кадр включаются все регистры, старые и текущие значения которых не равны. Если такие регистры есть (это возможно, например, если в предыдущем шаблоне был использован модификатор Off), то такой регистр помещается в кадр в соответствии с порядком в списке регистров. Например:

- Список регистров:

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

- Шаблон команды Plane:

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

- Шаблон команды 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. Если переключатель выключен, то все регистры, записанные в шаблоне, будут изменены, но строка управляющей программы не выводится.

Сопутствующие ссылки

Шаблоны постпроцессора

Преобразование шаблона в подпрограмму

Возможна ситуация, когда возможностей шаблона недостаточно. Чтобы использовать более мощные и гибкие подпрограммы без ручного программирования шаблона, предназначена функция Преобразовать шаблон в подпрограмму. Для преобразования шаблона используйте кнопку M -> Prg. Кнопка расположена рядом с переключателями шаблона.

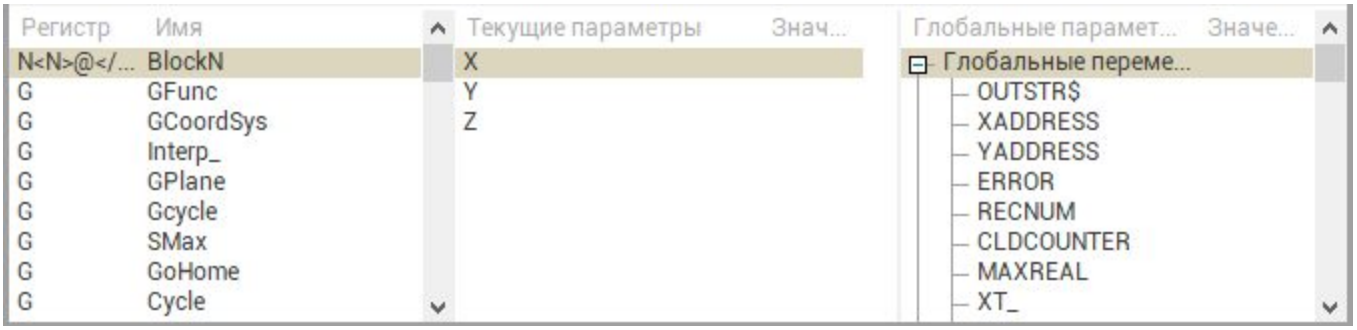
Чтобы отменить преобразование, необходимо через буфер обмена перенести строку шаблона из редактора подпрограмм в редактор шаблона и выставить переключатели.

Сопутствующие ссылки

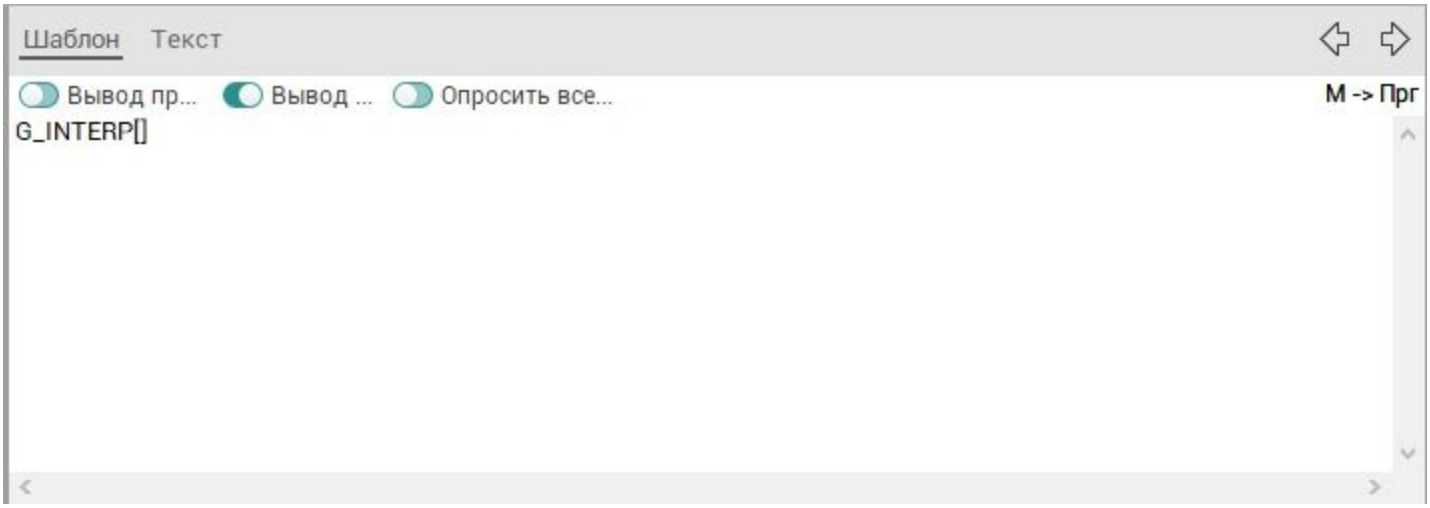
Шаблоны постпроцессора

Визуальные средства формирования шаблона

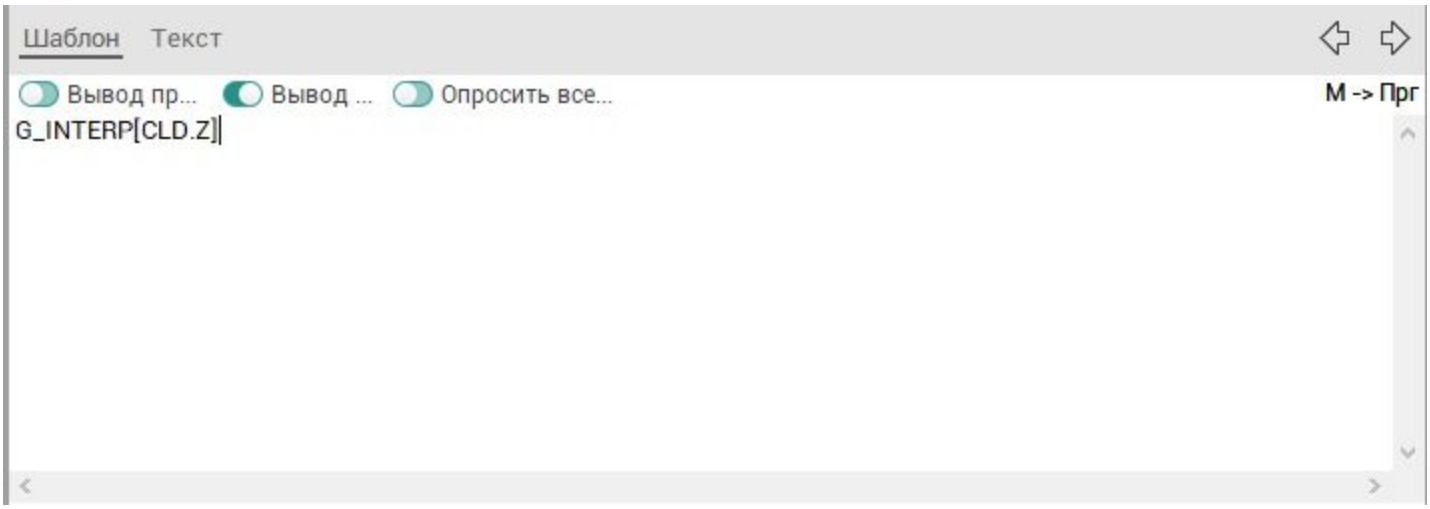
Для удобного редактирования шаблонов предназначены три списка. Эти списки расположены под редактором шаблона. Левый список — это список регистров. Второй список — это список параметров текущей команды CLData. Правый список — это список глобальных переменных и предопределённых функций.



Двойной щелчок в списке регистров вставляет имя выбранного регистра в позицию курсора. После имени регистра добавляются символы `[]`, а курсор перемещается в позицию между ними.



Двойной щелчок в списке текущих параметров вставляет имя параметра CLData в позицию курсора. К параметру добавляется префикс CLD.



Сопутствующие ссылки

Шаблоны постпроцессора

Описание языка

На этой странице нет содержимого. См. дочерние разделы

Основные понятия

На этой странице нет содержимого. См. дочерние разделы

Условные обозначения

При описании форматов операторов в этом документе используются следующие правила и условные обозначения:

- прописные буквы используются для обозначения зарезервированных слов;
- строчные буквы используются для обозначения имён, переменных, числовых или арифметических выражений;
- языковые конструкции обозначаются символами `. .`;
- необязательные части операторов обозначаются символами {..};
- альтернативные части операторов разделяются символом |.

Программы обработки технологических команд, комментарии

Постпроцессор генерирует код управляющей программы, используя файлы технологических команд (CLData). Упрощённо процесс постпроцессирования можно описать так:

1. Из файла считывается очередная команда.
2. Команда анализируется, определяются её код и свойства, инициализируются встроенный массив CLD и предопределённые переменные.
3. Если программа обработки технологической команды включена в постпроцессоре и её код задан, то система вызывает эту программу. Программа технологической команды должна содержать код, формирующий строки управляющей программы, соответствующие данным CLData.
4. Если для команды включён и задан шаблон, то он используется для формирования кода управляющей программы.

Программы обработки технологических команд пишутся на проблемно-ориентированном языке и состоят из математических выражений и функций, операторов ввода/вывода, операторов управления ходом программы, вызовов подпрограмм и операторов вывода кода управляющей программы.

Программа обработки технологической команды начинается с ключевого слова PROGRAM, за которым следует имя программы, и заканчивается ключевым словом END. Имя программы совпадает с именем команды CLData, которую она обрабатывает. Параметры технологической команды передаются через встроенный массив CLD и специальный оператор Cmd. Подробное описание команд CLData и значений их параметров приведено в главе Описание технологических команд.

Код программы состоит из операторов и комментариев. В одной строке можно записать несколько операторов, разделив их символом `;`. Перенос кода оператора на следующую строку допустим, если его части логически незавершены.

Комментарии удобны для пояснения более сложных частей программы. Чтобы поставить комментарий, введите символ `!`; весь текст после этого символа считается комментарием и игнорируется компилятором. Чтобы закомментировать/раскомментировать несколько строк, выделите нужные строки и нажмите `Ctrl+!` или `Ctrl+/`.

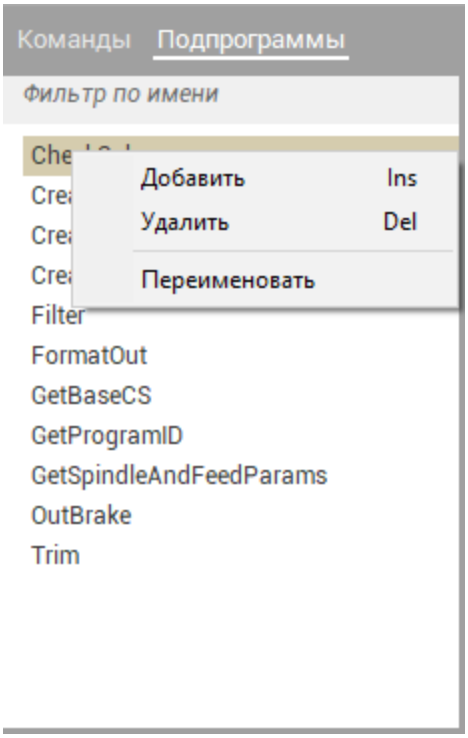
Также можно закомментировать любую часть текста, заключив её в пару фигурных скобок `{ - }`. Такой способ позволяет комментировать как произвольную часть строки, так и несколько последовательных строк, в том числе строки, закомментированные символом `!`.

При выполнении постпроцессора первой вызывается программа COMMON. Затем выполняется программа PARTNO, а программа FINI выполняется всегда последней.

Программа COMMON служит для определения глобальных переменных: все переменные, объявленные в программе COMMON, доступны в любой части постпроцессора. Для собственных нужд любая программа может использовать переменные, не объявленные в COMMON и потому не являющиеся общедоступными. Такие переменные называются локальными.

Подпрограммы

Вкладка Subprograms содержит список пользовательских подпрограмм и подпрограмму Filter. Для добавления, удаления или переименования подпрограмм используйте контекстное меню. Код подпрограммы можно просматривать и редактировать в редакторе кода. В редакторе отображается код выбранной подпрограммы.



Имя подпрограммы может быть любым допустимым строковым идентификатором при условии, что оно не совпадает с именем технологической программы и не используется другой подпрограммой.

Принципы определения и синтаксис подпрограмм описаны в разделе Описание языка. Подпрограммы.

Сопутствующие ссылки

Главное окно

Понятие оператора языка

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

```
PRINT "symbol" ! печать литерной строки "symbol"
JUMP 1 ! переход на метку номер 1
```

Операторы языка могут содержать идентификаторы, числа, литерные строки и служебные символы.

Набор символов

При написании операторов языка используются следующие символы:

- `a..z`, `A..Z` – прописные и строчные буквы латинского алфавита;
- `0..9` – цифры от 0 до 9;
- `_` – знак подчёркивания;
- – пробел;
- `,` – запятая;
- `=` – равно;
- `;` – точка с запятой;
- `:` – двоеточие;
- `[` – левая квадратная скобка;
- `]` – правая квадратная скобка;
- `"` – двойные кавычки;
- `(` – левая круглая скобка;
- `)` – правая круглая скобка;
- `+` – плюс;
- `-` – минус;
- `*` – звёздочка;
- `/` – 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 – это непрерывная последовательность букв, цифр и символов `_`, `@` и `$`, начинающаяся с буквы. Прописные и строчные буквы в идентификаторах не различаются.

Число – это непрерывная последовательность цифр, содержащая не более одной десятичной точки. Перед числом допустимы знаки `+` или `-`. По умолчанию число считается положительным.

Литерные строки – это произвольные последовательности символов, заключённые в двойные кавычки.

Переменные

Переменная – проименованная область памяти, используемая для обращения к данным, находящимся в этой области. Иными словами, переменная – это имя, с которым может быть связано некоторое значение.

Переменные могут иметь разные типы. Тип переменной определяет множество значений, которые могут храниться в переменной, и множество операций, которые могут быть с ней выполнены. Переменные в генераторе постпроцессоров могут иметь один из следующих типов:

- Integer (целое число) – используйте Integer для представления целых чисел. Диапазон значений переменной Integer: от -2147483648 до 2147483647.
- Real (вещественное число) – используйте Real для хранения чисел с дробной частью. Диапазон значений, представимых переменной типа Real: 5.0*10^-324 .. 1.7*10^308. Тип Real имеет 15–16 значащих цифр.
- String (символьные строки) – используйте String для хранения последовательности до ~2^31 символов.
- Array (массивы или векторы) – сложный тип, представляющий последовательность значений простого типа.

Переменные могут иметь локальную или глобальную область видимости. Переменные с локальной областью видимости доступны только в текущей программе или подпрограмме. К глобальным переменным можно обращаться в любой программе или подпрограмме постпроцессора. Чтобы определить глобальную переменную, объявите её в специальной программе COMMON. Все переменные, объявленные в программе COMMON, являются глобальными. В одной строке допустимо объявлять несколько переменных одного типа.

Язык генератора постпроцессоров позволяет определять переменные в любом месте кода программы или подпрограммы, однако хорошим стилем считается определение используемых в программе переменных в её начале. Для объявления переменной используйте следующий синтаксис:

`Variable name 1`, {`Variable name 2`}: Тип переменной

Здесь:

- Имя переменной – уникальный в пределах текущей области видимости идентификатор переменной; идентификатор должен начинаться с буквы или знака подчёркивания (_) и не может содержать пробелов; после первого символа допустимы буквы, цифры, знаки подчёркивания и символы `@` и `$`. Идентификаторы нечувствительны к регистру. В качестве идентификаторов нельзя использовать зарезервированные слова, имена операторов, имена программ технологических команд и имена подпрограмм.

- Тип переменной – один из типов: Integer, Real, String, Array.

Ниже приведён пример объявления и использования переменных.

Пример

```
sub Sub1
  nn: Integer ! Объявление переменной целого типа
  xx, yy, zz: Real ! Объявление переменных вещественного типа
  ss: String ! Объявление переменной строкового типа
  nn = 2 ! Присвоение значения переменной
  xx = 7.43 ! Присвоение значения переменной
  yy = 12.6 ! Присвоение значения переменной
  rr = nn * sqr(xx^2 + yy^2) ! использование переменных в выражении
  ss = "Result = " ! Присвоение значения переменной
  print ss, rr ! отладочный вывод значения переменной
subend
```

Массивы

Массив представляет собой индексированную совокупность элементов одного типа. Массив характеризуется своим именем, типом хранимых элементов, размером (количеством хранимых элементов) и нумерацией элементов. Массив можно представить в виде таблицы:

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

В таблице выше N – размер массива (количество элементов в массиве). Максимальный размер массива не ограничен, однако использовать чрезмерно большие массивы не рекомендуется. Все массивы индексируются от 1 до размера массива.

Язык постпроцессора поддерживает как одномерные, так и двумерные массивы. Элементы массива могут быть любого простого типа: (Integer), (Real), (String) или (Record).

Массивы можно использовать везде, где допустимы простые переменные того же базового типа. Для обращения к элементу массива необходимо указать имя массива и индекс элемента; индексом может быть любое числовое выражение, не выходящее за границы диапазона:

```
<ArrayIdentifier>[<ItemIndex>, {<ItemIndex>}]
```

Для объявления переменной-массива используется следующий синтаксис:

Имя переменной-массива: array Размер массива of Базовый тип массива

Где

- Имя переменной-массива – любой допустимый идентификатор;
- Размер массива – положительное целое число;
- Базовый тип массива – один из простых типов (Integer, Real или String).

Для объявления двумерного массива используется следующий синтаксис:

Имя переменной-массива: array [Первый размер массива, Второй размер массива] of Базовый тип массива

Пример

```
V: array 10 of Real ! Объявление массива из 10 вещественных элементов
V[5] = 10.67 ! Присвоение значения пятому элементу массива
i = 7
! Использование массива в выражениях
i = 7
V[10] = 15*sqr((sin(V[i]))^2 + (cos(V[i+1]))^2)


V: array [3, 3] of Real ! Объявление двумерного массива

for i = 1 to 3 do for j = 1 to 3 do V[i, j] = i*10+j
```

Массив с заранее заданным количеством элементов называется статическим. Массив, размер которого может меняться в ходе выполнения программы, называется динамическим. Чтобы объявить динамический массив, опустите размер массива в его объявлении. Динамические массивы индексируются с 1. Размер динамического массива устанавливается равным максимальному индексу элемента, к которому производилось обращение.

Ниже приведён пример объявления и использования массива строк.

Пример

```
ArrayS: array of string ! Объявление массива
! Заполнение элементов массива
ArrayS[1] = "Hello," ! Нет необходимости указывать
ArrayS[2] = " World" ! количество элементов –
ArrayS[3] = "!" ! память выделяется автоматически
! Использование массива
output ArrayS[1] + ArrayS[2] + ArrayS[3]
```

Математические выражения и функции

Математическое выражение – это 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, разделённых знаками математических операций и круглыми скобками. Простейшие примеры математических выражений – число и числовая переменная. В языке допустимы следующие математические операции:

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

Следует помнить, что два знака операций нельзя располагать один за другим. Для управления порядком вычислений используйте круглые скобки `(` и `)`.

Также в выражениях можно использовать предопределённые функции и переменные.

Предопределённые переменные и функции

Язык постпроцессора предоставляет следующие встроенные и предопределённые функции и переменные:

- Стандартные математические функции;
- Функции работы со строками;
- Служебные функции и переменные;
- Функции и операторы работы с CLData.

Стандартные математические функции

Реализованы следующие математические функции:

- `SIN(x)` – синус угла x (в градусах).
- `COS(x)` – косинус угла x (в градусах).
- `TAN(x)` – тангенс угла x (в градусах).
- `ATN(x)` – арктангенс x, результат в градусах.
- `ASIN(x)` – арксинус x, результат в градусах.
- `ACOS(x)` – арккосинус x, результат в градусах.
- `SQR(x)` – квадратный корень из x.
- `ABS(x)` – модуль x.
- `SGN(x)` – знак x. Возвращает "-1", если x меньше нуля; возвращает "+1", если x больше нуля; возвращает "0", если x равно нулю.
- `ROUND(x, y)` – возвращает значение x, округлённое до y знаков после запятой.
- `LOG(x, y)` – возвращает логарифм y по основанию x.
- `LG(x)` – возвращает десятичный логарифм x.
- `LN(x)` – возвращает натуральный логарифм x.

Здесь x и y – аргументы функции. Ими могут быть числовая константа, переменная, функция или любое выражение, дающее числовое значение.

Сопутствующие ссылки

Предопределённые переменные и функции

Функции работы со строками

Реализованы следующие функции работы со строками:

- `CHR(x)` – возвращает символ, соответствующий заданному значению ASCII x.
- `ORD(Str)` – возвращает порядковое значение первого символа строки Str.
- `LEN(Str)` – возвращает длину строки Str (количество символов в строке).
- `POS(Pat, Str)` – возвращает индекс первого вхождения строки Pat в строку Str. Возвращает 0, если строка Pat не найдена в строке Str.
- `NUM(Str)` – преобразует строку Str в числовое представление; если строка некорректна, функция возвращает 0.
- `STR(n)` – преобразует число n в строковое представление.
- `COPY(Str, n, m)` – возвращает подстроку строки Str, содержащую m символов и начинающуюся с индекса n.
- `UPCASE(Str)` – возвращает копию строки Str в верхнем регистре.
- `REPLACE(<string variable>, <looked string>, <string for replace>)` Оператор замены подстроки в строке REPLACE.

Где

- Str, Pat – выражения, дающие строковое значение (переменные, литерные константы, функции или выражения);
- n, m – числовые значения.

Сопутствующие ссылки

Предопределённые переменные и функции

Оператор замены подстроки в строке REPLACE

Служебные функции и переменные

Язык постпроцессора предоставляет ряд служебных функций и переменных, использование которых упрощает разработку постпроцессора.

- Функции и операторы работы с CLData.
- Регистры. Каждому регистру постпроцессора соответствует переменная, имя которой совпадает с именем регистра. В этой переменной хранится текущее значение регистра. Регистровые переменные имеют тип Real.
- Старые значения регистров. Переменная, имя которой совпадает с именем регистра и заканчивается символом @, служит для хранения предыдущего значения регистра. Например, X@ – предыдущее значение регистра X. Эта переменная инициализируется при выводе кадра в управляющую программу.
- Массив RGS. Массив RGS позволяет получить доступ к регистрам по их индексу в списке. Это бывает полезно в некоторых случаях (например, в многоканальных постпроцессорах, где для каждого канала может быть свой набор регистров и нужно быстро переключаться между ними) вместо использования имени регистра.
- Массив CLD. Массив CLD – это массив вещественных чисел, инициализируемый системой параметрами текущей команды CLData.
- RecNum возвращает количество параметров текущей технологической команды, занесённых в массив CLD. Индекс верхнего элемента массива CLD равен RecNum.
- Массив GMA. Используется для обработки параметров технологической команды MULTIGOTO.
- Переменная CLData$. Инициализируется системой текстовыми данными технологической команды, если параметр команды является текстовой строкой. CLData$ хранит переданное значение до тех пор, пока оно не будет переопределено другой текстовой технологической командой. Тип CLData$ – String.
- Переменная OutStr$. В неё заносится текущая строка кода УП после выполнения команды FORMBLOCK или OUTBLOCK. OutStr$ имеет тип String.
- Переменная BlockStep содержит значение приращения при автоматической нумерации кадров УП. Автоматическая нумерация кадров включается в диалоге Информация о станке. Если автонумерация включена, то при формировании каждого нового кадра его номер увеличивается на BlockStep. Если задать BlockStep равным 0, автоматическая нумерация кадров отключается.

- Строковая переменная NCName$ содержит имя файла управляющей программы (без пути и расширения).
- Строковая переменная NCPath$ содержит полное имя файла управляющей программы, включая путь и расширение.
- Строковая переменная SPPName$ содержит имя файла постпроцессора без пути и расширения.
- Строковая переменная SPPPath$ содержит полный путь к файлу постпроцессора, включая расширение.
- CurDate – функция возвращает строку с текущей датой.
- CurTime – функция возвращает строку с текущим временем.
- CurISODate – функция возвращает строку с текущими датой и временем в формате ISO 8601.
- Переменная Error – целочисленная переменная, значение которой устанавливается равным коду последней ошибки, если во время выполнения программы произошла ошибка.
- Вещественные переменные Xt, Yt, Zt – координаты текущего положения инструмента. Эти переменные инициализируются системой непосредственно перед обработкой команд CIRCLE, FROM, ABSMOV, GOHOME.
- Вещественные переменные Xp, Yp, Zp – координаты предыдущего положения инструмента. Эти переменные инициализируются системой непосредственно перед обработкой команд CIRCLE, FROM, ABSMOV, GOHOME.
- Переменные Xc, Yc, Zc – координаты центра окружности (инициализируются системой перед вызовом программы CIRCLE).
- Interp – текущий режим интерполяции (значение этой переменной присваивается автоматически перед выполнением программ CIRCLE, FEDRAT, ABSMOV, RAPID, GOHOME). Набор значений переменной Interp:
  - 0 – ускоренное перемещение,
  - 1 – линейная интерполяция,
  - 2 – круговая интерполяция по часовой стрелке,
  - 3 – круговая интерполяция против часовой стрелки.
- ToolRad – радиус инструмента (инициализируется системой перед выполнением LOADTL).

- ArcPlane – текущая плоскость круговой интерполяции (значение присваивается непосредственно перед вызовом программы PLANE); переменная может принимать одно из следующих значений:
    - 33 – XY,
    - 37 – YZ,
    - 41 – XZ,
    - 133 – YX,
    - 137 – ZY,
    - 141 – ZX.
- Feed – текущее значение подачи (переменная устанавливается перед вызовом FEDRAT).
- Переменные TlComp, TrComp – значения коррекции на длину и на радиус соответственно (устанавливаются перед выполнением команд CUTCOM и LOADTL).
- Переменные FromX, FromY, FromZ – координаты исходной точки (устанавливаются перед командой FROM).
- FlagIn возвращает:
    - 1 – если следующее пересечение обходится изнутри;
    - 0 – если следующее пересечение обходится снаружи;
    - -1 – если тип обхода определить невозможно (например, два соседних кадра образуют коллинеарные отрезки при линейной интерполяции).
- Cross – функция возвращает 0, если текущий элемент траектории сопрягается со следующим элементом, и 1, если элементы пересекаются. Как правило, два отрезка всегда пересекаются, а отрезок и дуга могут сопрягаться или пересекаться. Эту функцию можно использовать для реализации коррекции.
- NextToolNum{(Mode)} – возвращает номер следующего инструмента. Необязательный параметр Mode определяет поведение после последней смены инструмента. Если Mode равен 0 (значение по умолчанию, когда параметры не заданы), то после последней команды смены инструмента возвращается -1. Если Mode равен 1, то возвращается номер первого инструмента в проекте.
- ToolChange – возвращает признак смены инструмента (будет ли далее в списке команд CLData команда LOADTL):
    - 1 – смены инструмента ещё не было, но она будет в следующих командах либо команда смены инструмента обрабатывается в данный момент;
    - 0 – смена инструмента уже была или обрабатывается в текущей команде;
    - -1 – текущая смена инструмента является последней в списке команд CLData.

Таблица ниже иллюстрирует работу функции ToolChange.

Список 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

- WorkingMode – возвращает режим работы генератора постпроцессоров:
  - 1 – генератор постпроцессоров запущен CAM-системой во время выполнения операции при включённом режиме «Моделирование по G-коду». Если в CAM-системе выполняются сразу все операции, то PARTNO и FINI являются первой и последней технологическими командами соответственно. Если операции выполняются по одной, команды PARTNO и FINI не формируются;

- 0 – генератор постпроцессоров запущен как отдельное приложение либо при генерации управляющей программы из CAM-системы на вкладке Machining.

Сопутствующие ссылки

Предопределённые переменные и функции

Массив GMA

Записи

`Record` – ключевое слово для описания переменных-структур. Синтаксис объявления переменной типа `Record`:

```
RecVarName: Record
  RecFieldName1: Type1
  RecFieldName2: Type2
  .....
  RecFieldNameN: TypeN
End
```

Где

- RecVarName – имя переменной;
- RecFieldName1...RecFieldNamen – имена полей переменной;
- TypeX – тип поля записи. Может быть любым простым типом, записью или статическим массивом перечисленных типов.

Пример

```
Rec1, Rec2: record
  X, Y, Z: real
  P: array [3,3] of integer;
  S1, S2: string
end
RR: array [10] of record
  X, Y, Z: real
end
Rec11: record
  field1: integer
  field2: array 10 of string
  field3: record
    nestedField1: real
    nestedField2: array 5 of integer
  end
end
```

Использование записей

Доступ к полям записи осуществляется с помощью оператора доступа «.».

Чтобы присвоить значение полю записи, используйте следующее выражение:

```
<RecordName>.<FieldName1>{.<NestedFieldName2>...} = x
```

Пример

```
Rec1.X = 1; Rec1.Y=2
Rec1.P[1,1] = 0
Rec1.S1 = "W100"
for i=1 to 10 do begin
   RR[i].X = i
   RR[i].Y = 0
   RR[i].Z = 50
end
rec3.field3.nestedField2[1]=33
```

Значение, хранящееся в поле, можно использовать так:

```
x = <RecordName>.<FieldName1>{.<NestedFieldName2>...}
```

Пример

```
x1=Rec1.X; y1=Rec1.Y
z2=RR[2].Z
```

Объекты

Переменные этого типа представляют собой сложную структуру, состоящую из полей и методов. Объекты описывают структуру динамической переменной, которая появляется только в процессе выполнения программы. Такая переменная является экземпляром объекта.

Свойства – это переменные типа REAL, INTEGER, STRING, RECORD или массив. Свойства объекта – это элементы данных, которые копируются в каждый экземпляр объекта.

Метод – это процедура или функция, связанная с объектом. Внутри методов доступны только свойства того экземпляра объекта, для которого был вызван метод. Функция отличается от процедуры тем, что возвращает результат. Её можно использовать в операторах присваивания, в математических и строковых выражениях, в операторе print, в логических выражениях. Результат работы функции в её описании находится в предопределённой переменной `result`.

Объявление полей производится после объявления объекта, описание методов – после объявления полей.

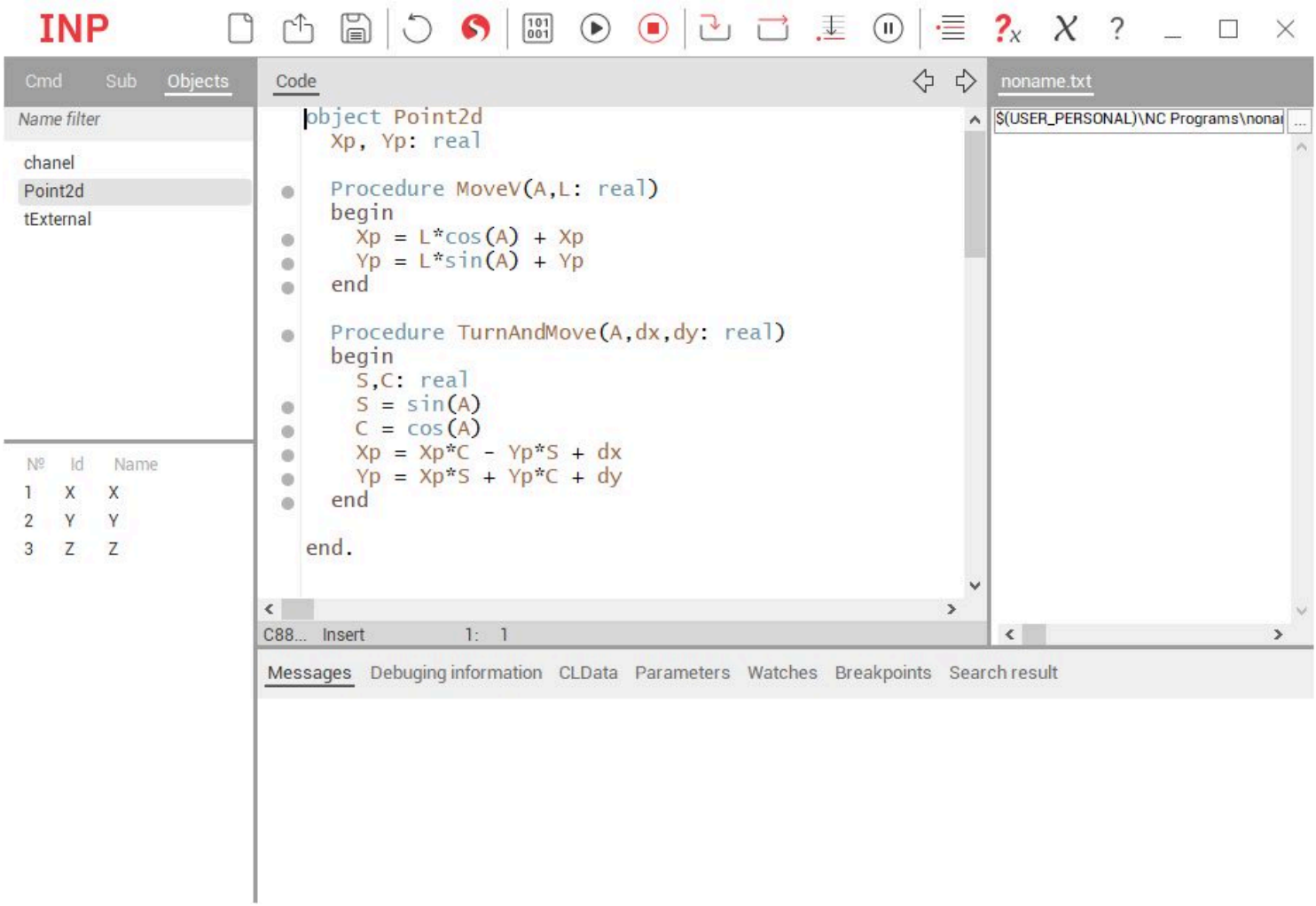
Предопределённая переменная `self` – это скрытый параметр каждого метода объекта. Она позволяет методу обратиться к объекту. Это особенно важно, когда параметры метода совпадают с переменными объекта, предназначенными для хранения этих переданных значений. Пример:

```
object sample
  xx1, yy1, zz1: real;
procedure init(xx1, yy1, zz1: real)
begin
  self.xx1=xx1; self.yy1=yy1; self.zz1=zz1
end
```

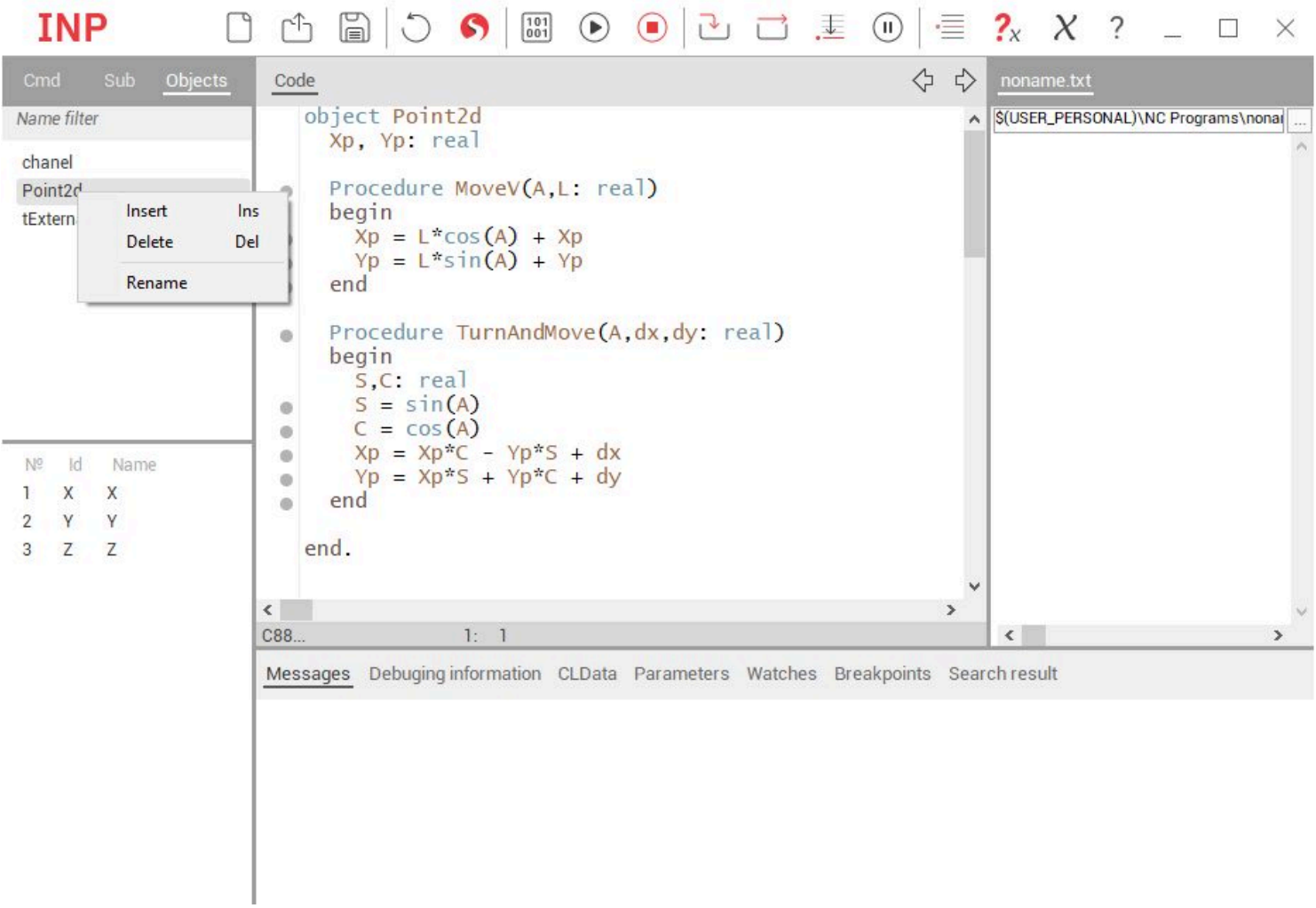
Пример передачи объекта в подпрограмму:

```
object sample2
procedure proc
begin
  call subproc(self)
end
```

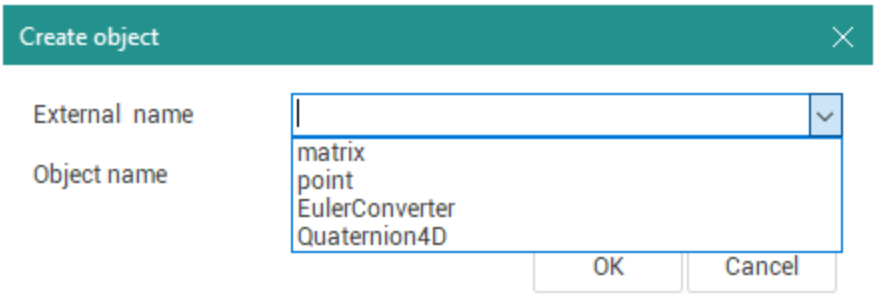
Описание объекта завершается командой `end.`.



Создание нового объекта, удаление или переименование существующего объекта выполняется через контекстное меню в списке объектов.



При создании нового объекта появится окно:



Здесь External name – имя внешней библиотеки, вся реализация которой выполнена на Delphi. Object name – имя, под которым объект будет доступен в системе.

На данный момент доступны:

- EulerConverter – пространственные повороты с использованием углов Эйлера.
- Matrix – трёхмерная матрица.
- Point3D – трёхмерная точка.
- Quaternion4d – кватернион (4D-вектор).
- NCChannel – решает задачу одновременного управления несколькими каналами.

- StringHelper – базовые функции для работы со строками, датой и временем.

Примечание: вы можете создавать наследников от представленных объектов, добавляя в них свои параметры, но только после уже имеющихся параметров!

Объявление экземпляра объекта производится в командах или подпрограммах аналогично массивам и структурам.

Пример объявления и использования экземпляра объекта:

```
pp: point2D
pp.xp=10
pp.yp=10
pp.MoveV(45, 10)
pp.TurnAndMove(45, 15, 25)
print pp.xp, "; ", pp.yp
```

Объекты можно присваивать целиком, как обычные переменные. Продолжая предыдущий пример:

```
pp2: point2d
pp2=pp
```

Также, как и обычные переменные, объекты могут быть параметрами процедур, функций и подпрограмм:

```
DrawPoint(pp)
```

EulerConverter

Euler Converter — 3D-конвертер вращения.

Параметры

- RotatingSequence: integer, варианты:
  - 1 — XYZ
  - 2 — XZY
  - 3 — YXZ
  - 4 — YZX
  - 5 — ZXY
  - 6 — ZYX
  - 7 — XYX
  - 8 — XZX
  - 9 — YXY
  - 10 — YZY
  - 11 — ZXZ
  - 12 — ZYZ
  - 13 — AB
  - 14 — AC
  - 15 — BA
  - 16 — BC
- MovableOrFixed: integer:
  - 1 — вращение вокруг подвижных осей
  - 0 — вращение вокруг фиксированных осей
- AngleInDegrees: integer
  - 1 — градусы
  - 0 — радианы

Методы

- Init — создать объект Euler Converter
- AnglesToMatrix — создать матрицу переноса по 3 углам и 3D-точке
- MatrixToAngles — получить 3 угла и точку из матрицы переноса
- AnglesToAngles — перевести 3 угла входного соглашения в 3 угла выходного соглашения
- AnglesToQuaternion — перевести 3 угла входного соглашения в кватернион

Matrix

Трёхмерная матрица

Параметры

- Vx: record
  - cX, cY, cZ: real
- A: real
- Vy: record
  - cX, cY, cZ: real
- B: real
- Vz: record
  - cX, cY, cZ: real
- C: real
- Vt: record
  - cX, cY, cZ: real
- D: real

Методы

- CreateIdentity – создать единичную матрицу
- CreateBy4points – создать матрицу по четырём 3D-точкам
- MakeRotMatrix – создать матрицу вращения по углу, типу угла (1 для градусов или 0 для радиан) и номеру оси (1 для «X», 2 для «Y», 3 для «Z»)
- MakeShiftMatrix – создать матрицу переноса по 3D-точке
- CreateAsAxisAngle – создаёт матрицу по оси и углу
- CreateByPointAndNormal – создаёт матрицу по точке и нормали
- CreateByVectorAndAngle – создаёт матрицу по вектору и углу
- Add – добавить другую матрицу к исходной
- Subtract – вычесть другую матрицу из исходной
- Matr_X_Matr – умножить исходную матрицу на другую
- GetDeterminant – получить определитель матрицы
- GetCoord – получить координату по строке и столбцу матрицы
- Inverse – обратить матрицу
- Transpose – транспонировать матрицу
- CreateBy3Points – создать базис по трём 3D-точкам
- GetLocalPoint – представить глобальную 3D-точку в локальной системе координат
- TransformPoint – представить локальную 3D-точку в глобальной системе координат
- GetLocalVector – представить глобальный 3D-вектор в локальной системе координат
- TransformVector – представить локальный 3D-вектор в глобальной системе координат
- GetLocalMatrix – представить глобальную 3D-матрицу в локальной системе координат

- TransformMatrix – представить локальную 3D-матрицу в глобальной системе координат
- ToString – создать строку с числами из матрицы
- Parse – создать матрицу 4x4 из строки

NCChannel

Объект Channel решает задачу одновременного управления несколькими каналами. Каждый канал управляет собственным изолированным списком регистров. Процедура Activate позволяет переключить текущий активный канал. Процедуры OutBlock, FormBlock и Output позволяют выполнять соответствующие функции для канала, не переключая его в активный режим.

Параметры

- FileName: string
- ChName: string – имя канала.

Методы

- Init – инициализация объекта.
- Activate – переключение канала в текущее состояние.
- OutBlock – этот оператор формирует содержимое кадра в соответствии с заданным форматом переменных, порядком и форматом регистров и выводит кадр в управляющую программу.
- FormBlock – этот оператор формирует кадр управляющей программы в соответствии с заданным форматом, порядком и форматом регистров. Кадр формируется в переменной OutStr$ без вывода в файл управляющей программы.
- Output – вывод строки
- IsActivCh – является ли канал активным: 1 – да, 0 – нет.
- CloseFile – фиксация состояния текущего файла.

Методы доступа к регистрам:

- PutReg – занести значение в регистр.
- GetReg – получить значение из регистра.

Point3D

Параметры

- cX, cY, cZ: real – координаты точки.

Методы

- Init – создать 3D-точку по 3 координатам
- Init_arr – создать 3D-точку по массиву из 3 координат
- Scale – масштабировать координаты 3D-точки на постоянное число
- Inverse – получить 3D-точку с координатами, противоположными по знаку исходным
- Length – получить расстояние от начала системы координат до 3D-точки
- DistanceTo – получить расстояние от данной 3D-точки до другой 3D-точки
- Norm – получить нормализованную 3D-точку
- Add – добавить другую 3D-точку к исходной
- Subtract – вычесть другую 3D-точку из исходной
- VmV – получить скалярное произведение исходного вектора и другого
- VxV – получить векторное произведение исходного вектора и другого

Quaternion4d

Quaternion – группа вращений вокруг неподвижной точки в четырёхмерном евклидовом пространстве.

Параметры

- cX, cY, cZ, cW: real – координаты кватерниона

Методы

- Init – создать объект кватернион
- q_mul_q – получить скалярное произведение кватернионов
- q_x_q – получить векторное произведение кватернионов
- s_x_q – масштабировать координаты кватерниона
- q_add_q – сложить кватернион
- q_sub_q – вычесть кватернион

StringHelper

Методы

- Split – разбиение строки на массив подстрок
- Join – объединение массива строк через разделитель в одну строку
- Insert – вставка одной строки в другую
- ReplaceStr – замена символа или подстроки в строке
- Substr – взятие подстроки по индексу начального элемента и длине
- CurrDate – возврат текущей даты в стандартном формате
- CurrTime – возврат текущего времени в стандартном формате
- FormatDateTime – возврат текущего времени или даты в пользовательском формате
- FormatReg – возвращает строку и число, отформатированные в соответствии с шаблоном регистра

Операторы

На этой странице нет содержимого. См. вложенные разделы.

Оператор начала программы обработки технологической команды PROGRAM

Каждая программа обработки технологических команд начинается с заголовка, состоящего из ключевого слова PROGRAM и имени программы, и заканчивается ключевым словом END.

Формат

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

Описание

Текст программы пишется на специальном проблемно-ориентированном языке и может содержать математические выражения, функции, операторы ввода/вывода, условные операторы, циклы, оператор перехода, вызовы подпрограмм, операторы формирования кадров управляющей программы и операторы работы с файлом технологических команд.

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
```

Оператор присваивания

Этот оператор используется для присваивания переменной значения выражения. В результате выполнения этого оператора переменные могут изменять свои значения.

Формат

```
<Numerical variable> = <Math expression>
```

или

```
<String variable> = <String expression>
```

Описание

Ключевым словом этого оператора является имя переменной. За именем следует символ =, а затем выражение, значение которого будет присвоено переменной. Этот оператор указывает, что результат интерпретации выражения должен быть сохранён в ячейке памяти, идентифицируемой указанной переменной. Тип переменной должен совпадать с типом, возвращаемым выражением, то есть значение математического выражения должно присваиваться числовой переменной, а значение строкового выражения – строковой переменной.

Примечание. При выполнении оператора сначала вычисляется значение правого выражения, а затем полученное значение присваивается переменной в левой части оператора. Это правило позволяет использовать одну и ту же переменную в левой и правой частях оператора. В этом случае в вычислении правого выражения участвует старое значение переменной, и лишь после этого значение правого выражения присваивается той же переменной.

Примеры

```
! Использование оператора присваивания
Zet = 41
Ddel = Mn / Cos(Beta) * Zet
Number = Number + 5
! Формирование строки о станке с ЧПУ
System$ = " 2C42 "
! Присваивание имени системы ЧПУ переменной System$
NAME$ = " TEST PROGRAM " + System$
```

Оператор вывода PRINT

Этот оператор предназначен для вывода результатов на экран (в окно отладочной информации) в процессе тестовой генерации управляющей программы. Если управляющую программу формирует исполняющая система постпроцессора (`Inp.exe`), она игнорирует этот оператор.

Формат

```
PRINT <Math expression> | <String expression>
{, <Math expression> | <String expression>}
{ > | >> output file name}
```

Описание

Ключевым словом этого оператора является слово PRINT. За ключевым словом следует список, содержащий одно или более математических или строковых выражений. Число выражений в этом операторе не ограничено; если их больше одного, они должны разделяться запятыми.

Значения выражений последовательно вычисляются и выводятся на вкладке Отладочная информация.

Чтобы сохранить выводимую информацию на диске, необходимо переназначить вывод в файл. Для этого достаточно указать после последнего из выводимых выражений символ > или >> и имя файла вывода.

Использование символов > и >> указывает режим вывода информации в файл:

- символ > – начать вывод с первой записи, предварительно очистив файл вывода. Если файл с таким именем уже существует, старое содержимое файла теряется;
- символ >> – добавлять выводимые параметры в конец файла. Если файла с таким именем нет, он будет создан и содержимое текущего оператора вывода станет первой записью в новом файле.

Примеры

```
PRINT "Runs the subprogram ABSMOVE"
PRINT CLD[1]-XT, " increase by X"
PRINT AA, " ", CLD[1]
```

```
PRINT CLD[1] > "c:\my_folder\log.txt" ! Вывод в файл
```

Сопутствующие ссылки

Оператор ввода INPUT

Оператор ввода INPUT

Этот оператор предназначен для ввода пользовательских данных с клавиатуры или из файла в процессе выполнения программного модуля постпроцессора.

Ввод с клавиатуры

Ввод с внешнего файла

Сопутствующие ссылки

Оператор вывода PRINT

Ввод с внешнего файла

Синтаксис

```
INPUT <variable> | <string variable> < | << "<file name>"
```

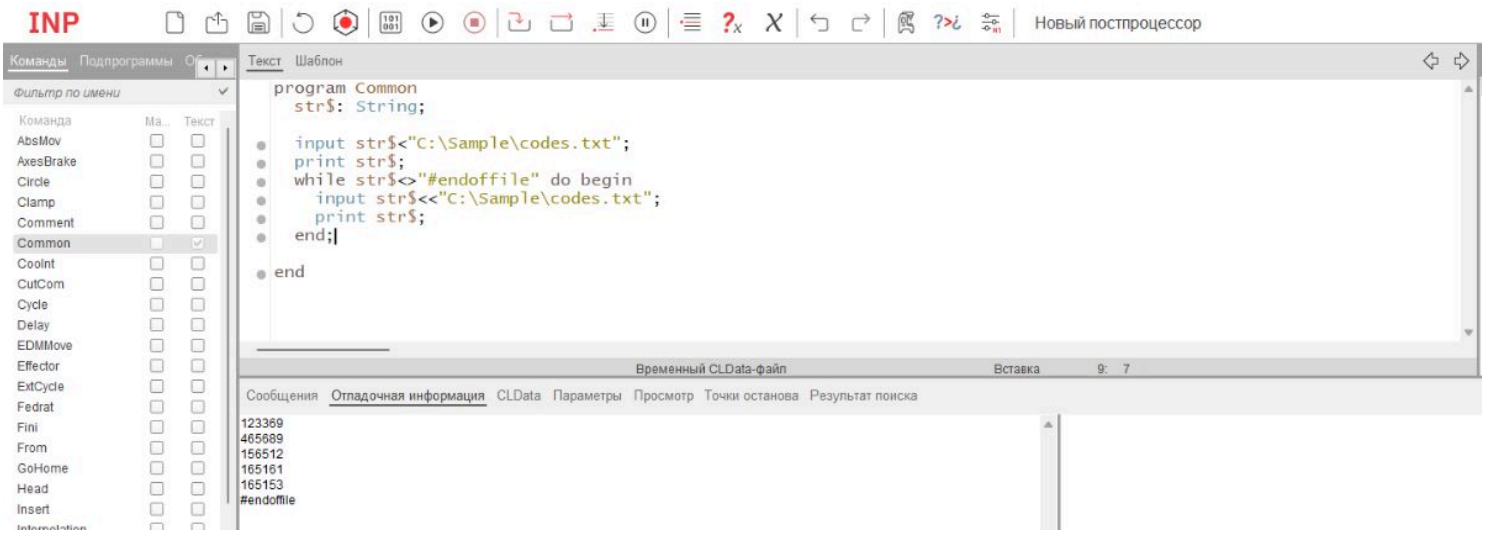
Описание

Для ввода данных с диска переназначьте ввод, добавив после параметров символы < или << и имя файла. В этом случае при выполнении оператора сообщение в кавычках (литерная строка) выводится на экран, а ввод данных производится из указанного файла.

- символ "<" используется для начала ввода значений в переменные текущего оператора ввода с первой строки файла.
- символ "<<" продолжает ввод со строки, следующей за позицией окончания предыдущей операции ввода из этого файла.

При достижении последней строки будет получено значение "#endoffile".

Пример



Ввод с клавиатуры

Эта функциональность предназначена для ввода пользовательских данных с клавиатуры в процессе выполнения программного модуля постпроцессора.

Синтаксис

```
INPUT <caption>, <variable> | <string variable>
{, <caption>, <variable> | <string variable> }
{,<caption>, checkbox, <variable>}
{,<caption>,array, <variable>}
caption = "literal string"
```

Описание

Этот оператор использует ключевое слово INPUT, за которым следует список, состоящий из одной или более числовых или строковых переменных и литерных строк. Число параметров в списке не ограничено; если их больше одного, они разделяются запятыми.

При выполнении оператора создаётся окно ввода, состоящее из литерных строк и полей ввода. Если переменным перед вызовом оператора были присвоены какие-либо значения, то эти значения будут занесены в соответствующие поля окна в качестве значений по умолчанию. При вводе система выполняет контроль типа переменных. В числовую переменную могут вводиться только числовые значения, в строковую – любые символы.

После ввода всех значений необходимо закрыть окно нажатием кнопки OK. При этом переменным будут присвоены соответствующие им введённые значения.

Чтобы задать заголовок диалогового окна, передайте в качестве caption значение "title##caption". Обратите внимание, что каждый следующий параметр "title##caption" переопределяет заголовок окна.

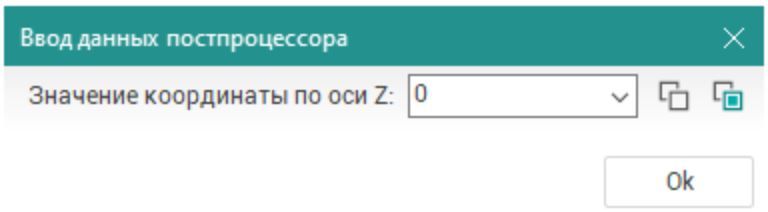
Чтобы добавить изображение в окно ввода, используйте строку "pic=filename". Окно ввода будет расширено, чтобы вместить изображение.

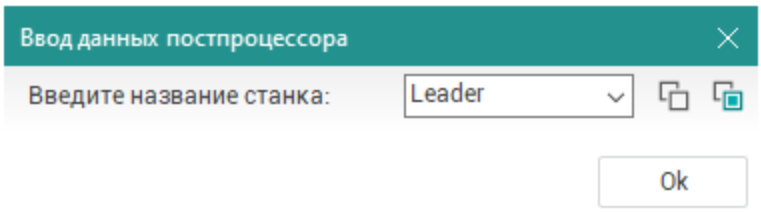
Ключевое слово checkbox создаёт элемент-флажок. Изначально флажок отмечен, если переменная содержит ненулевое значение. Возвращаемое значение переменной равно нулю для снятого состояния и не равно нулю для отмеченного.

Для ввода данных из массива необходимо добавить ключевое слово array. Тип массива может быть string, real или integer.

Этот оператор работает аналогично оператору присваивания, с тем ключевым отличием, что значения переменных присваиваются не из программы, а запрашиваются для ввода при каждом выполнении. Использование этого оператора позволяет один раз написать программу постпроцессора и запускать её многократно, вводя каждый раз разные исходные данные и получая новые результаты в зависимости от вводимых значений.

У каждого поля ввода есть две кнопки для работы с буфером обмена Windows:

- 

– копирует текст поля ввода в буфер обмена (сочетание клавиш: [Ctrl+C]);
- 

– вставляет текст из буфера обмена в поле ввода (сочетание клавиш: [Ctrl+V]).

Примеры

```
INPUT "Sample data input: ", Zt
MachineName$ = "Leader"
INPUT "Input the NC-machine name: ", MachineName$
wx = 10; wy = 10; wz = 5
INPUT "Workpiece dimensions",
"Along X axis", wx,
"Along Y axis", wy,
"Height by Z", wz
! Настройка заголовка и названия окна через переменные. Обратите внимание на приём
'""+', заставляющий ПП использовать переменные caption и title:
rr = 2
title$ = "title"
caption$ = "caption"
input "" + title$ + "##" + caption$, rr
! Выбор переменных из списка:
arr: array of real
arr[1] = 1.1
arr[2] = 2.2
input "caption", arr, rr
! Пример ввода флажка. Обратите внимание на приём '""+', заставляющий ПП
использовать переменную caption
title$ = "checkbox example"
```

```
caption1$ = "initialy checked"
caption2$ = "initialy not checked"
init_checked = 1
init_unchecked = 0
input "" + title$ + "##" + caption1$, checkbox, init_checked, "" + caption2$,
checkbox, init_unchecked
```



Сопутствующие ссылки

Оператор вывода PRINT

Оператор условного выполнения IF

Этот оператор предназначен для выполнения одного из входящих в него операторов в зависимости от некоторого условия.

Формат

```
IF <Conditional expression> then <Statement 1>
{else <Statement 2>}
```

Описание

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

Ключевым словом оператора является слово IF. Вслед за ним следует условное выражение; если оно истинно, выполняется оператор, следующий за ключевым словом THEN. Если условное выражение ложно, то выполняется оператор, следующий за ключевым словом ELSE. ELSE – необязательная часть этого оператора. Если она отсутствует и значение условного выражения ложно, то выполнение оператора IF прекращается и выполняется следующий оператор программы.

Условное выражение может быть одним простым условным выражением или последовательностью простых условных выражений, связанных логическими операциями OR и AND, то есть:

```
<Simple conditional expression> {{AND <Simple conditional expression>} |
{OR <Simple conditional expression>}}
```

Условное выражение истинно, если истинны все простые условные выражения, связанные операцией AND, или если истинно хотя бы одно из простых условных выражений, связанных операцией OR.

Простые условные выражения – это два математических или строковых выражения, связанных операциями сравнения, то есть:

```
<Math expression> <Comparison operator> <Math expression>
```

или

```
<String expression> <Comparison operation> <String expression>
```

Выражения по обе стороны от операции сравнения должны быть одного типа; в противном случае при трансляции вы получите сообщение об ошибке "Incompatible types".

Операция сравнения – одна из следующих операций:

- = – значения левого и правого выражений РАВНЫ;
- > – значение левого выражения БОЛЬШЕ значения правого выражения;
- < – значение левого выражения МЕНЬШЕ значения правого выражения;
- # или <> – значения выражений РАЗЛИЧНЫ;
- >= или => – значение левого выражения РАВНО ИЛИ БОЛЬШЕ значения правого выражения;
- <= или =< – значение левого выражения РАВНО ИЛИ МЕНЬШЕ значения правого выражения.

Следует обратить особое внимание на сравнение строковых выражений. Сравнение двух строк выполняется посимвольно слева направо с учётом их ASCII-кодов. Учтите, что прописные буквы меньше строчных.

Если строки имеют разную длину, но более длинная строка целиком включает более короткую, то более длинная строка больше. Строки равны, если они содержат одни и те же символы и имеют одинаковую длину.

При выполнении условного оператора сначала вычисляются значения математических и строковых выражений в простых условных выражениях, затем выполняются операции сравнения, а затем без приоритета выполняются операции AND и OR. После этого в зависимости от результата всего условного выражения выполняется оператор THEN или ELSE.

Примеры

```
! Пример 1.
! Пример использования условного оператора
IF kadr <224 THEN PRINT "This is a short program "
ELSE PRINT "You are the monster in programming"
! Пример 2.
! Этот пример показывает использование вложенных условных операторов
IF POS(" ",str$) = 2 THEN
b$ = "The first char in string is a letter"
ELSE
IF POS(" ",str$) <= 4 THEN
b$ = "First in string is a word, it is length is 4 symbols"
```

```
  ELSE
  b$ = "First in string is a word greater than 5 symbols length"
```

Сопутствующие ссылки

Оператор множественного условного выполнения CASE

Оператор множественного условного выполнения CASE

Оператор Case позволяет проверить значение выражения и в зависимости от результата выполнить разный код.

Формат

```
CASE <expression> OF
  <values list 1> : <statement 1>
  ...
  <values list N>: <statement N>
  ELSE <statement M>
END
```

Описание

В этой конструкции выражение должно быть числовым. Недопустимо использовать строковые выражения.

Списки значений могут содержать одну или несколько разделённых запятыми констант. За списком значений следует символ `:`. После него записывается оператор, который выполняется, если выражение равно одному из значений списка. После выполнения этого оператора выполнение структуры CASE завершается и выполняется следующий оператор.

Если результат выражения не определён ни в одном из списков, то выполняется оператор, следующий за ключевым словом ELSE. Часть ELSE является необязательной. Если ELSE отсутствует и в списках нет соответствующего значения, то не выполняется ничего.

Пример

```
CASE i OF
  1,2,3,4,5: Str = "Less or equal 5"
  6,7,8,9: Str = "Greater 5"
  ELSE Str = "Error value"
END
```

Сопутствующие ссылки

Оператор условного выполнения IF

Оператор перехода на метку JUMP

Оператор используется для нарушения порядка выполнения операторов.

Формат

```
JUMP <Label number>
...
<Label number>:
```

Описание

За ключевым словом JUMP должна быть указана одна из меток, определённых в текущей программе. В отличие от объявления метки, символ `:` после номера метки не указывается.

После оператора JUMP будет выполнен оператор, помеченный соответствующей меткой. Соответствующую метку необходимо определить, иначе при трансляции появится сообщение об ошибке.

Использование оператора JUMP в программе, как правило, не рекомендуется.

Пример

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

Оператор цикла FOR

Этот оператор повторяет один или несколько операторов заданное число раз.

Синтаксис

```
FOR <numerical cycle control variable> = <math expression> TO <math expression>
{STEP <number> | <number variable>} DO <operator>
```

Описание

Этот оператор обеспечивает повторяющееся выполнение некоторого оператора с одновременным увеличением переменной цикла до тех пор, пока её значение не превысит заданный предел.

Ключевым словом оператора является слово FOR. Вслед за ним следует конструкция, аналогичная присваиванию:

```
<Numerical cycle control variable> = <Math expression>
```

По сути они очень похожи, поскольку эта конструкция присваивает числовой переменной цикла начальное значение, определяемое математическим выражением.

За зарезервированным словом TO следует математическое выражение, определяющее верхнюю границу. Когда числовая переменная цикла достигает этой границы, выполнение оператора цикла прекращается и выполняется следующий оператор программы. В качестве верхней границы цикла может быть указано любое математическое выражение.

Необязательная часть этого оператора, содержащая зарезервированное слово STEP и следующее за ним число или числовую переменную, определяет шаг увеличения переменной цикла. По умолчанию шаг равен 1. Шагом не может быть выражение.

Оператор, который должен выполняться циклически, указывается после ключевого слова DO. Все остальные параметры цикла служат для организации циклического выполнения этого оператора.

Если тело цикла содержит несколько операторов, они должны быть заключены между ключевыми словами BEGIN и END.

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
```

Сопутствующие ссылки

Оператор цикла REPEAT

Оператор цикла WHILE

Оператор BREAK

Оператор CONTINUE

Оператор CONTINUE

Синтаксис

```
CONTINUE
```

Описание

Оператор CONTINUE прерывает выполнение текущей итерации цикла. В случае цикла с предусловием (FOR или WHILE) управление передаётся в начало цикла, где проверяется условие (для цикла FOR перед этим вычисляется новое значение переменной цикла). Если условие истинно, выполнение цикла продолжается. В случае цикла с постусловием (REPEAT) управление передаётся в конец цикла, где проверяется условие. Если условие ложно, выполнение цикла продолжается.

Использование оператора CONTINUE допустимо только внутри тела цикла; во всех остальных случаях выдаётся сообщение об ошибке.

Сопутствующие ссылки

Оператор цикла FOR

Оператор цикла REPEAT

Оператор цикла WHILE

Оператор BREAK

Оператор BREAK

Синтаксис

```
BREAK
```

Описание

Оператор BREAK предназначен для прерывания выполнения цикла. Управление передаётся оператору, следующему за прерванным циклом. В случае вложенных циклов прекращается выполнение того цикла, в теле которого находится оператор BREAK.

Использование оператора BREAK допустимо только внутри тела цикла; во всех остальных случаях выдаётся сообщение об ошибке.

Сопутствующие ссылки

Оператор цикла FOR

Оператор цикла REPEAT

Оператор цикла WHILE

Оператор CONTINUE

Оператор цикла REPEAT

Структура `REPEAT...UNTIL` используется для повторяющегося выполнения совокупности операторов, называемой телом цикла, до тех пор, пока не станет истинным некоторое условие.

Синтаксис

```
REPEAT
  <operators of the cycle body>
UNTIL <conditional expression>
```

Описание

Структура работает следующим образом. Выполняются операторы тела цикла, после чего вычисляется условное выражение. Если условное выражение возвращает ложь, то выполнение операторов тела цикла повторяется, и после этого условное выражение вычисляется снова. Повторение продолжается до тех пор, пока условное выражение не вернёт истину. После этого выполнение цикла завершается и выполняется следующий за структурой `REPEAT...UNTIL` оператор.

Поскольку проверка условия выполняется после выполнения тела цикла, оно выполняется хотя бы один раз, даже если условие истинно заранее. С другой стороны, условие должно однажды вернуть истину, иначе цикл будет выполняться бесконечно.

Сопутствующие ссылки

Оператор цикла FOR

Оператор цикла WHILE

Оператор BREAK

Оператор CONTINUE

Оператор цикла WHILE

Структура `WHILE...DO` используется для повторяющегося выполнения совокупности операторов, называемой телом цикла, пока истинно некоторое условие.

Формат

```
WHILE <conditional expression> DO <statement>
```

Структура работает следующим образом. Сначала вычисляется условное выражение. Если результат истинен, то выполняется тело цикла, после чего условное выражение вычисляется снова. Повторение продолжается до тех пор, пока выражение не вернёт ложь. После этого выполнение цикла завершается и выполняется следующий за структурой `WHILE...DO` оператор.

Поскольку проверка условия выполняется перед выполнением тела цикла, то, если условное выражение ложно заранее, тело цикла не выполняется ни разу. В этом основное отличие структуры `WHILE...DO` от структуры REPEAT...UNTIL. С другой стороны, условие должно однажды вернуть истину, иначе цикл будет выполняться бесконечно.

Сопутствующие ссылки

Оператор цикла FOR

Оператор цикла REPEAT

Оператор BREAK

Оператор CONTINUE

Составной оператор BEGIN...END

Используется для объединения группы операторов в один оператор. Является аналогом операторных скобок.

Формат

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

Описание

Составной оператор обеспечивает выполнение входящих в него операторов по порядку. Служебные слова BEGIN и END являются аналогом операторных скобок. Глубина вложенности составных операторов не ограничена. Составной оператор может использоваться везде, где допустимо использование одиночного оператора.

Пример

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

Оператор вызова подпрограммы CALL

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

Формат

```
CALL <program name> {(<the list of the parameters>)}
```

Описание

Ключевым словом этого оператора является слово CALL. После этого слова необходимо задать имя программы – строковое выражение или литерную строку без двойных кавычек. Значение выражения является именем подпрограммы.

Список параметров – список фактических параметров через запятую: переменные, литерные константы, выражения и массивы. Типы фактических параметров должны соответствовать типам формальных параметров, указанных в объявлении подпрограммы. Если параметры в процедуру не передаются, то круглых скобок быть не должно.

Оператор CALL передаёт управление указанной подпрограмме, а после её выполнения возвращает управление в программу, из которой был произведён вызов.

Пример

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

Сопутствующие ссылки

Оператор начала подпрограммы SUB

Оператор окончания подпрограммы SUBEND

Оператор начала подпрограммы SUB

Используется для объявления подпрограммы и списка параметров, передаваемых программе при вызове.

Формат

```
SUB <subprogram name> {(<the list of the parameters>)}
```

Описание

Ключевым словом этого оператора является SUB. За ним следует имя подпрограммы – литерная строка без двойных кавычек, а затем необязательный список параметров в круглых скобках.

Список параметров – это последовательность числовых и строковых переменных или массивов. Если число параметров больше одного, они разделяются запятыми.

Переменные, определённые в списке, при вызове будут содержать значения, указанные в операторе вызова. Поэтому эти переменные объявлены в подпрограмме и могут использоваться везде.

Пример

```
sub GetProgramID(PrgID: Integer)
  ! Извлечение числового идентификатора из строкового имени NC-программы
  i: Integer
  j: Integer
  k: Integer
  s$: String
  i = 1
  j = 0
  k = 0
  while i<=Len(NCName$) do begin
    s$ = Copy(NCName$, i, 1)
    case Ord(s$) of
      48, 49, 50, 51, 52, 53, 54, 55, 56, 57: begin
        if j<1 then j = i
        if (k<1) or (k=(i-1)) then
          k = i
      end
    end
    i = i + 1
  end
```

```
  if (j>0) and (k>0) then begin
    s$ = Copy(NCName$, j, k-j+1)
    PrgID = Num(s$)
  end
subend
```

Сопутствующие ссылки

Оператор вызова подпрограммы CALL

Оператор окончания подпрограммы SUBEND

Оператор окончания подпрограммы SUBEND

Оператор используется для указания окончания подпрограммы.

Формат

```
SUBEND
```

Описание

Оператор объявляет конец подпрограммы. При выполнении SUBEND выполнение подпрограммы завершается, значения всех переменных, указанных в списке параметров, присваиваются переменным, указанным в операторе CALL, и продолжается выполнение вызвавшей подпрограммы.

Сопутствующие ссылки

Оператор вызова подпрограммы CALL

Оператор начала подпрограммы SUB

Оператор начала процедуры PROC

Оператор предназначен для объявления начала и списка параметров подпрограммы в процедуре обработки команды CLDATA.

Формат

```
PROC <procedure name> {(<list of formal parameters>)}
```

Описание

За ключевым словом PROC объявляется имя процедуры. Имя процедуры – идентификатор, не совпадающий с ключевыми словами генератора постпроцессоров и именами переменных. Это имя служит для идентификации процедуры при вызове, поэтому имена всех процедур должны быть уникальны. За именем процедуры следует необязательный список формальных параметров. Это последовательность числовых и строковых переменных и массивов. Если число параметров больше одного, они должны разделяться запятыми.

В отличие от оператора SUB, переменные, определённые в списке формальных параметров, не инициализируются, то есть перед включением в список этим переменным необходимо присвоить какие-либо значения.

Переменные, объявленные в списке формальных параметров, будут содержать значения, заданные в операторе CALL. Поэтому эти переменные определены в программе и могут использоваться везде.

Оператор PROC похож на оператор SUB. Главное отличие в том, что процедура имеет доступ ко всем переменным и массивам той программы или подпрограммы, в теле которой она расположена.

Объявленную процедуру можно располагать в любом месте программы.

Сопутствующие ссылки

Оператор возврата из процедуры RETURN

Оператор возврата из процедуры RETURN

Используется для указания окончания процедуры.

Формат

```
RETURN
```

Описание

Этот оператор является последним оператором процедуры. При выполнении оператора RETURN производится обратное присваивание значений формальных параметров переменным, указанным в операторе вызова, и начинается выполнение следующего за оператором вызова оператора.

Пример

```
program TestProc
  NErr: Integer
  S: String
  proc PrintDebugInfo(NErr, S)
    case NErr of
      1: Print "Interpolation error: ", S
      2: Print "Approximation error: ", S
      else Print "Unknown error: ", S
    end
  return
  PrintDebugInfo(CLD[1], "parameter 1")
  PrintDebugInfo(CLD[2], "parameter 2")
  PrintDebugInfo(CLD[3], "parameter 3")
end
```

Сопутствующие ссылки

Оператор начала процедуры PROC

Оператор вывода кадра OUTBLOCK

Этот оператор формирует содержимое кадра в соответствии с заданным форматом переменных, порядком и форматом регистров и выводит кадр в управляющую программу.

Формат

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

Описание

Кадр управляющей программы формируется по следующему алгоритму:

Регистры перебираются по порядку, начиная с первого. Если предыдущее и текущее значения регистра различаются, то регистр выводится в кадр, а его текущее значение приравнивается предыдущему; в противном случае регистр в кадр не выводится.

При выводе регистра в кадр сначала записывается его идентификатор, а затем его значение, умноженное на масштаб. Число выводится в кадр в соответствии с заданным форматом. После этого вызывается программа обработки, если она определена. Кадр формируется в переменной OutStr$ и по окончании формирования выводится в программу отдельной строкой.

Используйте необязательный параметр `FileName$`, чтобы указать файл, в который система выводит кадр УП. Если параметр опущен, то кадр выводится в текущий файл. Если в качестве имени файла указано "*", то кадр будет выведен в главный файл, при этом текущий файл также будет изменён. Если `FileName$` задаёт имя файла, то будет создана соответствующая закладка, а текущий файл вывода будет соответственно изменён.

Сопутствующие ссылки

Предопределённые функции

Оператор формирования кадра FORMBLOCK

Оператор прямого вывода в кадр OUTPUT

Оператор формирования кадра FORMBLOCK

Этот оператор формирует кадр управляющей программы в соответствии с заданным форматом, определённым порядком и форматом регистров. Кадр формируется в переменной OutStr$ без вывода в файл управляющей программы.

Формат

```
FORMBLOCK
```

Описание

Процесс формирования аналогичен оператору OUTBLOCK, но переменная OutStr$ в кадр управляющей программы не выводится. Далее с этой переменной можно работать как со строковой переменной и вывести её в файл управляющей программы оператором OUTPUT.

Сопутствующие ссылки

Предопределённые функции

Оператор вывода кадра OUTBLOCK

Оператор прямого вывода в кадр OUTPUT

Оператор прямого вывода в кадр OUTPUT

Этот оператор выводит содержимое строковой переменной отдельным кадром управляющей программы.

Формат

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

Описание

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

Используйте необязательный параметр FileName$, чтобы указать файл, в который система выводит кадр УП. Если параметр опущен, то кадр выводится в текущий файл. Если в качестве имени файла указано "*", то кадр будет выведен в главный файл, при этом текущий файл также будет изменён. Если FileName$ задаёт имя файла, то будет создана соответствующая закладка, а текущий файл вывода будет соответственно изменён.

Примеры

```
! Вывод первого символа в NC-код
OutPut "%"
Name$ = " John "
F$ = "Lord"
! Вывод строки в управляющую программу
OutPut "Programmer "+Name$+" "+F$
! Строка "Programmer John Lord" будет записана в управляющую программу
! Пример использования FormBlock и Output вместо OutBlock
FormBlock
Output OutStr$
! Действие этих двух операторов аналогично OutBlock
```

Сопутствующие ссылки

Оператор вывода кадра OUTBLOCK

Оператор формирования кадра FORMBLOCK

Оператор замены подстроки в строке REPLACE

Оператор ищет подстроку в строке и заменяет её на требуемую.

Формат

```
REPLACE (<string variable>, <looked string>, <string for replace>)
```

Описание

Этот оператор ищет искомую подстроку, начиная с начала строковой переменной. Если искомая подстрока найдена, то она заменяется на строку для замены и поиск прекращается. Если искомая подстрока не найдена, замена не производится.

Пример

```
! Заполнение исходной переменной:
S$ = "GXX"
! Поиск и замена "XX" на "21":
REPLACE(S$, "XX", "21")
! Переменная S$ имеет значение "G21".
```

Оператор формирования кадра по шаблону MASK

Оператор Mask формирует кадр NC-кода по строке шаблона в соответствии с правилами и выводит строку в управляющую программу.

Формат

```
MASK (<mask string>) {, OutBlock | FormBlock {, Poll}}
```

Описание

Оператор Mask формирует строку NC-кода в соответствии с правилами, описанными в строке шаблона.

Опции OutBlock и FormBlock являются взаимоисключающими.

Опция Poll соответствует переключателю Опросить все регистры. Подробнее см. раздел Переключатели шаблона.

- OutBlock соответствует переключателю Вывод текущего кадра;
- FormBlock включает режим формирования переменной OutStr$, аналогичный FORMBLOCK, без вывода в текст УП.

Пример

```
! Формирование строк NC-кода 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
! Формирование строки NC в переменной OutStr$
MASK(G_INTERP[INTERP]X[CLD.X]Y[CLD.Y]Z[CLD.Z]), FormBlock, Poll
! В OutStr$ содержится сформированная строка NC
Output OutStr$
```

Оператор смены текущего файла УП

ChangeNCFile меняет текущий файл вывода УП. Последующие операторы вывода NC-кода, в которых файл УП не указан, будут записывать кадры в текущий файл УП.

Синтаксис

```
ChangeNCFile NewNCFileName$
```

Описание

`NewNCFileName$` – выражение, результатом вычисления которого является строка.

Пример

```
! Пример изменения текущего файла УП
ChangeNCFile "File1.nc" ! смена текущего файла УП
Output "NC block" ! эта строка будет выведена в файл "File1.nc"
```

Сопутствующие ссылки

Оператор вывода кадра OUTBLOCK

Оператор прямого вывода в кадр OUTPUT

Оператор вызова внешней задачи

Используйте оператор ShellExec для выполнения команды среды или вызова внешней задачи.

Синтаксис

```
ShellExec CommandLine$
```

Описание

`CommandLine$` – строковое выражение.

При выполнении этого оператора постпроцессор запустит интерпретатор команд операционной системы (CMD.EXE) и передаст ему на исполнение команду `CommandLine$` (по сути этот оператор выполняет "cmd /c CommandLine$"). Обратите внимание, что окно консоли не отображается.

Пример

```
! Выполнение команды ОС
ShellExec "erase temp\*.*" ! Очистить папку temp
```

CustomData - оператор обращения к пользовательским данным CAM-системы

Используйте оператор CustomData для обращения к пользовательским данным проекта CAM-системы.

Синтаксис

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

Описание

`ItemName$` – выражение, результатом вычисления которого является строка.

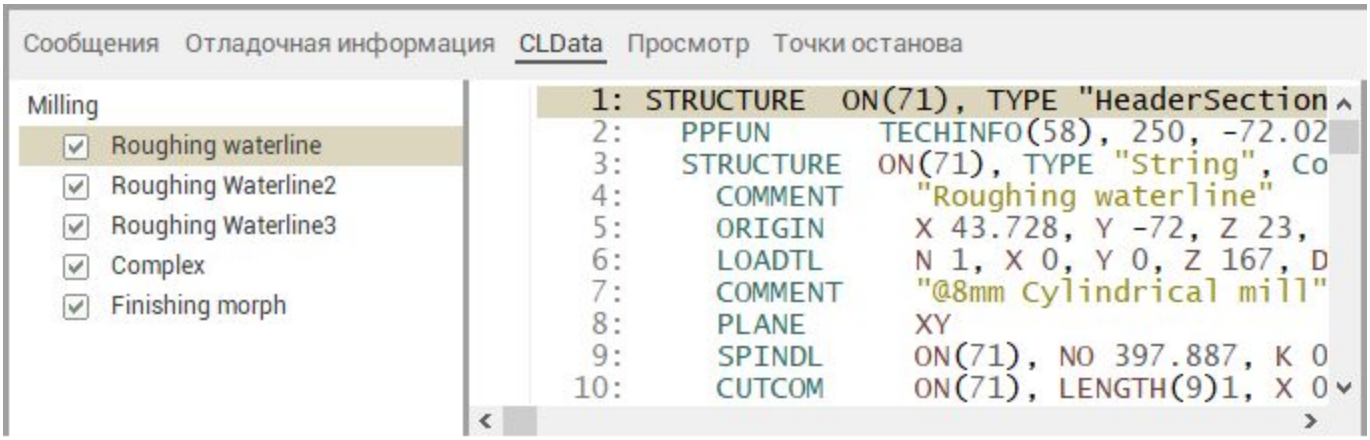
Оператор CustomData возвращает значение пользовательского элемента данных с именем ItemName$, установленное пользователем в проекте CAM-системы.

Пример

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

Операторы работы с NC-подпрограммами

NC-подпрограмма представляет собой последовательность технологических команд, оформленную в виде отдельного файла технологических команд *.mcd и обрамлённую 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.

В отличие от обычных файлов *.mcd, технологические команды в которых начинают обрабатываться автоматически, если соответствующий файл отмечен галочкой рядом с именем, технологические команды в NC-подпрограммах обрабатываются 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 и доступен в процедурах обработки через предопределённый массив CLD. Таким образом, в процедуре обработки выражение `CLD[2]` вернёт номер соответствующей NC-подпрограммы. Также NC-подпрограмма может характеризоваться строковым именем. Начальное значение строкового имени задаётся в технологической команде COMMENT, расположенной перед технологической командой PPFUN STARTSUB(50). В процедурах обработки технологических команд строковое имя NC-подпрограммы может быть прочитано, а также изменено оператором NCSUB.NAME(n), где n – уникальный номер NC-подпрограммы.

В процессе трансляции технологических команд в управляющую программу может потребоваться идентифицировать текст 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.

Сопутствующие ссылки

Оператор вывода NC-подпрограммы NCSUB.OUTPUT

Оператор NCSUB.OUTPUTALL

Оператор определения имени NC-подпрограммы NCSUB.NAME

Оператор определения начальной метки NC-подпрограммы NCSUB.STARTLABEL

Оператор определения конечной метки NC-подпрограммы NCSUB.ENDLABEL

Методика работы с NC-подпрограммами

Оператор вывода NC-подпрограммы NCSUB. OUTPUT

Оператор передаёт управление в файл технологических команд *.mcd, соответствующий определённой NC-подпрограмме.

Формат

```
NCSUB.OUTPUT {(<NC-subroutine Number> {, <the Mode>}<, File name>)}
```

Оператор является функцией, то есть 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 `NC-subroutine Number`. Номер NC-подпрограммы обычно передаётся через предопределённый 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 Mode 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 Mode считается, что режим равен 1.

В режиме 0 выполняется трансляция технологических команд с изменением значений регистров и глобальных переменных, однако результат трансляции не выводится в текст управляющей программы. Таким образом, осуществляется "холостой прогон", позволяющий имитировать вызов программы в случае, когда текст подпрограммы должен появиться в другом месте управляющей программы.

В режиме 1 (по умолчанию) трансляция технологических команд NC-подпрограммы производится так же, как и в режиме 0, однако результат трансляции выводится в текст управляющей программы, если NC-подпрограмма ещё 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 выводится в управляющую программу только один раз.

При вызове оператора в режиме 2 результат трансляции NC-подпрограммы выводится в управляющую программу всегда, то есть столько раз, сколько вызывается оператор. Такой

режим позволяет "развернуть" NC-подпрограммы в случае, когда система ЧПУ не поддерживает выполнение подпрограмм.

Возвращаемый результат показывает, производился ли вывод NC-подпрограммы в управляющую программу (УП). Результат равен 0, если вывода не было (был осуществлён "холостой прогон"). Результат равен 1, если 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 после первого вызова оператора для каждой NC-подпрограммы.

Используйте необязательный строковый параметр File Name, чтобы указать имя файла, в который будет сохранён NC-код подпрограммы. Если параметр опущен, то подпрограмма выводится в файл основной управляющей программы. Если в параметре File Name не указан путь к файлу, то файл NC-подпрограммы будет создан в той же папке, что и основная программа.

Примеры.

Например, 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 с номером 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
```

Допустим, что 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 этом 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 перейдёт на файл с NC-подпрограммой номер 1, так как 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 вернётся в основной файл, и будут обработаны все оставшиеся команды (строки 11 – 15).

Другие примеры вызова оператора приведены ниже:

```
! Выполняет трансляцию всех имеющихся NC-подпрограмм
NCSub.Output
! Выполняет трансляцию NC-подпрограммы с номером 3 в режиме 0
NCSub.Output(3, 0)
! Выполняет трансляцию NC-подпрограммы с номером, взятым из
! второго параметра текущей технологической команды, в режиме 2
NCSub.Output(CLD[2], 2)
if NCSub.Output(CLD[2]) = 0 then ! Выполняет трансляцию
begin ! NC-подпрограммы в режиме 1.
Output "CALL "+ ! Выводит строку "CALL ", если вывод
NCSub.StartLabel(CLD[2]) ! в УП уже был выполнен
end
! Транслирует NC-подпрограмму с номером 7 в режиме 1
! и выводит результат в файл Sub7.txt
NCSub.Output(7, "Sub7.txt")
```

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

Сопутствующие ссылки

Операторы работы с NC-подпрограммами

Оператор NCSUB.OUTPUTALL

Оператор определения начальной метки NC-подпрограммы NCSUB.STARTLABEL

Оператор NCSUB.OUTPUTALL

Используйте `NCSUB.OUTPUTALL` для вывода всех NC-подпрограмм. Как и оператор NCSUB.OUTPUT при вызове без параметров, оператор `NCSUB.OUTPUTALL` транслирует все файлы CLData NC-подпрограмм, но, в отличие от NCSUB.OUTPUT, `NCSUB.OUTPUTALL` может выводить все NC-подпрограммы в отдельные файлы.

Синтаксис

```
NCSUB.OUTPUTALL(<Mode>)
```

Mode – режим выполнения:

- 0 – работает точно так же, как NCSUB.OUTPUT без параметров;
- 1 – оператор сохранит тексты NC-подпрограмм в файлах с именами NCSUB.NAME для каждой подпрограммы соответственно.

Расширение файлов определяется настройками постпроцессора. Файлы NC-подпрограмм будут сохранены в папку основной управляющей программы.

Сопутствующие ссылки

Операторы работы с NC-подпрограммами

Оператор вывода NC-подпрограммы `NCSUB.OUTPUT`

Оператор определения имени NC-подпрограммы `NCSUB.NAME`

Оператор определения начальной метки NC-подпрограммы NCSUB.STARTLABEL

Оператор возвращает строковое значение начальной метки NC-подпрограммы, а также позволяет задать начальной метке NC-подпрограммы произвольное текстовое значение.

Формат

```
S$ = NCSUB.STARTLABEL(<NC-subroutine Number>)
```

или

```
NCSUB.STARTLABEL(<NC-subroutine Number>) = S$
```

Описание

- `NC-subroutine Number` – 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 обработки строк.

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

Обычно на момент начала трансляции технологических команд в управляющую программу (УП) место расположения конкретной подпрограммы в УП неизвестно. В это время начальная метка NC-подпрограммы установлена в значение по умолчанию SLabelNxxx, где xxx – номер NC-подпрограммы. В процессе реализации NC-подпрограммы, то есть её вывода в УП, обычно становится возможным определить начальную метку. Поэтому в процедуре обработки технологической команды начала подпрограммы PPFUN STARTSUB(50) используйте оператор определения начальной метки для 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 предопределённый массив параметров технологической команды CLD.

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

Перед вызовом подпрограммы обычно требуется определить, где расположена данная подпрограмма. Поэтому в процедуре обработки технологической команды вызова подпрограммы PPFUN CALLSUB(52) используйте оператор определения начальной метки подпрограммы:

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

Затем выведите определённую таким образом метку в текст управляющей программы в необходимом формате.

Примечание. Если выполнение процедуры обработки технологической 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 вернёт 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, выполняемые 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 выполнять не при получении значения метки, а перед присвоением значения метки в процедуре обработки технологической команды начала подпрограммы PPFUN STARTSUB(50).

Сопутствующие ссылки

Операторы работы с NC-подпрограммами

Оператор определения конечной метки NC-подпрограммы NCSUB.ENDLABEL

Оператор определения конечной метки NC-подпрограммы NCSUB.ENDLABEL

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

Формат

```
S$ = NCSUB.ENDLABEL(
<NC-subroutine Number>
)
```

или

```
NCSUB.ENDLABEL(<NC-subroutine Number>) = S$
```

Описание

- `NC-subroutine Number` – уникальный идентификатор NC-подпрограммы.
- `S$` – произвольная строковая переменная. Во втором случае также может задаваться строковая константа или произвольное выражение, возвращающее строку.

Оператор может использоваться как строковая переменная в выражениях и функциях обработки строк.

Оператор используется для определения конечной метки NC-подпрограммы при выводе в управляющую программу текста, идентифицирующего конец подпрограммы. В качестве метки можно указать номер кадра.

Обычно на момент начала трансляции технологических команд в управляющую программу (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, то есть её вывода в УП, обычно становится возможным определить конечную метку. Поэтому в процедуре обработки технологической команды конца подпрограммы PPFUN ENDSUB(51) используйте оператор определения конечной метки для установки конечной метки подпрограммы:

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

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

Примечание. При включённой автоматической нумерации кадров управляющей программы перед началом выполнения процедуры обработки технологической команды конца подпрограммы PPFUN ENDSUB(51) конечной метке автоматически присваивается текущее значение регистра номера кадра. Если это устраивает, можно не использовать оператор для присвоения значения конечной метки.

Перед вызовом подпрограммы обычно определено, где расположена данная подпрограмма. Поэтому в процедуре обработки технологической команды вызова подпрограммы PPFUN CALLSUB(52) используйте оператор определения конечной метки подпрограммы:

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

Затем выведите определённую таким образом метку в текст управляющей программы в необходимом 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 вернёт значение по умолчанию вида ELabelNxxx. 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, выполняемые 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 выполнять не при получении значения метки, а перед присвоением значения метки в процедуре обработки технологической команды конца подпрограммы PPFUN ENDSUB(51).

Сопутствующие ссылки

Операторы работы с NC-подпрограммами

Оператор определения начальной метки NC-подпрограммы `NCSUB.STARTLABEL`

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(<NC-subroutine Number>)
```

или

```
NCSUB.NAME(<NC-subroutine Number>) = S$
```

Описание

- `NC-subroutine number` – 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 обработки строк.

Для вывода в управляющую программу текстового идентификатора подпрограммы в необходимом 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 текста комментария (технологической команды COMMENT), расположенного перед технологической командой начала NC-подпрограммы PPFUN STARTSUB(50) соответствующей NC-подпрограммы.

Пример

```
St$: text
n1: Integer
St$ = "Example"
n1 = 3
! Присвоение имени значения "Example3":
NCSub.Name(n1) = St$ + Str(n1)
```

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

Сопутствующие ссылки

Операторы работы с NC-подпрограммами

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, как можно реализовать каждый из указанных случаев.

Подпрограммы выводятся в УП до вывода основной программы

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

Чтобы выполнить вывод подпрограмм в самом начале, необходимо вставить соответствующий код в процедуру обработки технологической команды PARTNO, которая выполняется первой. Вид процедуры может быть следующим.

```
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 NC-подпрограммы, конец NC-подпрограммы и вызов NC-подпрограммы, необходимо написать процедуру-обработчик технологических команд PPFUN STARTSUB(50), PPFUN ENDSUB(51) и PPFUN CALLSUB(52), как показано ниже.

```
program PPFun
  if CLD[1]=50 then begin ! Команда StartSub
    ! Вывод строки, идентифицирующей начало
```

```
    ! подпрограммы (здесь "NameSubroutine")
    Output "o"+NCSub.Name(CLD[2])
    ! Присвоение начальной метке текущего
    ! значения регистра номера кадра
    NCSub.StartLabel(CLD[2]) = Str(BlockN)
  end
  else if CLD[1]=51 then begin ! Команда EndSub
    ! Вывод строки, идентифицирующей конец подпрограммы
    ! (здесь "Nxx M99", где Nxx - номер кадра)
    FormBlock
    Output OutStr $ + "M99"
    ! Присвоение конечной метке значения
    ! из регистра номера кадра
    NCSub.EndLabel(CLD[2]) = Str(BlockN)
  end
  else if CLD[1]=52 then begin ! Команда CallSub
    ! Вывод строки вызова подпрограммы вида
    ! "Nxx M98 Nbb, Nee", где
    ! Nxx - текущий кадр,
    ! Nbb - кадр начала подпрограммы,
    ! Nee - кадр конца подпрограммы
    FormBlock
    Output OutStr$ + "M98" +
    NCSub.StartLabel(CLD[2]) + "," +
    NCSub.EndLabel(CLD[2])
    ! Холостой прогон подпрограммы
    NCSub.Output(CLD[2], 0)
  end
end
```

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

В операторе вывода NC-подпрограммы NCSUB.OUTPUT указан режим 0, однако, поскольку 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 процедуры вывода технологических команд начала, конца и вызова подпрограмм (PPFUN STARTSUB(50), PPFUN ENDSUB(51) и PPFUN CALLSUB(52)) по сравнению с предыдущим случаем остаётся неизменным. Однако необходимо учитывать, что значения, возвращаемые функциями работы с метками (NCSUB.STARTLABEL и NCSUB.ENDLABEL), до момента окончания трансляции являются не теми метками, которые им присвоены, а имеют значения по умолчанию вида SLabelNxxx и ELabelNxxx, где xxx – номер NC-подпрограммы, так как реальные значения меток на момент запроса не определены. Реальные значения меток подставляются лишь после окончания трансляции управляющей программы в целом.

Подпрограммы выводятся в УП в месте первого вызова

В этом случае подпрограмма одновременно является частью основной программы, обращение к которой может производиться многократно, с указанием, например, начального и конечного кадров. Таким образом, подпрограмма выводится при первом вызове как часть основного кода, а при последующих – как вызов существующего кода. Весь код, отвечающий за NC-подпрограммы, помещается в процедуру обработки технологических команд начала, конца и вызова подпрограмм (PPFUN STARTSUB(50), PPFUN ENDSUB(51) и PPFUN CALLSUB(52)).

```
program PPFun
  if CLD[1]=50 then begin ! Команда StartSub
    ! Присвоение начальной метке текущего значения
    ! регистра номера кадра
    NCSub.StartLabel(CLD[2]) = Str(BlockN)
  end
  else if CLD[1]=51 then begin ! Команда EndSub
    ! Присвоение конечной метке значения
    ! из регистра номера кадра
    NCSub.EndLabel(CLD[2]) = Str(BlockN)
  end
  else if CLD[1]=52 then begin ! Команда CallSub
    ! Если вывод подпрограммы уже был
    if NCSub.Output(CLD[2])=0 then begin
```

```
        ! Вывод строки вызова подпрограммы вида
        ! "Nxx M98 Nbb, Nee", где
        ! Nxx - текущий кадр,
        ! Nbb - кадр начала подпрограммы,
        ! Nee - кадр конца подпрограммы
        FormBlock
        Output OutStr$ + "M98" +
        NCSub.StartLabel(CLD[2]) +
        ","+NCSub.EndLabel(CLD[2])
      end
    end
  end
```

В приведённом примере первый вызов оператора `[NCSUB.OUTPUT(CLD[2])](operator-of-a-output-of-nc-subroutine.md)` для каждой подпрограммы осуществит вывод NC-подпрограммы в управляющую программу и вернёт значение 1. При каждом последующем вызове будет осуществлён "холостой прогон" и возвращено значение 0. После этого 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, однако при включённой автоматической нумерации подобные действия производятся автоматически и указанный код необязателен.

Подпрограммы не поддерживаются

Когда система ЧПУ не поддерживает использование подпрограмм, а последовательность технологических команд содержит NC-подпрограммы и их вызов, можно выполнить "развёртывание" NC-подпрограмм, то есть во всех местах вызова подпрограммы необходимо вставить саму подпрограмму вместо строки вызова. Для этого в процедуру обработки технологической команды вызова подпрограммы PPFUN CALLSUB необходимо 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 этом случае не требуется.

Сопутствующие ссылки

Операторы работы с NC-подпрограммами

Оператор вывода NC-подпрограммы NCSUB.OUTPUT

Оператор определения имени NC-подпрограммы NCSUB.NAME

Оператор определения начальной метки NC-подпрограммы NCSUB.STARTLABEL

Оператор определения конечной метки NC-подпрограммы NCSUB.ENDLABEL

ReplNCStr - оператор замены строк в файле УП

Оператор ReplNCStr заменяет все строки From$ на строки To$ в файле File$ (или во всех файлах, если File$ не указан или является пустой строкой "").

Формат

```
ReplNCStr(<From$>, <To$>{, <File$>})
```

Описание

`From$`, `To$`, `File$` – строковые выражения.

Пример

```
! Замена всех строк "placeholder" на "world!" во всех файлах УП
Output "hello, placeholder" ! файл УП: hello, placeholder
ReplNCStr("placeholder", "world!", "") ! файл УП: hello, world!
! Замена одной строки файла УП на две строки:
! Hello -> Hello,
! World!
Output "Hello"
ReplNCStr("Hello", "Hello," + Chr(13) + Chr(10) + "World!")
```

Содержание

Подсистема .NET-постпроцессоров создаёт постпроцессоры на .NET (C#). На этой странице перечислено содержимое раздела; темы ниже охватывают систему и её инструменты, пользовательский интерфейс и общий справочник CLData, читаемый обеими подсистемами постпроцессирования.

- Внешние ссылки и примеры
- .NET-постпроцессоры
- Введение
- Организация системы постпроцессирования
  - Запуск постпроцессирования
  - Инструменты разработки постпроцессоров
  - Набор файлов постпроцессора
- Пользовательский интерфейс
  - CLData Viewer
  - Окно настроек постпроцессора
  - Окно регистров
- Как писать постпроцессоры
  - Как подготовить компьютер к написанию постпроцессоров
- Разработка постпроцессоров с помощью ИИ
  - Как устроена разработка с помощью ИИ
  - Настройка инструментов
  - Организация рабочего пространства
  - Разработка через CAM Agent
  - Разработка в Visual Studio Code
  - Что дают расширения
  - Рабочий цикл SPPX
  - Рабочий цикл .NET
  - Проверка и подтверждение результата
  - Устранение проблем
  - Расширенная настройка
- Справочник CLData
  - Функции и операторы доступа к CLData
    - Именованные параметры CLData
    - Предопределённый массив CLD
    - Текущая команда (оператор Cmd)
    - Обход команд CLData
    - Файлы CLData (оператор CLDFile)
    - Информация о проекте (оператор Project)
    - Массив GMA

- Информация о проекте в CLData
- Описание технологических команд
  - AXESBRAKE - Управление тормозами осей станка
  - CIRCLE - Движение по окружности
  - CLAMP - Зажим креплений
  - COMMENT - Комментарии
  - COOLNT - Охлаждение
  - CUTCOM - Коррекция инструмента
  - CYCLE - Циклы обработки отверстий
    - CYCLE BORE5(209) - Цикл сверления G85
    - CYCLE BORE6(210) - Цикл сверления G86
    - CYCLE BORE7(211) - Цикл сверления G87
    - CYCLE BORE8(212) - Цикл сверления G88
    - CYCLE BORE9(213) - Цикл сверления G89
    - CYCLE BRKCHP(288) - Глубокое сверление с ломкой стружки
    - CYCLE DEEP(153) - Глубокое сверление с удалением стружки
    - CYCLE DRILL(163) - Цикл сверления G81
    - CYCLE FACE(81) - Цикл сверления G82
    - CYCLE TAP(168) - Цикл сверления G84
  - DELAY - Задержка
  - EDMMOVE - Электроэрозионное перемещение
  - EFFECTOR - Активация концевого эффектора
  - EXTCYCLE - Расширенный цикл
    - EXTCYCLE BRKCHP - Токарный цикл сверления с ломкой стружки
    - EXTCYCLE DEEP - Токарный цикл сверления с удалением стружки
    - EXTCYCLE DRILL - Токарный цикл сверления (G81)
    - EXTCYCLE FACE - Токарный цикл сверления (G82)
    - EXTCYCLE LATHEFINISH - Токарный чистовой цикл (G70, G73)
    - EXTCYCLE LATHEGROOVE - Токарный цикл прорезки канавок (G74, G75)
    - EXTCYCLE LATHEROUGH - Токарный черновой цикл (G71, G72)
    - EXTCYCLE LATHETHREAD - Токарный цикл нарезания резьбы (G76)
    - EXTCYCLE LATHETHREADG92(404) - Токарный однопроходный цикл нарезания резьбы (G92)
    - EXTCYCLE TAP - Токарный цикл нарезания резьбы метчиком (G84)
    - EXTCYCLE W5DBore5 - Цикл сверления (G85)
    - EXTCYCLE W5DBore6 - Цикл сверления (G86)
    - EXTCYCLE W5DBore7 - Цикл сверления (G87)
    - EXTCYCLE W5DBore8 - Цикл сверления (G88)
    - EXTCYCLE W5DBore9 - Цикл сверления (G89)
    - EXTCYCLE W5DChipBreaking - Цикл сверления с ломкой стружки (G73)

        - EXTCYCLE W5DChipRemoving - Цикл сверления с удалением стружки (G83)
        - EXTCYCLE W5DDrill - Цикл сверления (G81)
        - EXTCYCLE W5DFace - Цикл сверления (G82)
        - EXTCYCLE W5DHolePocketing - Цикл выборки отверстия
        - EXTCYCLE W5DTap - Цикл нарезания резьбы метчиком (G84)
        - EXTCYCLE W5DThreadMill - Цикл фрезерования резьбы
        - EXTCYCLE WProbing - Измерительный цикл
            - Дополнительные параметры
            - Параметры измерения бобышки
            - Параметры измерения внешнего угла с двумя стенками
            - Параметры измерения внутреннего угла с двумя стенками
            - Параметры измерения внешнего прямоугольника
            - Параметры измерения канавки
            - Защищённые параметры измерения канавки
            - Параметры измерения отверстия
            - Защищённые параметры измерения отверстия
            - Параметры измерения внутреннего прямоугольника
            - Защищённые параметры измерения внутреннего прямоугольника
            - Параметры измерения поверхности
            - Параметры измерения внешнего угла с тремя стенками
            - Параметры измерения внутреннего угла с тремя стенками
            - Параметры измерения перемычки (по трём точкам)
            - Параметры измерения перемычки
    - FEDRAT - Подача
    - Filter - Выходной фильтр (глобальный обработчик)
    - FINI - Завершающая запись
    - FROM - Исходная точка
    - GOHOME - Возврат в исходное положение
    - GOTO - Линейный переход
    - HEAD - Номер шпиндельной головки
    - INSERT - Вставка
    - INTERPOLATION - Режим интерполяции
    - LOADTL - Загрузка инструмента
    - MULTIARC - Многоосевое движение по окружности
    - MULTIGOTO - Многокоординатное перемещение
    - OPSKIP - Необязательный пропуск
    - OPSTOP - Вспомогательный останов
    - ORIGIN - Исходные координаты
    - PALETA - Смена палеты
    - PARTNO - Номер детали

- PHYSICGOTO - Перемещение по физическим осям станка
- PLANE - Рабочая плоскость
- POWER - Регулировка мощности
- PPFUN - Функция постпроцессора
  - PPFUN CALLSUB(52) - Команда вызова подпрограммы УП
  - PPFUN ENDSUB(51) - Команда окончания подпрограммы УП
  - PPFUN ENDTECHINFO(59) - Окончание технологической информации операции
  - PPFUN PPFUN(500) - Команда контроля стойкости инструмента
  - PPFUN STARTSUB(50) - Команда начала подпрограммы УП
  - PPFUN TECHINFO(58) - Команда технологических параметров
  - PPFUN WEDMConditions(56) - Команда задания условий электроэрозионной обработки
  - PPFUN WEDMPassItem(57) - Свойства прохода проволочной электроэрозионной обработки
- PPRINT - Печать постпроцессора
- RAPID - Ускоренная подача
- ROTABL - Поворот стола
- SAFPOS - Точка смены инструмента
- SELWORKPIECE - Выбор активной державки заготовки
- SINGLETHREAD - Однопроходная резьба
- SPINDL - Шпиндель
- STOP - Останов
- STRUCTURE - Структура траектории
- SYNCAXES - Синхронизация движения осей
- TAKEOVER - Перехват заготовки
- TLCONTACT - Контакт инструмента
- WAIT - Ожидание точки синхронизации

Внешние ссылки и примеры (.NET-постпроцессоры)

Внешние ссылки, генерируемый справочник API SDK и примеры постпроцессоров для подсистемы .NET-постпроцессирования.

Документация по продукту

Ссылка	URL
Портал СПРУТКАМ	https://sprut.ru/
Руководство по .NET-постпроцессорам	Открыть руководство
Учебник по разработке постпроцессоров	https://docs.sprut.ru/tutorials/lnp/17/index.html
Видеолекции по постпроцессированию	https://rutube.ru/plst/475570/
Загрузка СПРУТКАМ	https://sprut.ru/download/

Разработка с помощью ИИ

Адреса загрузки инструментов, описанных в разделе Разработка постпроцессоров с помощью ИИ.

Ссылка	URL
Инструменты постпроцессирования для VS Code — выпуски	https://gitverse.ru/spruttechnology/postprocessor-tools/releases
Центр документации СПРУТКАМ	https://docs.sprut.ru/portal/

Справочник API

Справочник API SDK .NET-постпроцессирования генерируется автоматически из XML-документации NuGet-пакетов SDK.

- Справочник API SDK .NET-постпроцессирования

Внешние ссылки

Ссылка	URL
Платформа .NET	https://dotnet.microsoft.com/
Загрузка .NET SDK	https://dotnet.microsoft.com/download
Документация языка программирования C#	https://docs.microsoft.com/en-us/dotnet/csharp/
Visual Studio Code	https://code.visualstudio.com/
Visual Studio Code — начало работы	https://code.visualstudio.com/docs/getstarted/userinterface
Расширение C# для VS Code	https://marketplace.visualstudio.com/items?itemName=ms-dotnettools.csharp
Репозиторий пакетов Nuget	https://www.nuget.org/

Примеры постпроцессоров

Вместе с CAM-системой поставляется набор примеров постпроцессоров — в папке Dotnet-постпроцессоров. Любой из них можно открыть в CLData Viewer и Visual Studio Code, изучить исходный код, собрать и использовать как отправную точку для своего постпроцессора.

Дополнительные примеры доступны в репозитории: https://gitverse.ru/spruttechnology/postprocessors-examples.

.NET-постпроцессоры

Это открытая система постпроцессирования на базе платформы .NET. В качестве языка программирования она использует C#, а в качестве среды разработки и отладчика кода — Visual Studio Code.

В CAM-системе есть две независимые подсистемы постпроцессирования; обе актуальны и могут использоваться по вашему выбору. В этом разделе документируется только система на базе .NET. Другая подсистема (Генератор постпроцессоров) описана в разделе Генератор постпроцессоров.

Почему эта система

- Основана на широко распространённых, бесплатных инструментах высшего класса с открытым исходным кодом: .NET, C# и Visual Studio Code. Разработчикам постпроцессоров доступны все современные технологии программирования.
  - Мощь и одновременно простота языка программирования C#: объектно-ориентированный стиль, строгая и безопасная типизация, автоматическая сборка мусора, а также такие возможности, как обобщения (generics), лямбда-выражения, LINQ и кортежи.
  - Подсветка синтаксиса и автодополнение с помощью IntelliSense.
  - Первоклассный отладчик с разными видами точек останова, стеками вызовов, интерактивной консолью, окном локальных переменных и списком наблюдения.
  - Высококлассный редактор кода с огромным числом полезных расширений на все случаи жизни.
  - Неограниченный объём документации, доступной в сети.
  - Документация постпроцессора, доступная прямо в коде в виде комментариев и всплывающих подсказок.
- Привычная концепция на основе регистров из Генератора постпроцессоров, переработанная с нуля на новом уровне с современной объектно-ориентированной парадигмой и с учётом десятилетий опыта разработки постпроцессоров. Это даёт такие преимущества, как:
  - Простое написание постпроцессоров, которые генерируют несколько файлов одновременно.
  - Возможность дописывать в произвольное место файла, а не только в конец.
  - Формирование выходных файлов в любой кодировке.
  - Несколько независимых наборов регистров.
  - Текстовые регистры в дополнение к числовым.
  - Настройки, определяемые разработчиком, индивидуальные для каждого постпроцессора.
  - Встроенная библиотека функций для геометрических расчётов.

- Возможность использовать сторонние библиотеки, широко доступные через Nuget.

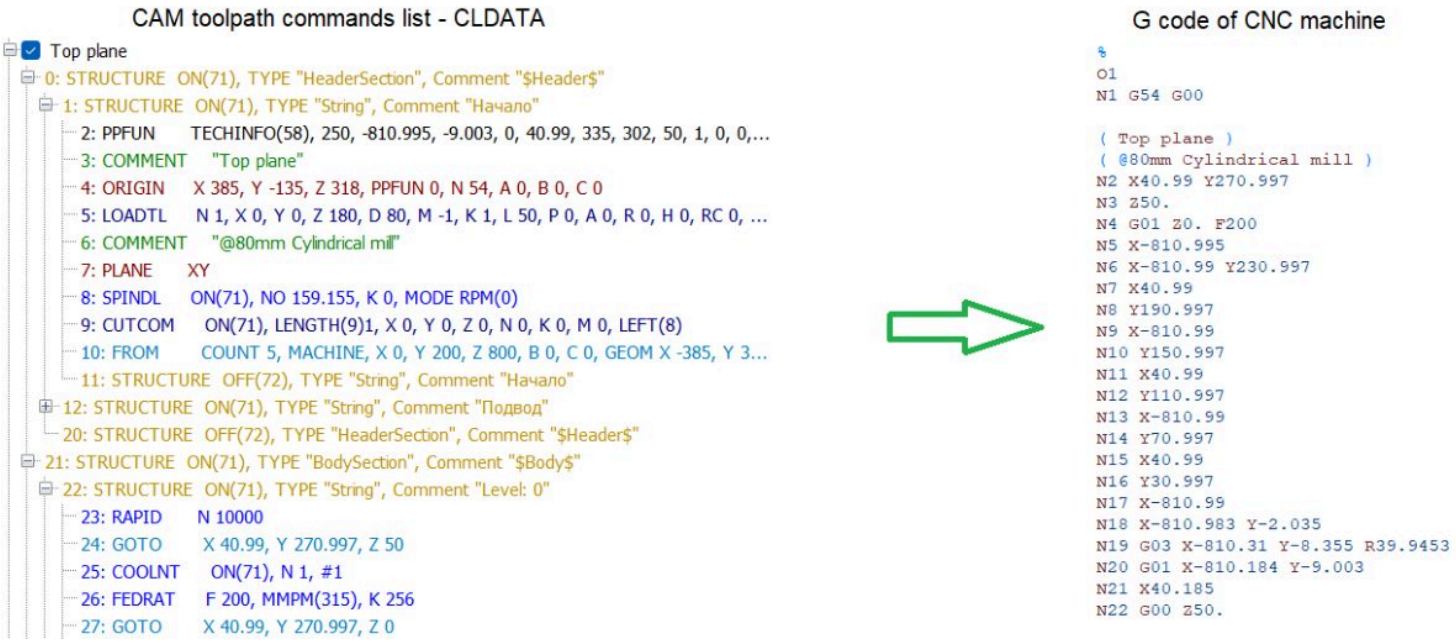
Документация

- Введение — что такое система постпроцессирования и её основные программные модули.
- Организация системы постпроцессирования
  - Запуск постпроцессирования
  - Инструменты разработки постпроцессоров
  - Набор файлов постпроцессора
- Пользовательский интерфейс
  - CLData Viewer
  - Окно настроек постпроцессора
  - Окно регистров
  - Пользовательский интерфейс Visual Studio Code
- Как писать постпроцессоры
  - Как подготовить компьютер к написанию постпроцессоров
- Документация по CLData — общая входная модель, читаемая обеими подсистемами постпроцессирования.
- Внешние ссылки и примеры — справочник API .NET SDK, внешние ссылки и примеры постпроцессоров.

Введение

Что такое система постпроцессирования в целом?

Система постпроцессирования — это программа, которая преобразует траекторию, сформированную CAM-системой, из универсального формата (CLData) в выходные файлы, понятные конкретному станку с ЧПУ.



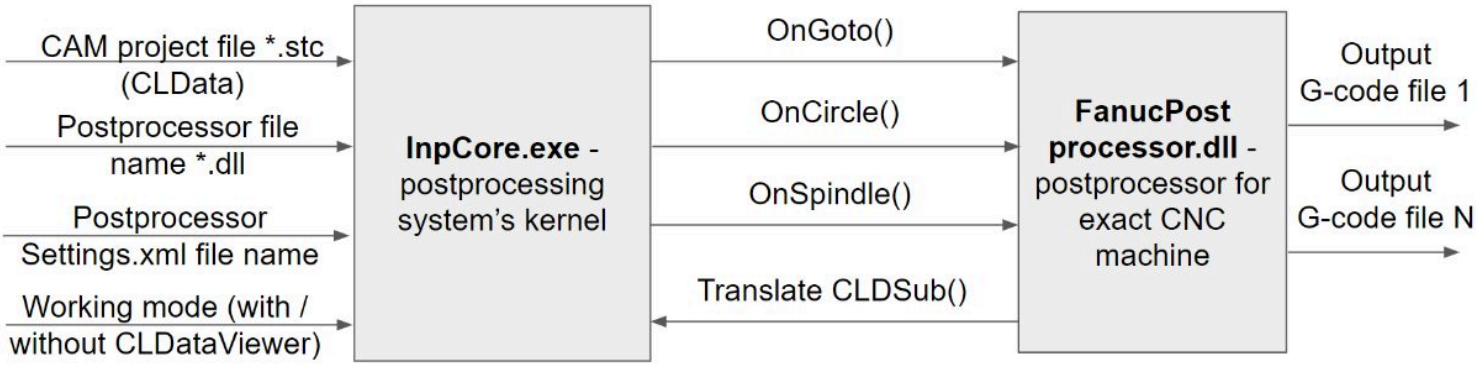
CLData (cutter location data) — это набор команд, таких как загрузка инструмента, перемещения, включение/выключение шпинделя и СОЖ и т. д.

Программа постпроцессирования получает их на вход и преобразует в формат, который требуется конкретному станку с ЧПУ, например в текстовый файл G-кода. Станков с ЧПУ очень много, и каждый тип станка может использовать собственный формат файлов. Поэтому система постпроцессирования должна состоять как минимум из двух частей.

- Первая — часть, независимая от конкретного типа станка. Она должна уметь читать входные CLData и предоставлять их для дальнейшей удобной обработки.
- Вторая часть — настраиваемая часть. Она должна давать гибкий способ трансляции универсальных команд в конкретные выходные файлы.

Основные программные модули системы

С точки зрения программных модулей это выглядит так, как показано ниже.



Первый основной стартовый модуль — утилита InpCore.exe. Это консольное приложение, то есть у него нет пользовательского интерфейса, есть только окно консоли. В этом окне консоли можно ввести параметры командной строки, такие как имя входного файла CLData (stc-проект), имя постпроцессора и некоторые дополнительные настройки (имя выходного файла, номер программы и т. д.), а затем запустить процесс трансляции. Эта утилита читает входной файл CLData и начинает вызывать особый вид подпрограмм — обработчики команд — которые индивидуальны для каждого типа команды. Например, для команды CLData LoadTool она вызовет подпрограмму с именем OnLoadTool, для команды Spindle — подпрограмму OnSpindle, для команды Goto — подпрограмму OnGoto и т. д. По умолчанию эти обработчики команд пусты и ничего не делают. Чтобы переопределить это пустое поведение, нужно загрузить вторую часть системы постпроцессирования — сам постпроцессор.

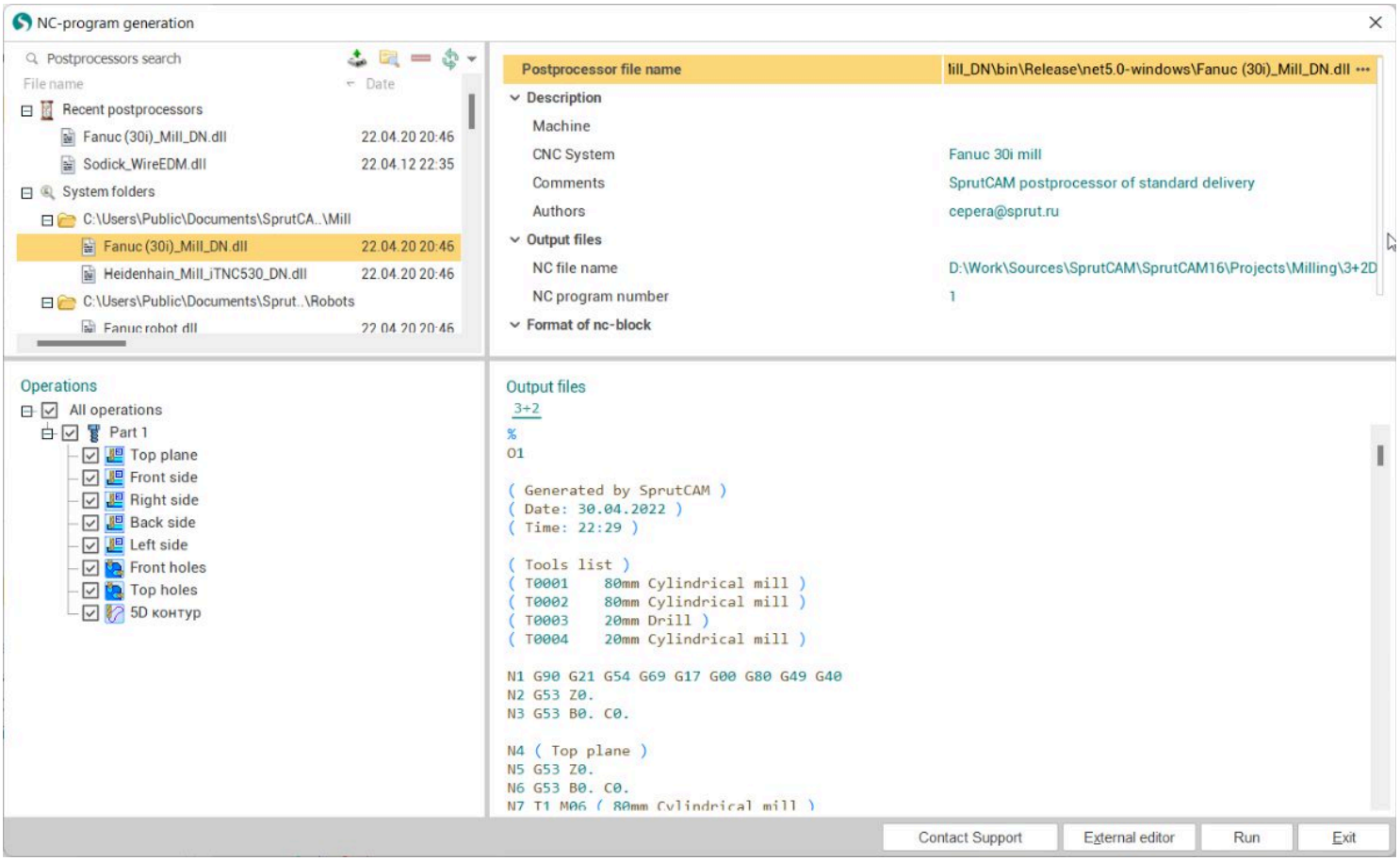
Постпроцессор — это дополнительный динамически загружаемый программный модуль с расширением dll. Имя может быть разным. Например, FanucPostprocessor.dll. Он может реализовать некоторые из обработчиков команд, которые переопределят пустое поведение и будут записывать в выходной файл конкретные инструкции, соответствующие команде, для конкретного станка с ЧПУ. Постпроцессор легко заменить на другой, например HeidenhainPostprocessor.dll, чтобы получить результирующий G-код для другого станка. Вы можете создать собственный постпроцессор, если хотите получить уникальное содержимое выходного файла.

Запуск постпроцессирования

Рассмотрим, как сгенерировать G-код с помощью постпроцессоров.

Окно генерации управляющей программы

Первый, самый простой способ — использовать окно CAM-системы, специально предназначенное для этого. Открыть его можно кнопкой "Postprocessor" на верхней панели инструментов главного окна.

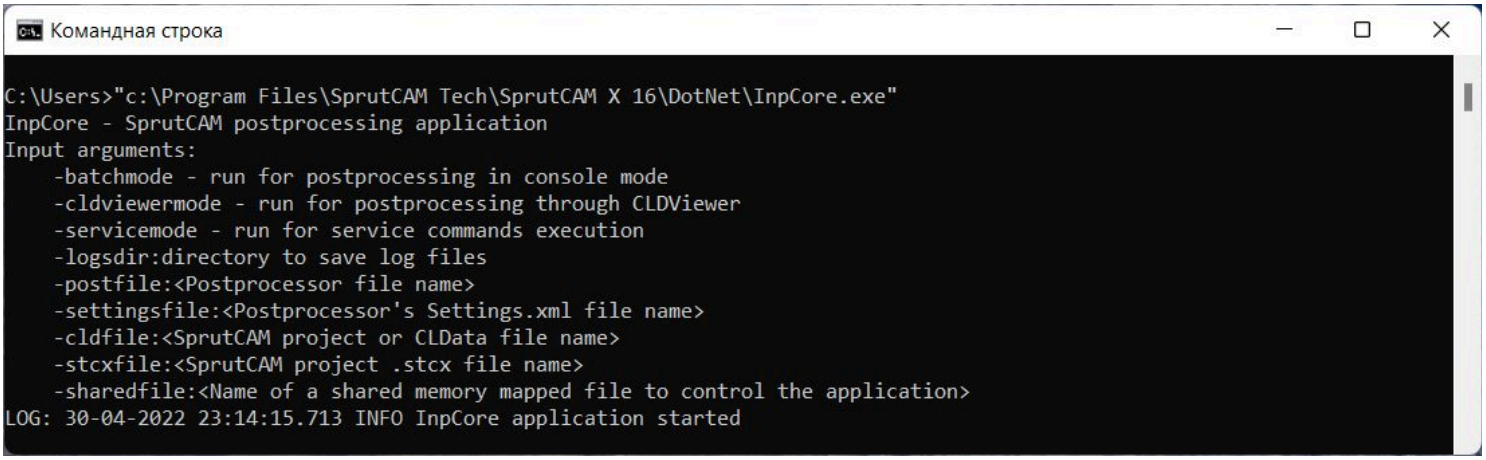


Здесь нужно лишь выбрать желаемый постпроцессор в левом верхнем списке, затем задать некоторые дополнительные настройки в правом верхнем списке свойств, например имя выходного файла УП, и, наконец, нажать кнопку "Run". В нижней части окна вы увидите содержимое сгенерированного файла G-кода. Чтобы получить сам файл, можно использовать пункт всплывающего меню "Show file in Windows explorer" панели с G-кодом.

Запуск постпроцессоров в консольном режиме

Система постпроцессирования — это самостоятельная программа, поэтому её можно запускать и независимо от CAM-системы, так же как CAM-система делает это за кулисами. Для этого просто запустите стандартную командную строку Windows из меню Пуск или сочетанием

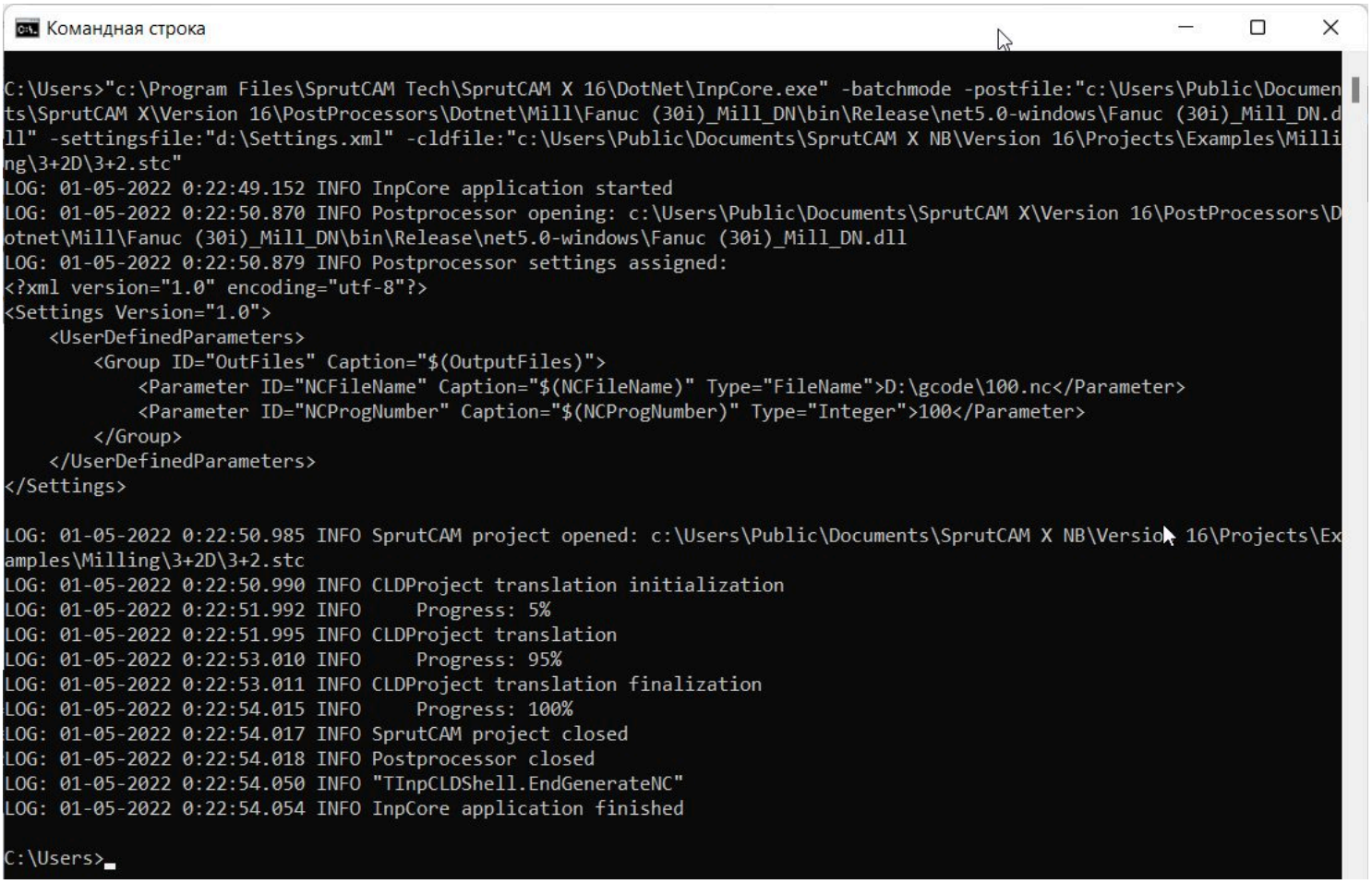
клавиш "Win+R" и введите "cmd". Здесь нужно ввести путь к утилите InpCore.exe из папки установки CAM. Она покажет приветственный экран со списком доступных параметров.



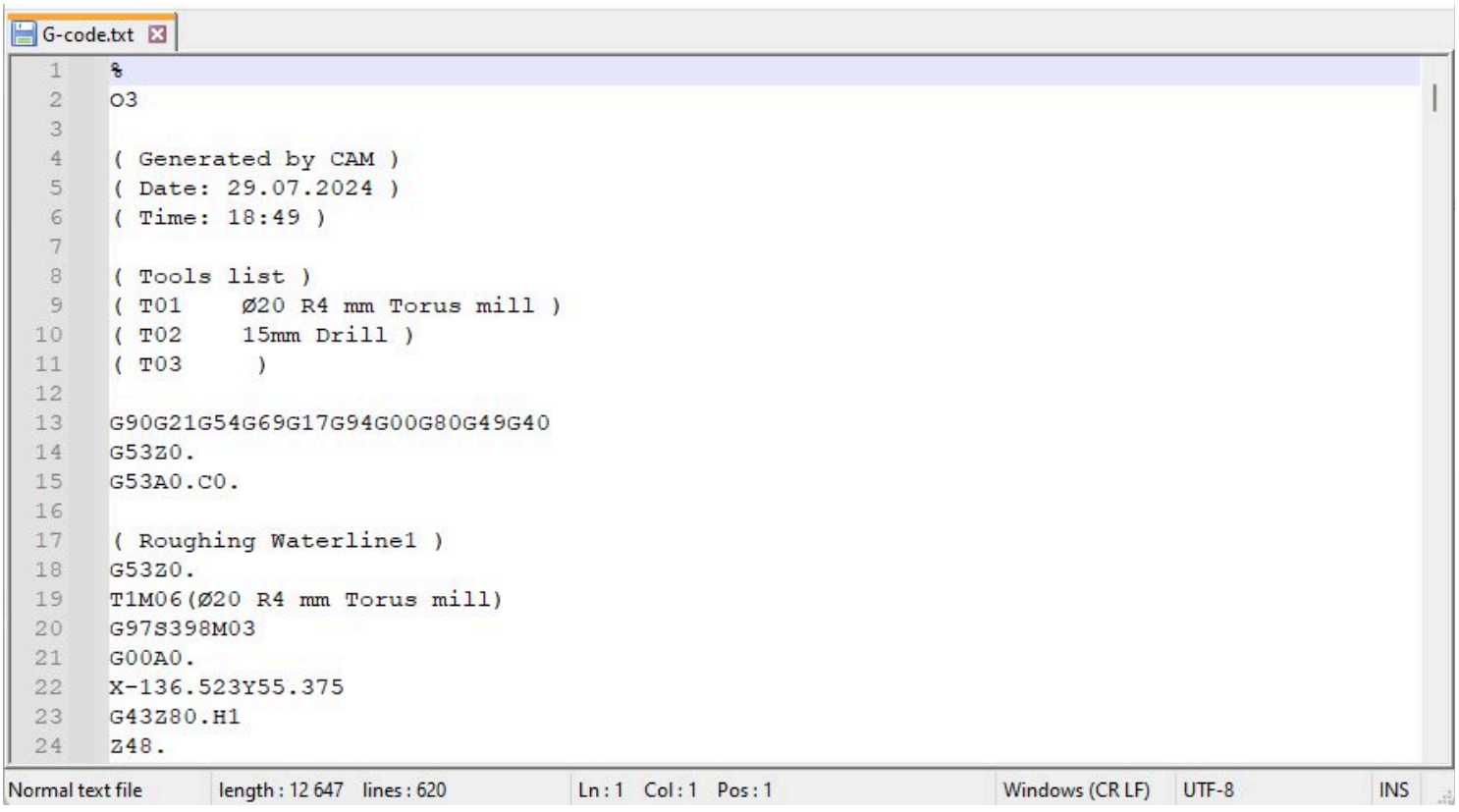
Чтобы сгенерировать G-код, следует указать следующие параметры.

- "-batchmode" — для запуска самостоятельного постпроцессирования без отладки постпроцессора через CLData Viewer.
- "-postfile:`<путь к dll постпроцессора>`" — полный путь к dll-файлу постпроцессора, который вы хотите использовать.
- "-cldfile:`<путь к проекту CAM stcp>`" — полный путь к файлу stcp проекта CAM, который содержит траекторию, которую вы хотите постпроцессировать.
- "settingsfile:`<путь к файлу settings.xml постпроцессора>`" — полный путь к xml-файлу, который должен содержать настройки, специфичные для конкретного постпроцессора. Например, внутри него можно указать имя выходного файла, номер программы G-кода, имя сотрудника и т. д.

Ниже приведён пример постпроцессирования в консольном режиме.



И результирующий G-код.



Инструменты разработки постпроцессоров

Постпроцессор — это обычный программный модуль с расширением dll, поэтому для создания собственных постпроцессоров следует использовать инструменты разработки программ. Чтобы сделать разработку постпроцессоров широкодоступной, простой и удобной, эта система построена на основе хорошо известных, бесплатных и удобных инструментов.

Платформа .NET

Платформа .NET (dotnet) — это бесплатный, кроссплатформенный набор инструментов с открытым исходным кодом, позволяющий создавать различные типы приложений. Чтобы начать создавать приложения .NET, нужно скачать и установить .NET SDK (Software Development Kit) — набор утилит, который помогает компилировать двоичные программные модули из исходных файлов, а также содержит богатый набор стандартных библиотек для упрощения процесса создания программ, поскольку предоставляет большое число готовых решений для выполнения рутинных задач (работа с файлами, строками, списками и т. д.). .NET — очень гибкий и расширяемый инструмент, так как позволяет легко подключать сторонние библиотеки из нескольких миллионов доступных в глобальном репозитории nuget. Если вы хотите реализовать что-то обычное (создать стандартный doc- или pdf-файл, прочитать файл базы данных, создать красивый html-отчёт и т. д.), то с высокой вероятностью кто-то другой уже сделал это до вас — просто воспользуйтесь этим.

Visual Studio Code

Visual Studio Code, также часто называемый VS Code, — это бесплатный, лёгкий и мощный редактор исходного кода от Microsoft. Среди возможностей — поддержка отладки, подсветка синтаксиса, интеллектуальное автодополнение кода, сниппеты и рефакторинг кода. Пользователи могут менять тему, сочетания клавиш, настройки и устанавливать расширения, добавляющие дополнительную функциональность. Visual Studio Code — один из самых популярных инструментов среды разработчика во всём мире.

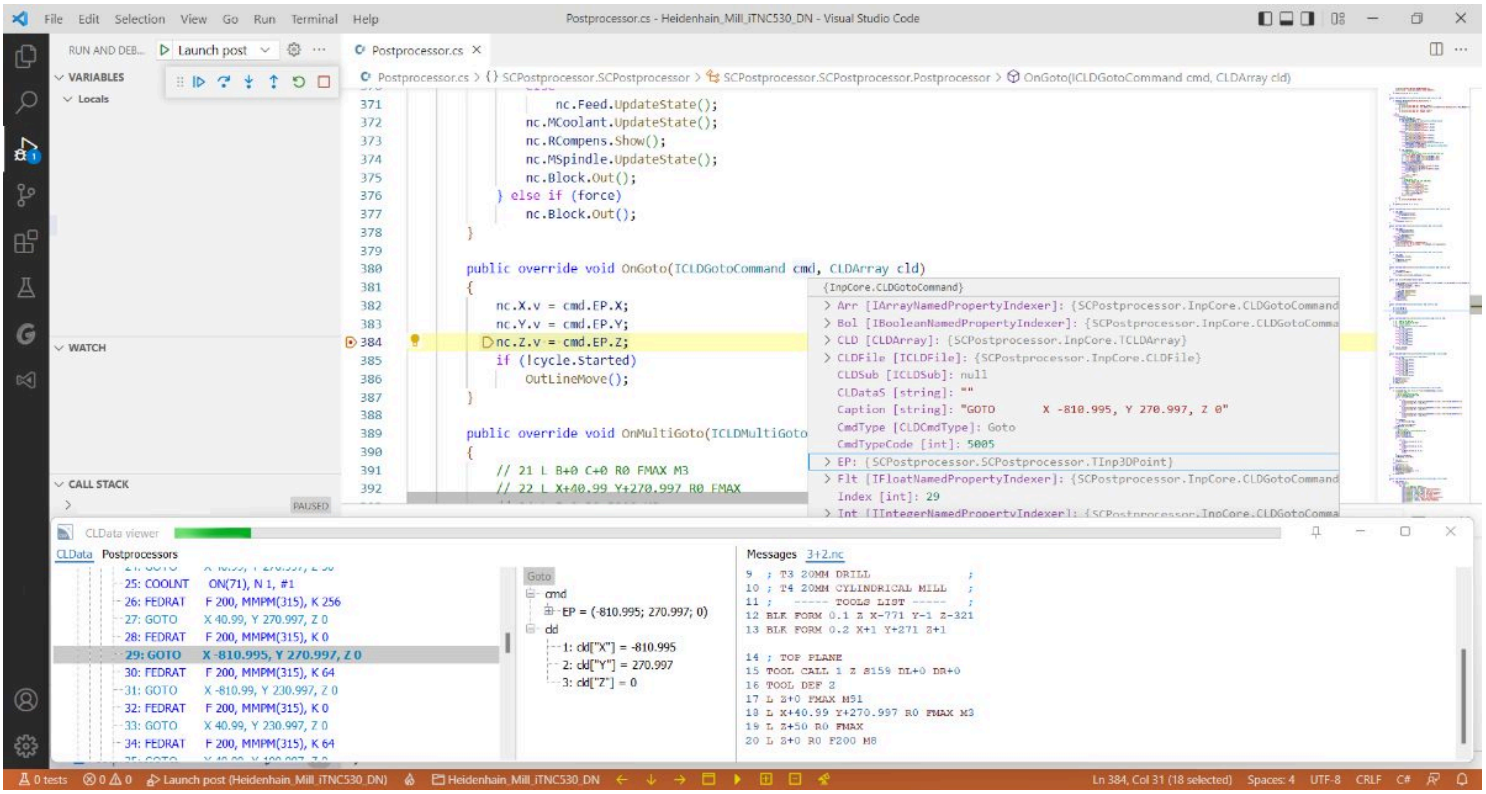
Язык программирования C#

Язык программирования C# (C-sharp) — это простой, современный, объектно-ориентированный язык программирования общего назначения. Он входит в топ-5 языков по общепризнанным рейтингам (таким как TIOBE, PYPL). В сети есть огромное количество учебных материалов и ответов на часто задаваемые вопросы. Чтобы включить поддержку `C#` в VS Code, нужно установить расширение C# из магазина.

CLData Viewer

CLData Viewer — это оконная программа, специально разработанная создателями CAM для отладки постпроцессоров. Она позволяет просматривать траекторию из проекта CAM в виде

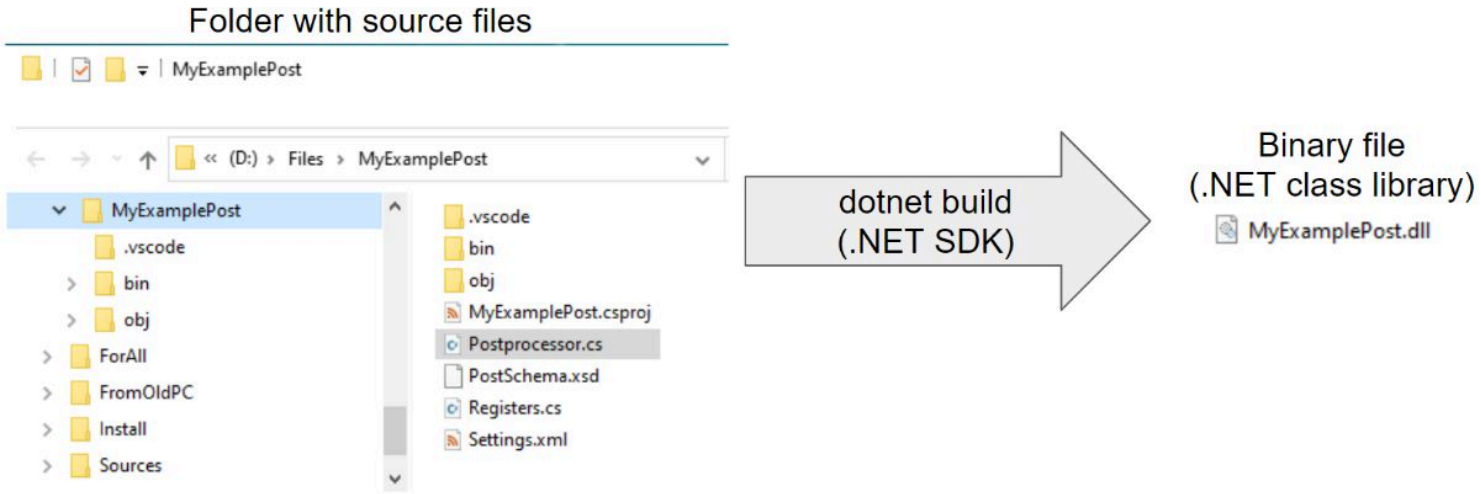
списка команд CLData и их параметров. Прямо во время отладки кода постпроцессора в VS Code CLData Viewer отслеживает выходные файлы в реальном времени. Она позволяет переходить от команды к строке в результирующем выходном файле и обратно. Она поддерживает точки останова CLData, помогает создавать новые постпроцессоры и редактировать их настройки.



Набор файлов постпроцессора

Каждый постпроцессор — это скомпилированная программа, состоящая из нескольких файлов. Все файлы, относящиеся к одному постпроцессору, обычно находятся в одной папке. Набор файлов можно разделить на две основные части.

- Файлы исходного кода — это текстовые файлы, содержащие программный код, который пишет разработчик постпроцессора на языке программирования C#. Исходные файлы нужны только на стороне разработчика постпроцессора. Обычно они не передаются клиенту (тому, кто будет использовать постпроцессор для получения G-кода).
- Двоичный файл постпроцессора с расширением *.dll, который создаётся компилятором платформы dotnet на основе исходных файлов. Компилятор обычно создаёт много двоичных файлов, большинство из которых вспомогательные. Есть только один основной результирующий двоичный dll-файл — файл, имя которого совпадает с именем папки постпроцессора (а также с именем файла проекта *.csproj). Этот файл самодостаточен для генерации G-кода, поэтому клиенту нужно передать только его.



Возьмём в качестве примера содержимое папки, представляющей собой набор файлов одного постпроцессора. Здесь синим цветом обозначены папки, зелёным — исходные текстовые файлы, красным — основной результирующий двоичный файл, серым — вспомогательные двоичные и текстовые файлы.

- MyExamplePost — главная папка постпроцессора. Имя этой папки определяет имя постпроцессора. Оно должно совпадать с именем файла .csproj (если вы хотите переименовать постпроцессор, следует переименовать и то, и другое — папку и файл .csproj).
  - MyExamplePost.csproj — файл C#-проекта постпроцессора. Он не содержит значимого кода постпроцессора. Он просто описывает, что это обычная C#-программа. Обычно в этом файле ничего менять не нужно.

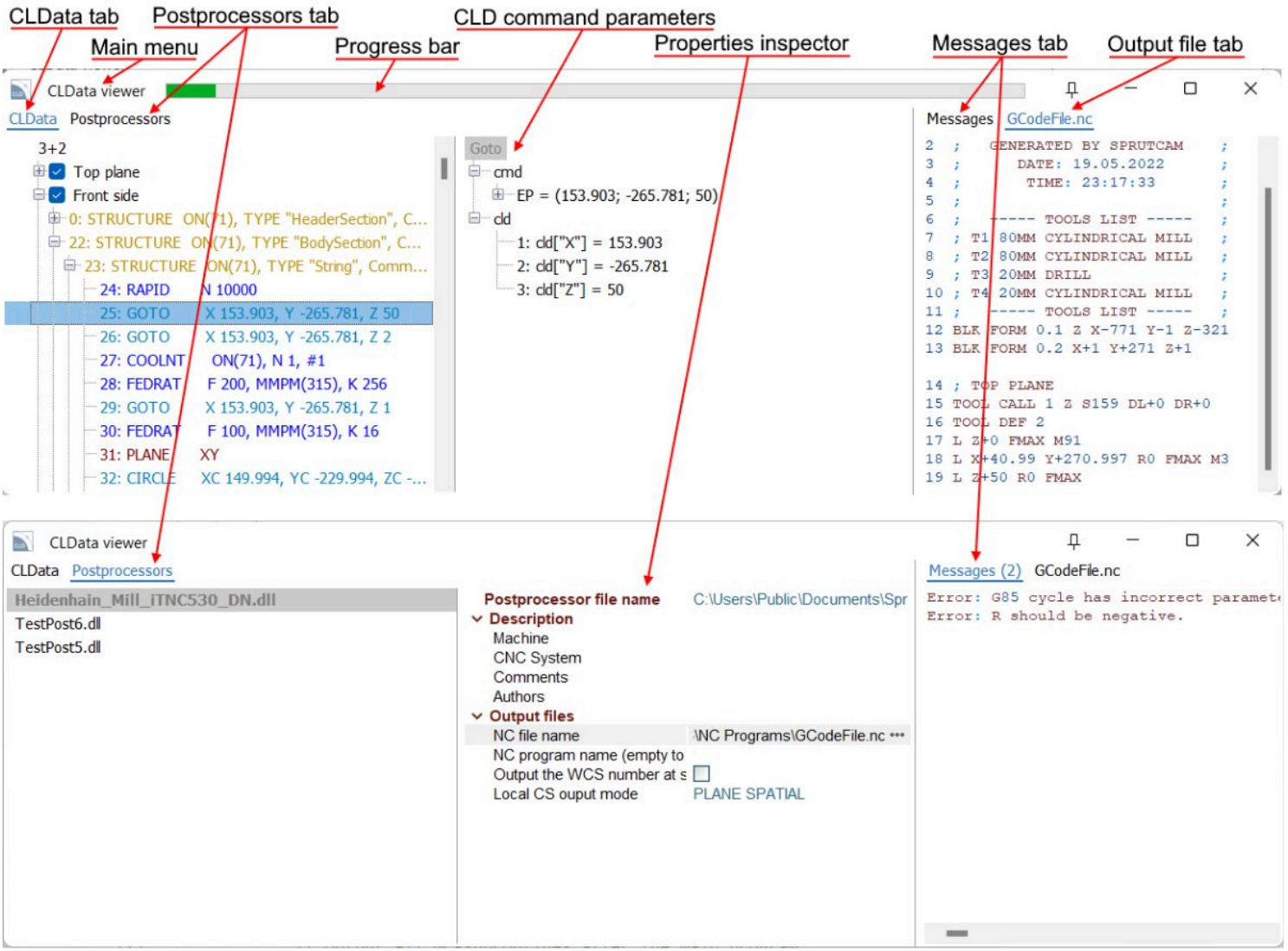
- Postprocessor.cs — главный исходный файл постпроцессора. Именно этот файл содержит значимый программный код, определяющий логику постпроцессора. Разработчик должен писать свой код здесь.
- Registers.cs — вспомогательный исходный файл, который содержит объявления регистров (NCWords), используемых для форматирования и фильтрации текста перед выводом его в G-код. Не размещайте здесь значимый код, используйте его только для объявления регистров. Вы можете изменять его вручную или с помощью окна редактирования регистров в CLData Viewer.
- Settings.xml — этот файл содержит описательные свойства постпроцессора (авторы, имя ЧПУ и станка), а также список дополнительных свойств, созданных разработчиком, которые показываются клиенту при использовании постпроцессора и которые он должен заполнить перед началом генерации G-кода (например, имя выходного файла, номер программы, имя пользователя и т. д.). Для изменения этого файла следует использовать окно "Edit postprocessor XML settings" в CLData Viewer.
- PostSchema.xsd — вспомогательный файл, описывающий правила синтаксиса для файла Settings.xml. Он нужен для подсветки ошибок и автодополнения при ручном редактировании файла Settings.xml. Файл PostSchema.xsd изменять не нужно.
- .vscode — папка, которая содержит настройки Visual Studio Code для автоматизации процесса компиляции и отладки постпроцессора. Обычно здесь ничего менять не нужно.
  - launch.json — файл содержит команду оболочки, которая должна выполняться для запуска отладки постпроцессора при нажатии клавиши "F5".
  - tasks.json — файл содержит команду оболочки, которая должна выполняться для запуска сборки постпроцессора при нажатии "Shift+Ctrl+B".
- bin\Debug — эта папка содержит несколько двоичных файлов, автоматически создаваемых во время компиляции. Если их удалить, они будут автоматически восстановлены при следующей компиляции.
  - MyExamplePost.dll — это основной результирующий самодостаточный выходной двоичный файл постпроцессора. Клиенту нужно поставлять только этот один файл.
  - SCPostprocessor.dll — стандартная библиотека CAM, которая содержит общее описание типов и классов постпроцессирования. Она автоматически копируется из папки установки CAM во время сборки.
  - STDefLib.dll — стандартная библиотека CAM, которая содержит вспомогательные функции общего назначения, доступные для использования постпроцессорами. Она автоматически копируется из папки установки CAM во время сборки.
  - VecMatrLib.dll — стандартная библиотека CAM, которая содержит вспомогательные функции для работы с геометрией, доступные для использования постпроцессорами. Она автоматически копируется из папки установки CAM во время сборки.

        - MyExamplePost.deps.json — этот файл описывает, от каких дополнительных библиотек зависит постпроцессор.
        - MyExamplePost.pdb — файл символов отладки.
        - SCPostprocessor.pdb — файл символов отладки.
        - STDefLib.pdb — файл символов отладки.
        - VecMatrLib.pdb — файл символов отладки.
        - SCPostprocessor.xml — текстовый файл документации для всплывающих подсказок.
        - STDefLib.xml — текстовый файл документации для всплывающих подсказок.
    - obj — автоматически создаваемая папка.
        - множество вспомогательных автоматически создаваемых файлов. Если их удалить, они будут автоматически восстановлены при следующей компиляции.

Главное окно CLData Viewer

CLData Viewer

CLData Viewer — это оконная программа, специально разработанная создателями CAM для отладки постпроцессоров. Она позволяет просматривать траекторию из проекта CAM в виде списка команд CLData и их параметров. Прямо во время отладки кода постпроцессора в VS Code CLData Viewer отслеживает выходные файлы в реальном времени. Она позволяет переходить от команды к строке в результирующем выходном файле и обратно. Она поддерживает точки останова CLData, помогает создавать новые постпроцессоры и редактировать их настройки.



Главное окно CLData Viewer содержит следующие элементы.

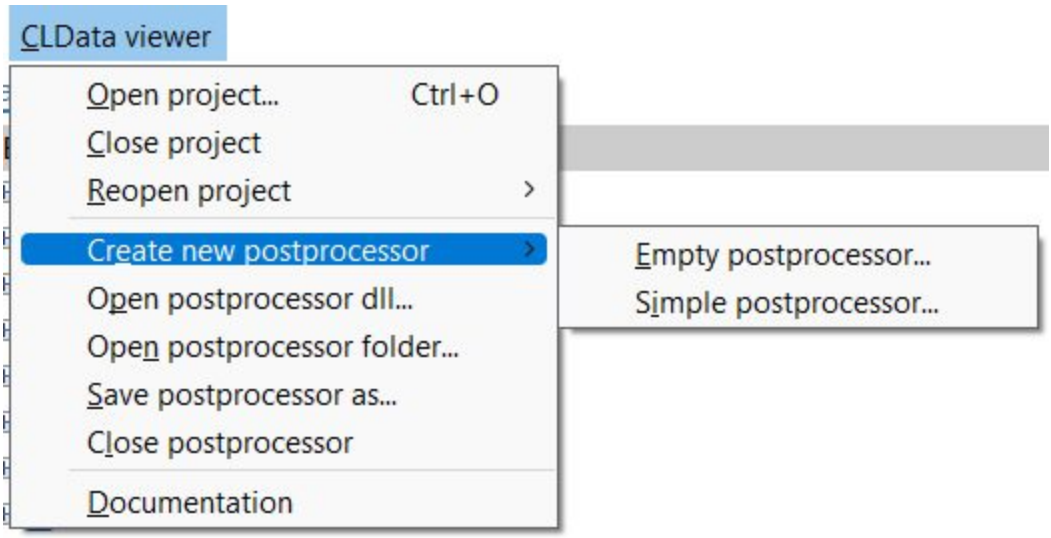
- Главное меню. Оно содержит основные действия, которые можно выполнять в CLData Viewer, например открытие проектов и постпроцессоров.
- Вкладка CLData — список команд CLData текущего открытого проекта CAM-системы. Используйте всплывающее меню для работы с проектами и точками останова на

командах CLData.

- Панель параметров команды CLD показывает список основных параметров команды CLData, выбранной на вкладке CLData. С помощью всплывающего меню этой панели можно легко получить имя параметра, по которому к нему можно обращаться из программного кода постпроцессора.
- Вкладка Postprocessors — список постпроцессоров, открытых в CLData Viewer. Здесь постпроцессор, который последним отлаживался в VS Code, выделен полужирным. Используйте всплывающее меню этой панели для выполнения действий над постпроцессорами.
- Инспектор свойств показывает свойства постпроцессора, выбранного на вкладке постпроцессоров. Здесь можно задать некоторые значения, которые будут входными данными для постпроцессора при генерации выходных файлов G-кода. Например, это может быть имя выходного файла, номер программы и т. д.
- Вкладка выходного файла — показывает содержимое выходного файла (G-код).
- Вкладка Messages — отображает список сообщений (ошибки, предупреждения, информация), которые постпроцессор генерирует во время работы.
- Индикатор прогресса — показывает ход постпроцессирования (какой процент от общего числа команд CLData уже обработан).

Главное меню

Главное меню CLData Viewer выглядит так, как показано на рисунке ниже.



- Open project... — позволяет открыть проект CAM-системы (файл *.stcp) с CLData, который будет использоваться для отладки постпроцессоров. Будет показан стандартный диалог открытия файла.
- Close project — закрывает открытый проект CAM-системы.
- Reopen project — показывает список недавно загруженных проектов CAM-системы для их быстрого открытия.

- Create new postprocessor — помогает создать новый постпроцессор на основе выбранного шаблона.
- Open postprocessor dll.. — позволяет открыть существующий постпроцессор в CLData Viewer, выбрав файл *.dll этого постпроцессора. Будет показан стандартный диалог открытия файла.
- Open postprocessor folder.. — позволяет открыть существующий постпроцессор в CLData Viewer, выбрав папку этого постпроцессора. Будет показан стандартный диалог открытия папки.
- Save postprocessor as... — здесь можно сохранить выбранный постпроцессор под другим именем.
- Close postprocessor — удаляет текущий выбранный постпроцессор из списка постпроцессоров, открытых в CLData Viewer.
- Documentation — открывает онлайн-документацию по постпроцессированию.

Вкладка CLData

Вкладка CLData показывает дерево команд CLData текущего открытого проекта CAM-системы.

Элементы верхнего уровня — это так называемые "файлы CLData", которые могут представлять технологические операции CAM-системы или CLD-подпрограммы.

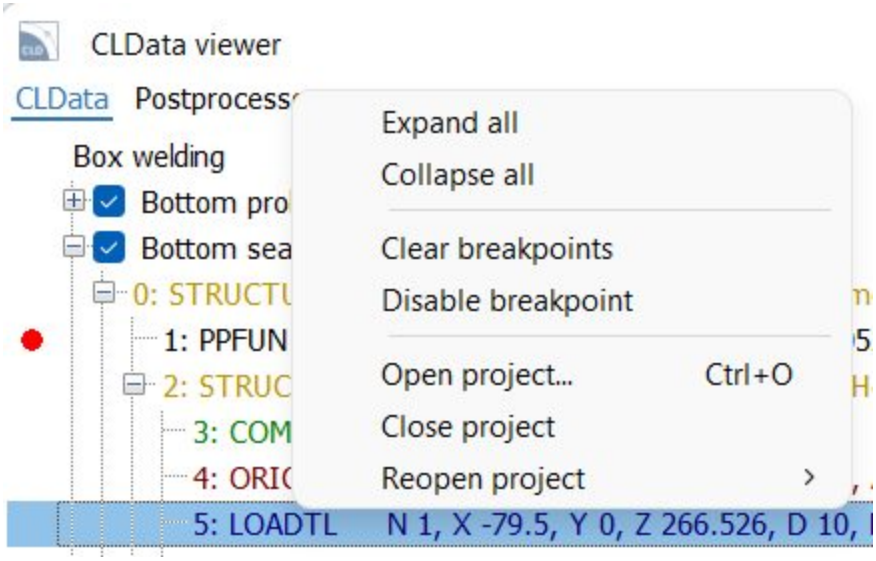
Вложенные узлы — это команды CLData, которые представляют траекторию каждой операции, рассчитанную CAM-системой. Цвет узла обозначает тип команды. Текущая постпроцессируемая команда во время отладки показывается полужирным. Подробный список параметров выбранной команды показывается на отдельной панели справа от списка CLData. С помощью всплывающего меню этой панели можно легко получить имя параметра, по которому к нему можно обращаться из программного кода постпроцессора. Также можно скопировать в буфер обмена значение любого параметра.

Двойной щелчок по команде CLData покажет вам строки внутри выходного файла, которые были сгенерированы этой командой.

Существует концепция так называемых "точек останова CLData". Щёлкнув по полю слева от команд CLData (свободное пространство слева от команд CLData) рядом с нужной командой, можно активировать особый вид точки останова. Она будет показана там жирной красной точкой. Это означает, что постпроцессирование должно приостановиться перед обработкой этой команды, и вы сможете подробнее изучить процесс выполнения программы для этой команды. Это работает только в связке с возможностью "Function breakpoint" Visual Studio Code. На панели точек останова (вкладка "Run and debug") Visual Studio Code можно добавить новую точку останова и указать имя функции, на которой вы хотите остановиться. Здесь следует указать специальное имя функции "StopOnCLData". В этом случае система постпроцессирования будет входить в функцию с таким именем каждый раз, когда достигает

команды CLData, помеченной точкой останова CLData. Разумеется, программный код вашего постпроцессора должен иметь (переопределять) функцию с этим именем. По умолчанию любой постпроцессор уже имеет её, если вы создали его с помощью стандартных шаблонов.

Всплывающее меню вкладки CLData выглядит так.

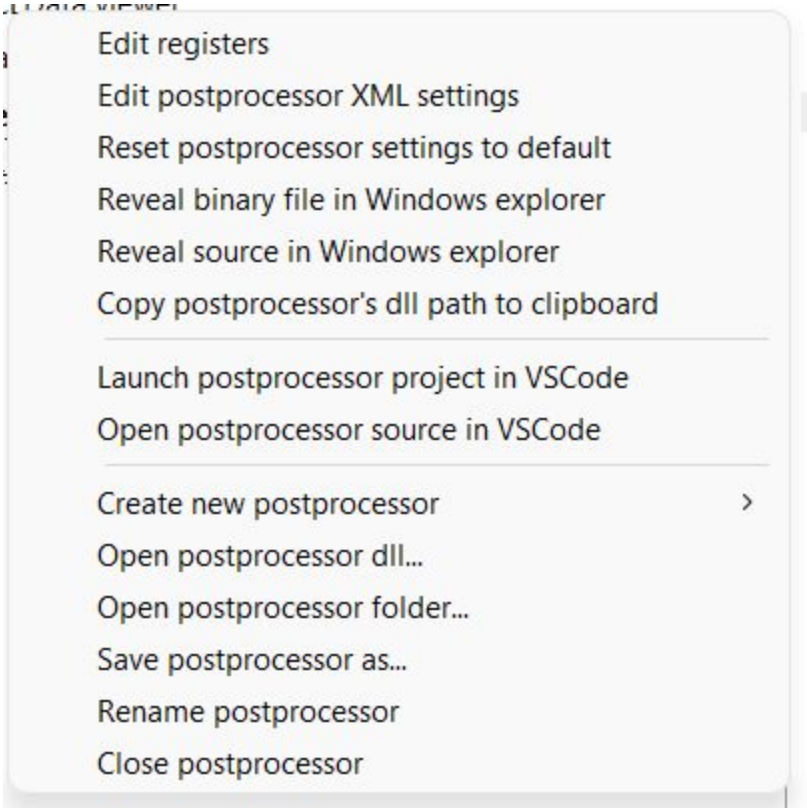


- Expand all — разворачивает все узлы CLData в дереве.
- Collapse all — сворачивает все узлы CLData в дереве.
- Clear breakpoints — удаляет все точки останова CLData из CLData Viewer.
- Disable breakpoint — временно деактивирует точку останова CLData на выбранной команде, пока вы снова её не включите.
- Open project... — позволяет открыть проект CAM-системы (файл *.stcp) с CLData, который будет использоваться для отладки постпроцессоров. Будет показан стандартный диалог открытия файла.
- Close project — закрывает открытый проект CAM-системы.
- Reopen project — показывает список недавно загруженных проектов CAM-системы для их быстрого открытия.

Вкладка Postprocessors

Вкладка постпроцессоров позволяет просматривать и редактировать параметры нескольких постпроцессоров, открытых в CLData Viewer. Левая часть показывает список постпроцессоров, правая часть показывает инспектор с параметрами выбранного постпроцессора. Постпроцессор, который последним отлаживался в Visual Studio Code, выделен в списке полужирным.

Для выполнения действий над постпроцессорами следует использовать всплывающее меню.



- Edit registers — открывает окно редактирования списка регистров выбранного постпроцессора.
- Edit postprocessor XML settings — открывает окно редактирования списка настроек выбранного постпроцессора.
- Reset postprocessor settings to default — восстанавливает значение по умолчанию для каждого параметра выбранного постпроцессора, определённого в его файле "settings.xml". Когда вы вводите какие-то значения в инспекторе свойств (например, имя выходного файла), это влияет только на процесс отладки и не сохраняется внутри самого постпроцессора. Чтобы изменить значения по умолчанию, используйте окно редактирования списка настроек или отредактируйте файл вручную.
- Reveal binary file in Windows explorer — открывает проводник Windows в папке, где расположен результирующий двоичный файл *.dll постпроцессора, и выделяет его.
- Reveal source in Windows explorer — открывает проводник Windows в папке, где расположен файл исходного кода Postprocessor.cs постпроцессора, и выделяет его.
- Copy postprocessor's dll path to clipboard — копирует в буфер обмена путь к результирующему двоичному файлу *.dll выбранного постпроцессора.
- Launch postprocessor project in VSCode — запускает Visual Studio Code (или активирует существующее окно) и открывает в нём папку выбранного постпроцессора. После этого его можно запускать и отлаживать. Либо можно просто дважды щёлкнуть по постпроцессору, чтобы сделать то же самое.
- Open postprocessor source in VSCode — запускает Visual Studio Code (или активирует существующее окно) и открывает в нём файл исходного кода Postprocessor.cs выбранного постпроцессора именно как вспомогательную дополнительную вкладку файла. Он не

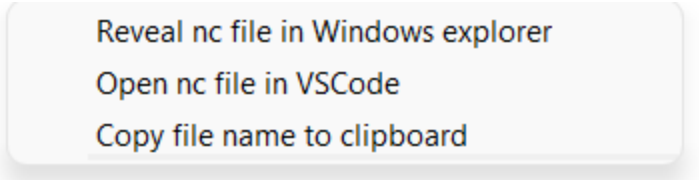
закрывает проект, открытый в данный момент в VS Code, поэтому не позволит запускать и отлаживать этот постпроцессор (потому что для отладки его нужно открыть как папку), но вы сможете его просматривать. Это может быть полезно, когда нужно сравнить исходный код нескольких постпроцессоров и скопировать его из одного в другой.
- Create new postprocessor — помогает создать новый постпроцессор на основе выбранного шаблона.
- Open postprocessor dll.. — позволяет открыть существующий постпроцессор в CLData Viewer, выбрав файл *.dll этого постпроцессора. Будет показан стандартный диалог открытия файла.
- Open postprocessor folder.. — позволяет открыть существующий постпроцессор в CLData Viewer, выбрав папку этого постпроцессора. Будет показан стандартный диалог открытия папки.
- Save postprocessor as... — здесь можно сохранить выбранный постпроцессор под другим именем.
- Rename postprocessor — позволяет легко и безопасно переименовать выбранный постпроцессор.
- Close postprocessor — удаляет текущий выбранный постпроцессор из списка постпроцессоров, открытых в CLData Viewer.

Вкладки выходных файлов и сообщений

Эти вкладки могут показать вам результаты работы после запуска постпроцессора в Visual Studio Code.

- Вкладка выходного файла — показывает содержимое выходного файла (G-код). Вкладок файлов может быть несколько, если постпроцессор генерирует много файлов одновременно. Она обновляется динамически во время работы и отладки постпроцессора. Двойной щелчок по строке в файле покажет вам команду CLData, которая сгенерировала эту строку.
- Вкладка Messages — отображает список сообщений (ошибки, предупреждения, информация), которые постпроцессор генерирует во время работы.

Всплывающее меню этой панели содержит следующие пункты.



- Reveal nc file in Windows explorer — открывает проводник Windows в папке, где расположен результирующий выходной файл, и выделяет его.
- Open nc file in VSCode — запускает Visual Studio Code (или активирует существующее окно) и открывает в нём результирующий выходной файл как вспомогательную дополнительную

вкладку файла. Он не закрывает проект, открытый в данный момент в VS Code.

- Copy file name to clipboard — копирует имя выбранного файла с полным путём в буфер обмена.

Окно настроек постпроцессора

Что такое настройки?

У каждого постпроцессора есть некоторый набор параметров, которые можно разделить на три основные группы.

1. Описательные свойства, такие как имя станка и системы ЧПУ, комментарии и список авторов.
2. Общие свойства, определяющие способ обработки некоторых часто встречающихся случаев. Например, следует ли разбивать движения по дуге на линии, половины или четверти.
3. Параметры, определяемые пользователем — это список дополнительных свойств, разработанных создателем конкретного постпроцессора, которые показываются пользователю постпроцессора и которые он должен заполнить перед началом генерации G-кода. Например, путь к выходному файлу, имя или номер программы, имя сотрудника и т. д.

Постпроцессор хранит свои настройки в отдельном файле settings.xml, который размещается в папке исходных файлов рядом с главным файлом postprocessor.cs. При компиляции этот файл встраивается внутрь главного двоичного dll-файла постпроцессора.

Пример содержимого файла Settings.xml

Ниже приведён пример содержимого файла settings.xml для реального постпроцессора робота Fanuc.

▼ Settings.xml — нажмите, чтобы развернуть.

```
<?xml version="1.0" encoding="utf-8"?>
<?xml-model href="PostSchema.xsd"?>
<Settings Version="1.0">
  <Description>
    <Comments />
    <CNCSystem />
    <MachineName />
    <Authors />
  </Description>
  <Common>
    <BreakArcs>No</BreakArcs>
    <!-- No, Cuts, Halfs, Quaters -->
    <BreakHelix>false</BreakHelix>
    <!-- false, true -->
    <ArcTolerance>0.01</ArcTolerance>
    <MaxCircleRadius>0</MaxCircleRadius>
```

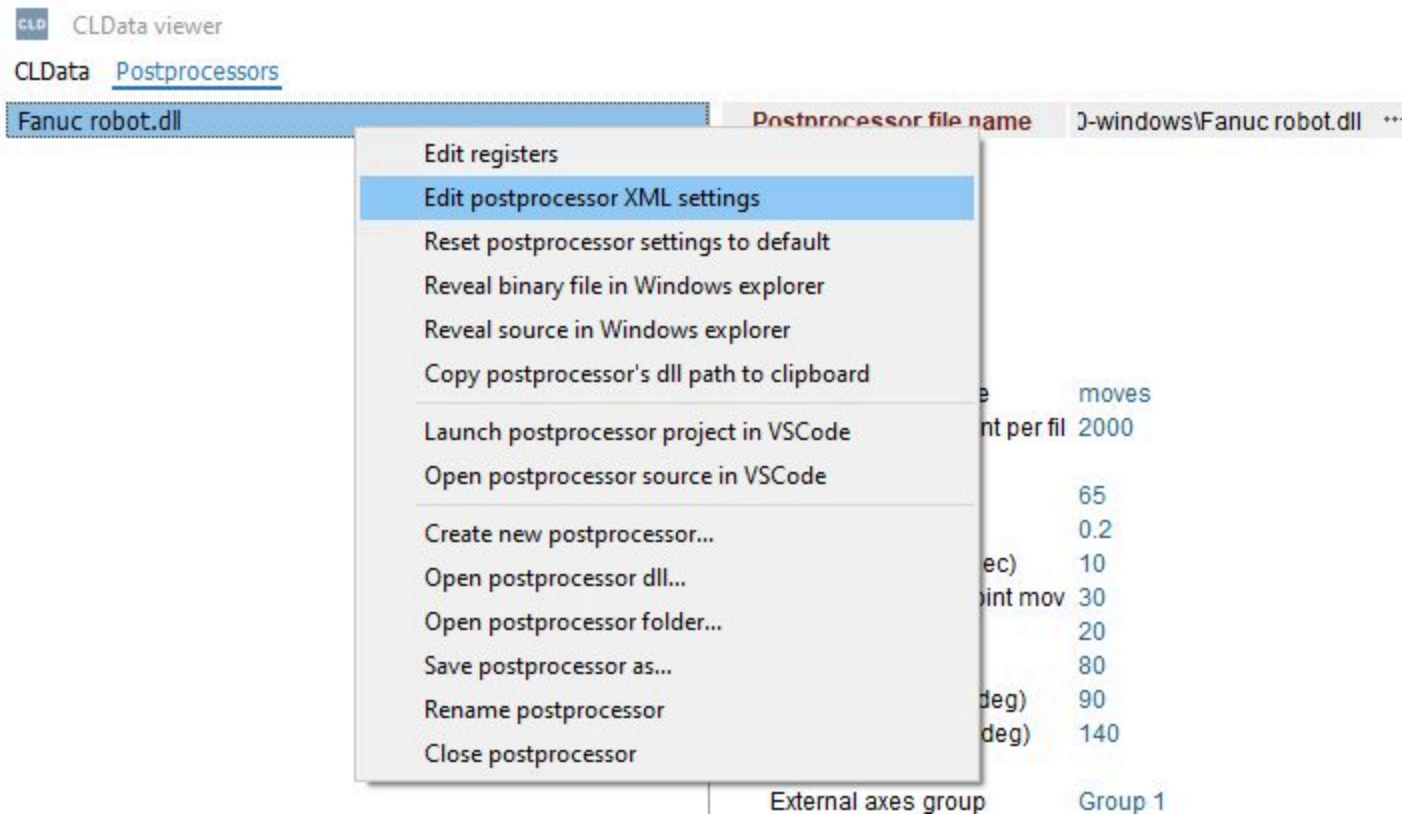
```
    <DecimalSeparator>.</DecimalSeparator>
  </Common>
  <UserDefinedParameters>
    <Group ID="OutFiles" Caption="$(OutputFiles)">
      <Parameter ID="OutputFolder" Caption="$(NCFilesDir)" Type="FolderName" />
      <Parameter ID="LSFileName" Caption="*.ls files short name"
Type="String">moves</Parameter>
      <Parameter ID="MaxMoveCount" Caption="Max movement count per file"
Type="Integer">2000</Parameter>
    </Group>
    <Group ID="Smoothing" Caption="Smoothing">
      <Parameter ID="DefaultACC" Caption="Default ACC value"
Type="Integer">65</Parameter>
      <Parameter ID="ShortLength" Caption="Short length"
Type="Double">0.2</Parameter>
      <Parameter ID="StartVelocity" Caption="Start velocity
(mm/sec)" Type="Double">10</Parameter>
      <Parameter ID="CntValueRapid" Caption="CNT for Rapid or Joint move (%)"
Type="Integer">30</Parameter>
      <Parameter ID="CntValueMin" Caption="Min CNT value (%)"
Type="Integer">20</Parameter>
      <Parameter ID="CntValueMax" Caption="Max CNT value (%)"
Type="Integer">80</Parameter>
      <Parameter ID="AngMin" Caption="Angle of min CNT (deg)"
Type="Integer">90</Parameter>
      <Parameter ID="AngMax" Caption="Angle of max CNT (deg)"
Type="Integer">140</Parameter>
    </Group>
    <Group ID="Format" Caption="Format">
      <Parameter ID="ExtAxesGroup" Caption="External axes group" Type="Integer">
        <Value Caption="Group 1" Default="true">1</Value>
        <Value Caption="Group 2">2</Value>
      </Parameter>
      <Parameter ID="RobotHolds" Caption="Robot holds" Type="Integer">
        <Value Caption="Tool" Default="true">0</Value>
        <Value Caption="Part">1</Value>
      </Parameter>
      <Parameter ID="UFrameFormat" Caption="UFRAME format" Type="Integer">
        <Value Caption="UFRAME_NUM=1" Default="true">0</Value>
        <Value Caption="UFRAME[1]=XYZABC">1</Value>
      </Parameter>
      <Parameter ID="UToolFormat" Caption="UTOOL format" Type="Integer">
        <Value Caption="UTOOL_NUM=1" Default="true">0</Value>
        <Value Caption="UTOOL[1]=XYZABC">1</Value>
      </Parameter>
    </Group>
```

```
    </UserDefinedParameters>
  </Settings>
```

Разумеется, этот файл можно редактировать вручную внутри Visual Studio Code. Файл PostSchema.xsd даже содержит правила синтаксиса, которые для этой цели обеспечат подсветку ошибок и автодополнение предопределённых слов. Но внутри CLData Viewer есть окно, специально предназначенное для редактирования настроек.

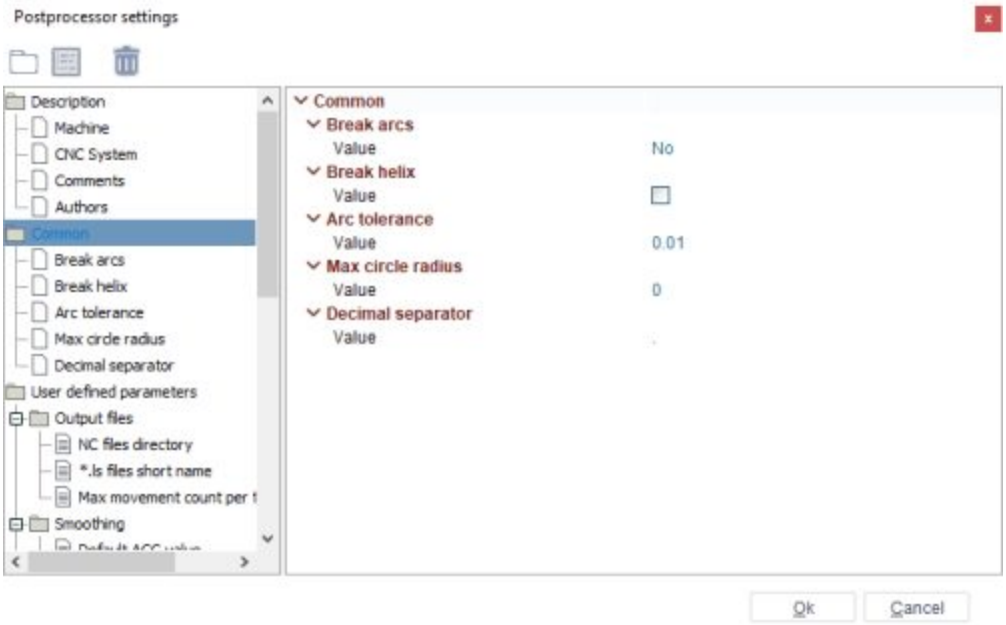
Окно настроек постпроцессора

Окно настроек постпроцессора позволяет работать с xml-настройками постпроцессора наиболее удобным способом. Чтобы открыть окно, нужно выбрать постпроцессор в CLData Viewer и щёлкнуть по нему правой кнопкой мыши. После этого нажмите Edit postprocessor XML settings в контекстном меню.



Окно настроек постпроцессора отображает параметры в виде визуального дерева узлов и инспектора свойств.

Окно настроек постпроцессора:



Элементы управления

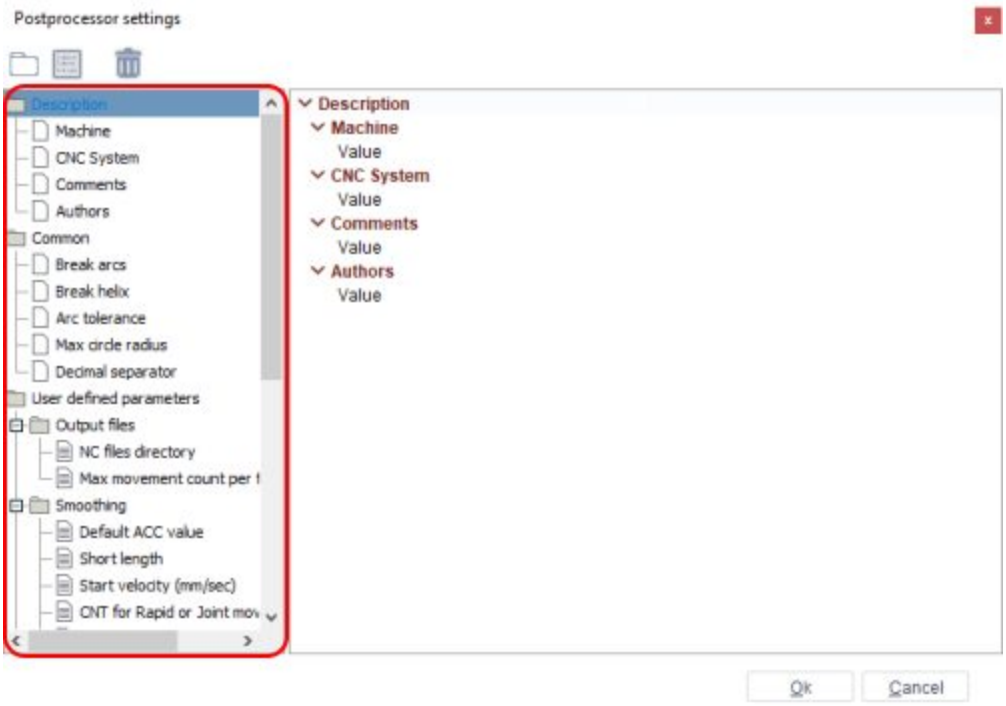
Панель управления элементами дерева находится в верхней левой части окна. На панели есть 3 элемента управления:



1. Add group — добавляет пользовательскую группу в выбранную папку раздела User defined parameters и в его корень;
2. Add parameter or enum value — добавляет пользовательский параметр в выбранную папку раздела User defined parameters и в его корень. Есть 7 вариантов добавления параметра:
    1. String parameter;
    2. Integer parameter;
    3. Double parameter;
    4. Boolean parameter;
    5. FileName parameter;
    6. FolderName parameter;
    7. Enum value — это значение добавляется только к выбранному параметру раздела User defined parameters.
3. Delete item — удаляет выбранный пользователем элемент в дереве элементов.

Элементы дерева

В левой части окна находится дерево с параметрами по умолчанию и пользовательскими параметрами.



Отображение дерева:

1. Секция/папка — ;
2. Параметр по умолчанию — ;
3. Пользовательский параметр — ;
4. Значение перечисления — ;

Параметры по умолчанию

Раздел Description и раздел Common являются заданными по умолчанию, и параметры внутри них нельзя редактировать, добавлять или удалять. Разрешено только вводить значение параметра. Раздел Description хранит строковые параметры для описания постпроцессора. Он имеет 4 параметра:

1. Machine;
2. CNC System;
3. Comments;
4. Authors.

Раздел Common хранит параметры для задания настроек постпроцессора по умолчанию. Он имеет 5 параметров:

1. Break arcs — позволяет задать режим разбиения дуг. Он имеет следующие значения:
    1. No — без разбиения;
    2. Cuts — с разбиением;
    3. Halves — разбиение дуги пополам;
    4. Quarters — разбиение дуги на четверти;

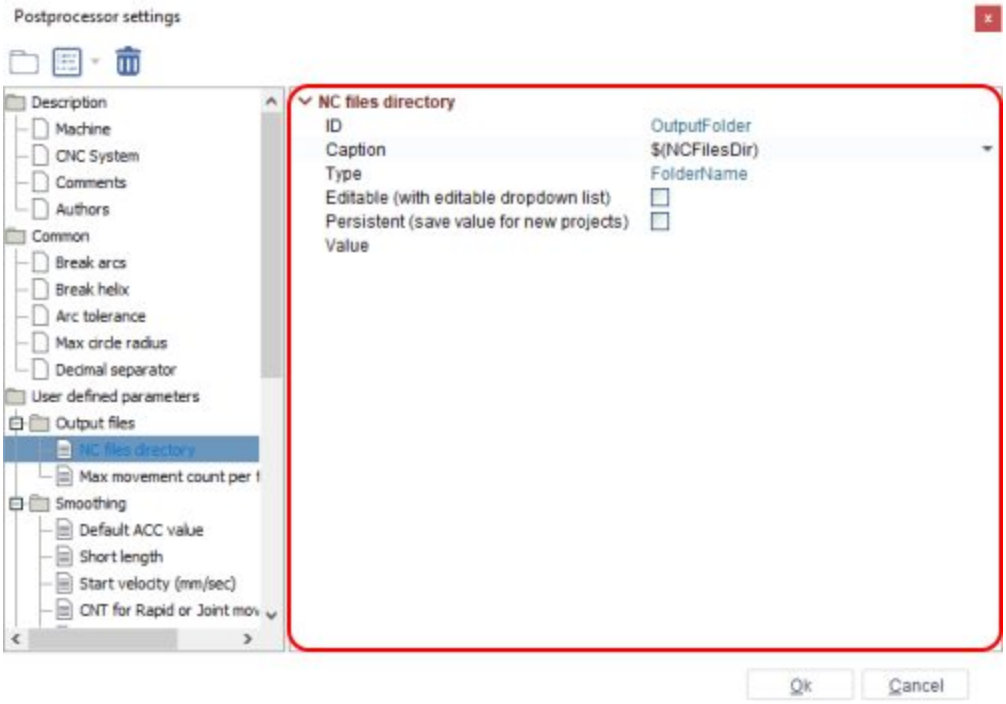
2. Break helix — позволяет включить или выключить режим разбиения спирали;
3. Arc tolerance — задание допуска дуги в виде значения double;
4. Max circle radius — задание максимального радиуса окружности в виде значения double;
5. Decimal separator — доступные значения: "." и ",".

Пользовательские параметры

Пользователь может создавать собственные параметры и хранить их в собственных папках в разделе User defined parameters. Папки можно хранить в других папках. Для перемещения параметров и папок можно использовать drag-n-drop, но Enum value можно перемещать только в пределах текущего параметра.

Панель атрибутов элементов

Панель атрибутов элементов находится в правой части окна.



В случае параметров по умолчанию на этой панели можно только вводить их значения. В случае пользовательских параметров можно выполнять полную настройку.

Каждый пользовательский элемент имеет атрибут Caption. Вы можете задать собственный заголовок, но есть несколько зарезервированных переменных для указания заголовка:

1. $(OutputFiles);
2. $(NCFileExt);
3. $(NCFileName);
4. $(NCFilesDir);

5. $(NCProgNumber). Эти переменные позволяют сделать переводимое имя (локализация). Также они позволяют генерировать некоторые значения по умолчанию в зависимости от смысла параметра, например файлы УП по умолчанию хранятся в папке, указанной в настройках CAM-системы.

Пользовательский элемент папки имеет атрибут ID — уникальный идентификатор элемента.

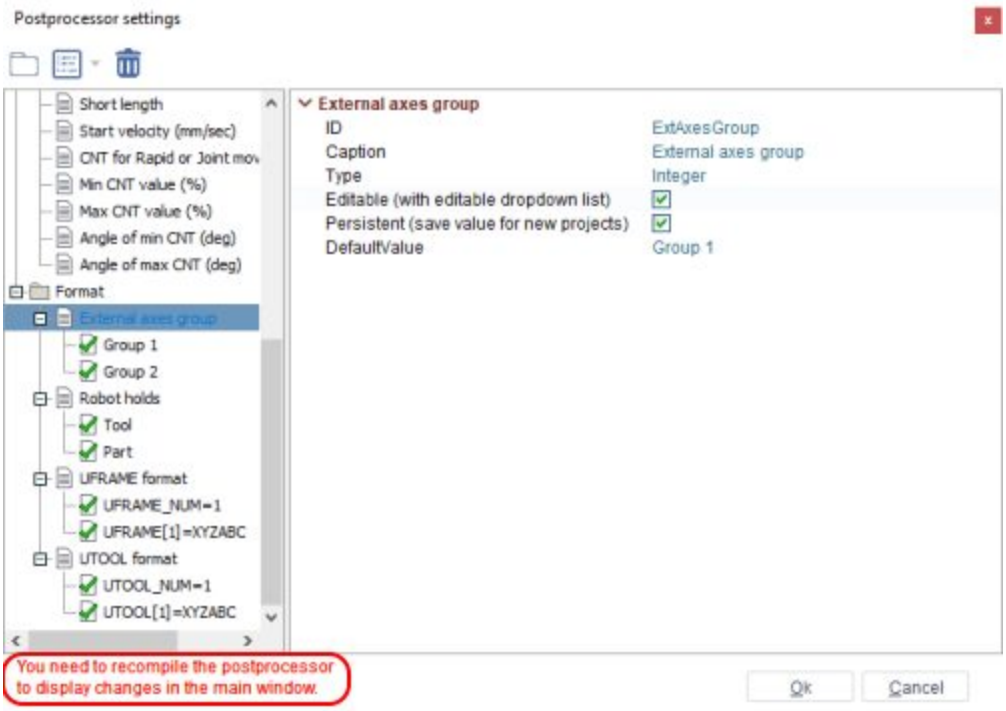
Пользовательский элемент параметра имеет следующие атрибуты:

1. ID — уникальный идентификатор элемента;
2. Type — тип параметра. Есть несколько значений:
    1. String — параметр со строковым значением;
    2. Integer — параметр с целым значением;
    3. Double — параметр со значением double;
    4. Boolean — параметр с булевым значением, флажок;
    5. FileName — параметр с диалоговым окном для поиска и указания пути файлов в значении параметра;
    6. FolderName — параметр с диалоговым окном для поиска и указания пути папки в значении параметра.
3. FilesFilter — расширение файла для поиска и открытия в диалоговом окне. Только для Type=FileName;
4. Editable (with editable dropdown list) — позволяет редактировать существующее значение перечисления и добавлять новое значение;
5. Persistent — сохраняет значение для новых проектов;
6. Value — значение параметра в соответствии с его типом;
7. DefaultValue — значение по умолчанию для параметра, имеющего перечисляемые значения, выпадающий список.

Элемент значения перечисления имеет атрибут Value. Тип значения такой же, как у родительского параметра.

Сохранение изменений

После каждого изменения и перемещения параметра в нижней левой части окна появится предупреждение: "You need to recompile the postprocessor to display changes in the main window".



Чтобы изменения были сохранены в xml-файле, нужно нажать кнопку Ok.

Главное окно CLData Viewer показывает настройки, встроенные в dll-файл постпроцессора, а не исходный xml-файл. Поэтому, чтобы увидеть результат редактирования в главном окне, следует пересобрать постпроцессор в Visual Studio Code.

Иногда также полезно сбросить настройки в состояние по умолчанию, потому что CLData Viewer также хранит значения, которые вы задаёте в главном окне для целей отладки, в отдельном временном файле. Для этого просто используйте пункт всплывающего меню Reset postprocessor settings to default постпроцессора или инспектора свойств.

Доступ к настройкам из кода C#

Ниже приведён пример того, как получить доступ к настройкам из C#-кода постпроцессора, используя свойства Settings и Settings.Params класса Postprocessor:

```
void StartNewFile() {
    ls = new LSFile();
    ls.Name = Settings.Params.Str["OutFiles.LSFileName"];
    ls.OutputFileName = Settings.Params.Str["OutFiles.OutputFolder"] + @"\" +
ls.Name + ".ls";

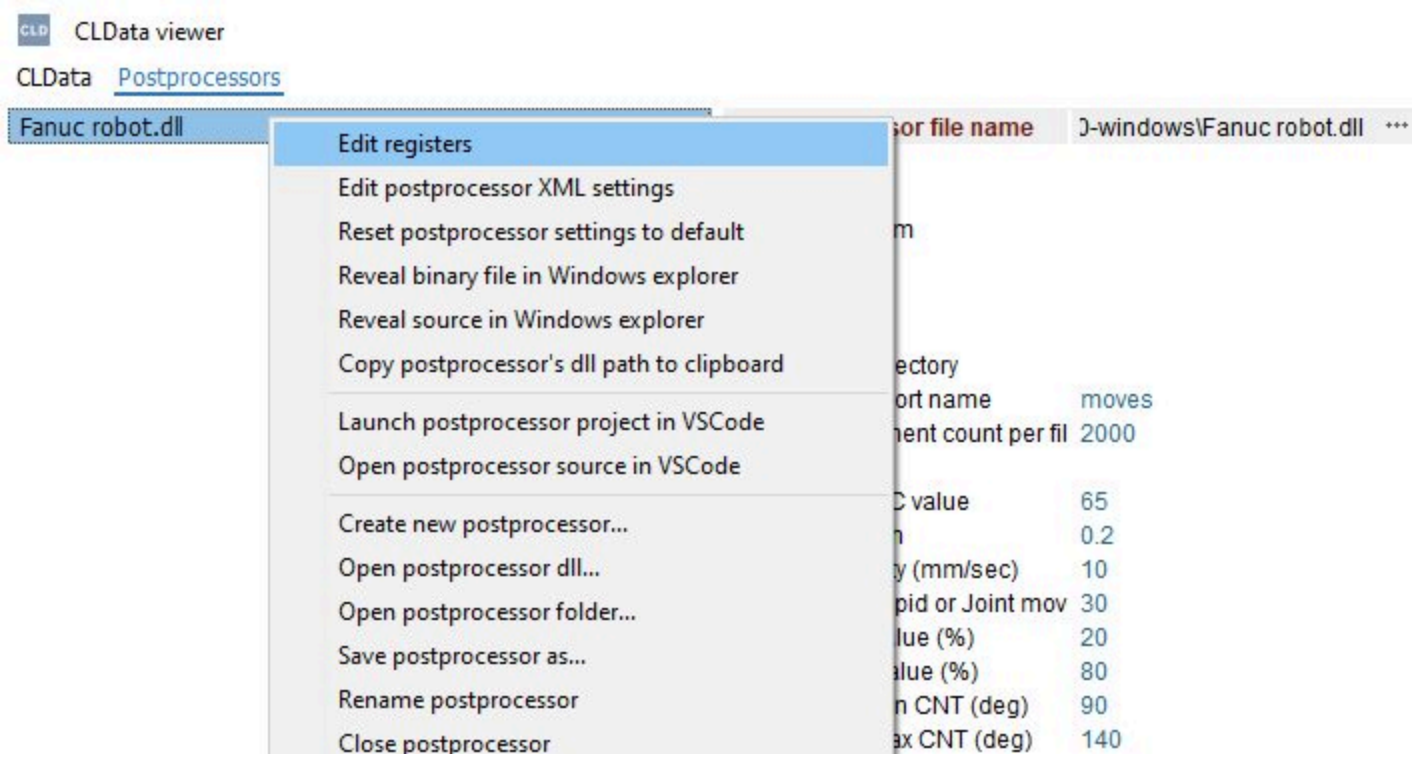
    unitsAreInches = prj.Int["Units"]>0;
    MaxMoveCount = Settings.Params.Int["OutFiles.MaxMoveCount"];

    smooth.accDefaultValue = Settings.Params.Int["Smoothing.DefaultACC"];
    smooth.ShortLength = Settings.Params.Flt["Smoothing.ShortLength"];
```

```
    smooth.StartVelocity = 60*Settings.Params.Flt["Smoothing.StartVelocity"];
    smooth.cntValueRapid = Settings.Params.Int["Smoothing.CntValueRapid"];
    smooth.cntValueMin = Settings.Params.Int["Smoothing.CntValueMin"];
    smooth.cntValueMax = Settings.Params.Int["Smoothing.CntValueMax"];
    smooth.angMin = Settings.Params.Int["Smoothing.AngMin"];
    smooth.angMax = Settings.Params.Int["Smoothing.AngMax"];
}
```

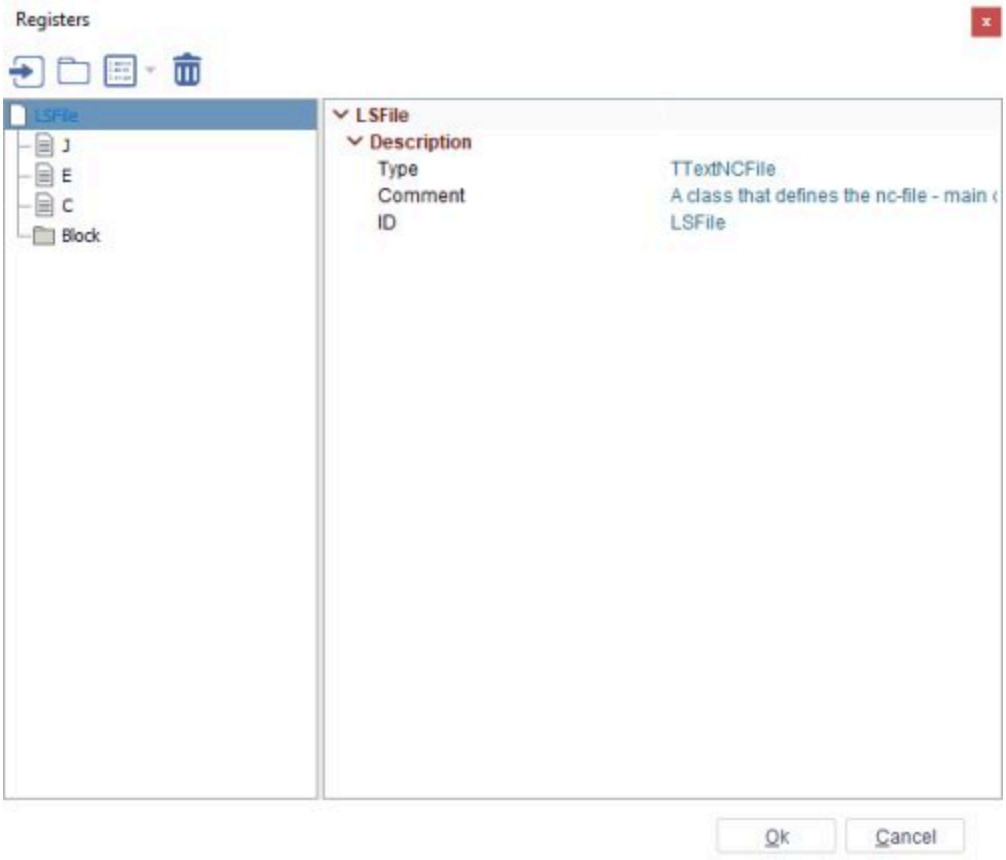
Окно регистров

Окно регистров позволяет работать с регистрами. Чтобы открыть окно, нужно выбрать постпроцессор в CLData Viewer и щёлкнуть по нему правой кнопкой мыши. После этого нажмите Edit registers в контекстном меню.



Окно регистров отображает регистры, блоки и файлы, которые хранятся в файле Registers.cs постпроцессора. Окно позволяет более удобно редактировать регистры, блоки и файлы.

Окно регистров:



Файл Registers.cs:

```
// Этот файл может генерироваться автоматически, не размещайте здесь значимый код.
// Используйте его только для определения списка nc-слов (регистров), которые могут
// появляться в блоках nc-файла.
namespace DotnetPostprocessing.SDK
{
    ///<summary>Класс, определяющий nc-файл — основной выходной файл, который должен
генерироваться постпроцессором.</summary>
    public partial class LSFile: TTextNCFile
    {
        ///<summary>Блок nc-файла — это упорядоченный список nc-слов</summary>
        public NCBlock Block;

        ///<summary>Счётчик автоматических номеров блоков</summary>
        // public CountingNCWord BlockN = new CountingNCWord("N{######}", 1, 1, 1);

        ///<summary>Форматтер значения сустава</summary>
        public NumericNCWord J = new NumericNCWord("{-000.000}", 0);

        ///<summary>Форматтер значения внешней оси</summary>
        public NumericNCWord E = new NumericNCWord("{-####.000}", 0);

        ///<summary>Форматтер значений X, Y, Z, W, P, R</summary>
```

```
        public NumericNCWord C = new NumericNCWord("{-000!000}", 0);

        public LSFile(): base()
        {
            Block = new NCBlock(this);
            OnInit();
        }

    }
}
```

Элементы управления

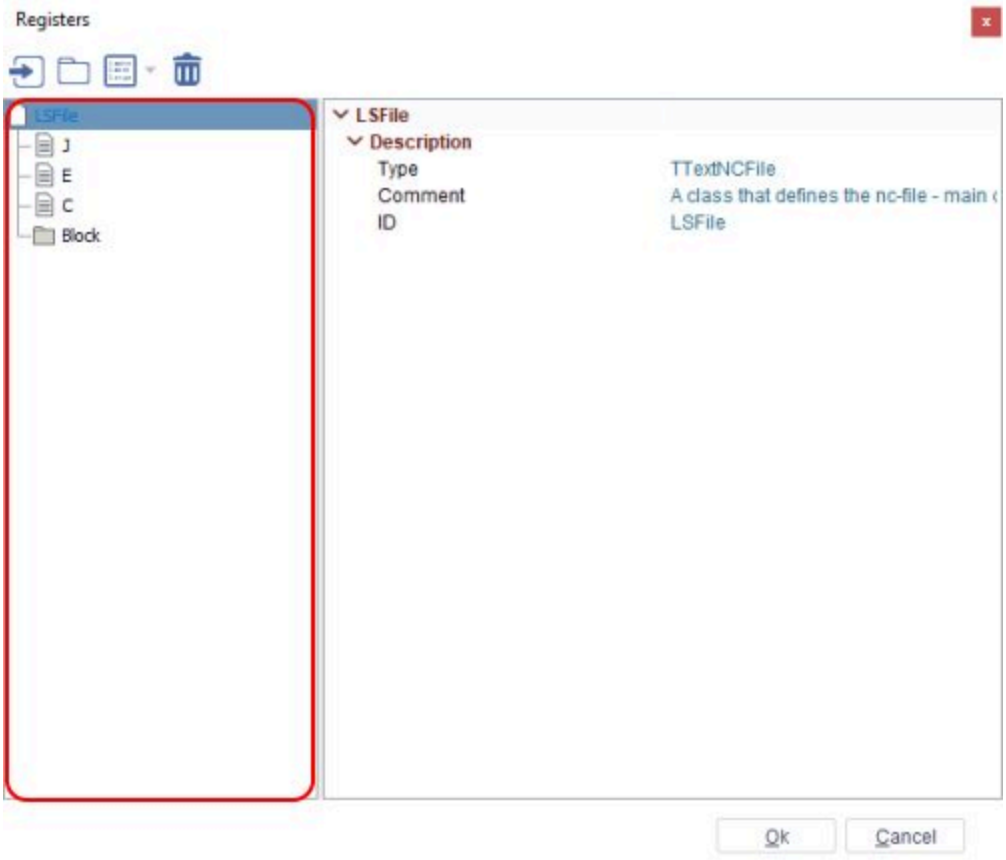
Панель управления элементами дерева находится в верхней левой части окна. На панели есть 4 элемента управления:



1. Add file — добавляет новый класс выходного файла;
2. Add block — добавляет новый блок, который может хранить регистры;
3. Add register — добавляет пользовательский регистр в выбранный блок или файл. Есть 4 варианта добавления регистра:
    1. Numeric register — регистр, хранящий числовое значение;
    2. Counting register — регистр, хранящий числовое значение, которое можно инкрементировать;
    3. Text register — регистр, хранящий строковое значение;
    4. Registers of current file. Этот вариант позволяет переместить существующий регистр в выбранный элемент (файл или блок). Если регистр перемещается в файл, то он удаляется из блока, в котором находился. Если регистр перемещается в блок, то он удаляется из файла, в котором находился. Но если он находился в другом блоке, то регистр перемещается в выбранный блок без удаления.
4. Delete item — удаляет выбранный элемент в дереве элементов.

Элементы дерева

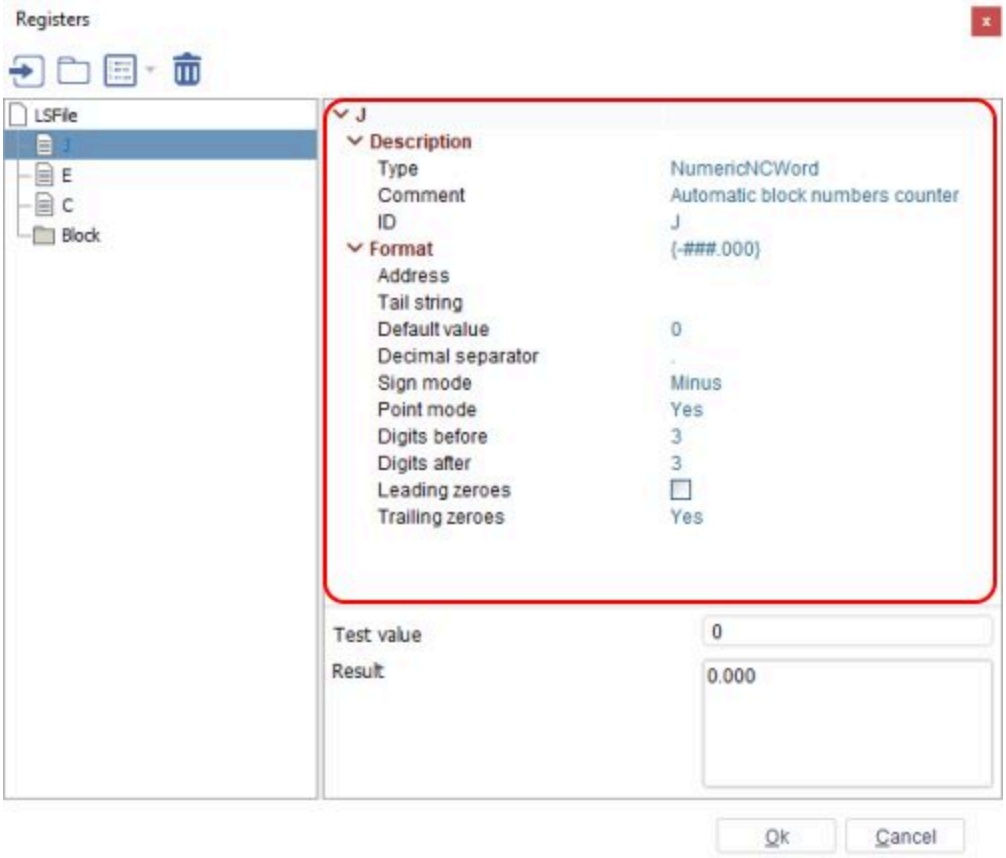
В левой части окна находится дерево со списком регистров , блоков и файлов .



Регистры могут располагаться в файле или в блоке. Можно перемещать регистры между блоками в файле. Регистр перемещается с помощью Drag'n'Drop. Регистр может располагаться либо только в корне файла, либо в одном или нескольких блоках данного файла. Чтобы скопировать регистр из одного блока в другой, нужно перемещать его с нажатой кнопкой Ctrl. Каждый элемент дерева имеет собственные параметры.

Панель параметров элементов

Панель параметров элементов находится в правой части окна.



Для каждого типа элемента существуют разные параметры.

Параметры элемента файла

Элемент файла имеет следующие параметры:

1. Type — тип класса выходного файла:
    1. TTextNCFile — основной класс для записи текстовых файлов;
    2. TSimpleTextNCFile — простейшая реализация выходного файла;
2. Comment — подсказка-комментарий для файла;
3. ID — уникальное имя класса файла.

Параметры элемента блока

Элемент блока имеет следующие параметры:

1. Comment — подсказка-комментарий для блока;
2. ID — уникальное имя класса блока.

Параметры элемента регистра

Элемент регистра имеет следующие параметры:

1. Type — тип класса регистра:

    1. CountingNCWord — это NumericNCWord, который автоматически увеличивает или уменьшает значение регистра на заданный шаг после каждого вызова Out() или Form();
    2. NumericNCWord — тип, преобразующий числовые значения в текстовый формат;
    3. TextNCWord — тип, преобразующий строковые значения в текстовый формат;
2. Comment — подсказка-комментарий для регистра;
3. ID — уникальное имя класса регистра; Элемент регистра имеет дополнительную секцию. Это формат. Вы можете ввести выходной формат в правой части строки формата, используя зарезервированные символы. После ввода параметров параметры внутри секции формата изменяются в соответствии с введённой строкой формата. И наоборот, при изменении параметров внутри секции формата строка формата изменяется. Каждый тип класса регистра имеет собственные параметры в секции формата.

Параметры формата регистра NumericNCWord

Регистр NumericNCWord имеет собственные параметры:

1. Address — символы, которые будут записаны в кадр УП перед значением регистра;
2. Tail string — символы, которые будут записаны в кадр УП после значения регистра;
3. Default value — значение по умолчанию текущего регистра;
4. Decimal separator — использование "." или "," в качестве десятичного разделителя;
5. Sign mode — определяет режим вывода знака значения регистра. Доступны следующие варианты:
    1. No — знак не выводится;
    2. Minus — выводится только минус;
    3. Plus and minus — выводятся плюс и минус;
6. Point mode — вывод десятичного разделителя. Это поле может иметь следующие значения:
    1. No — десятичный разделитель не выводится;
    2. Yes — десятичный разделитель будет выводиться, если значение регистра имеет дробную часть;
    3. Always — десятичный разделитель будет выводиться в любом случае;
7. Digits before — максимальное количество знаков в целой части числа;
8. Digits after — максимальное количество знаков в дробной части числа;
9. Leading zeroes — определяет режим вывода нулей перед значением регистра;
10. Trailing zeroes — определяет режим вывода нулей после значения регистра.

Параметры формата регистра CountingNCWord

Регистр CountingNCWord имеет те же параметры, что и регистр NumericNCWord, но имеет собственные дополнительные параметры:

1. Auto increment step — число, на которое будет увеличиваться выходное значение регистра;
2. Frequence — вывод значения регистра с заданной частотой. Например, выводить каждое третье значение (Frequence = 3);

3. Preserve indent — выравнивает все числовые значения по значению с наибольшим количеством символов.

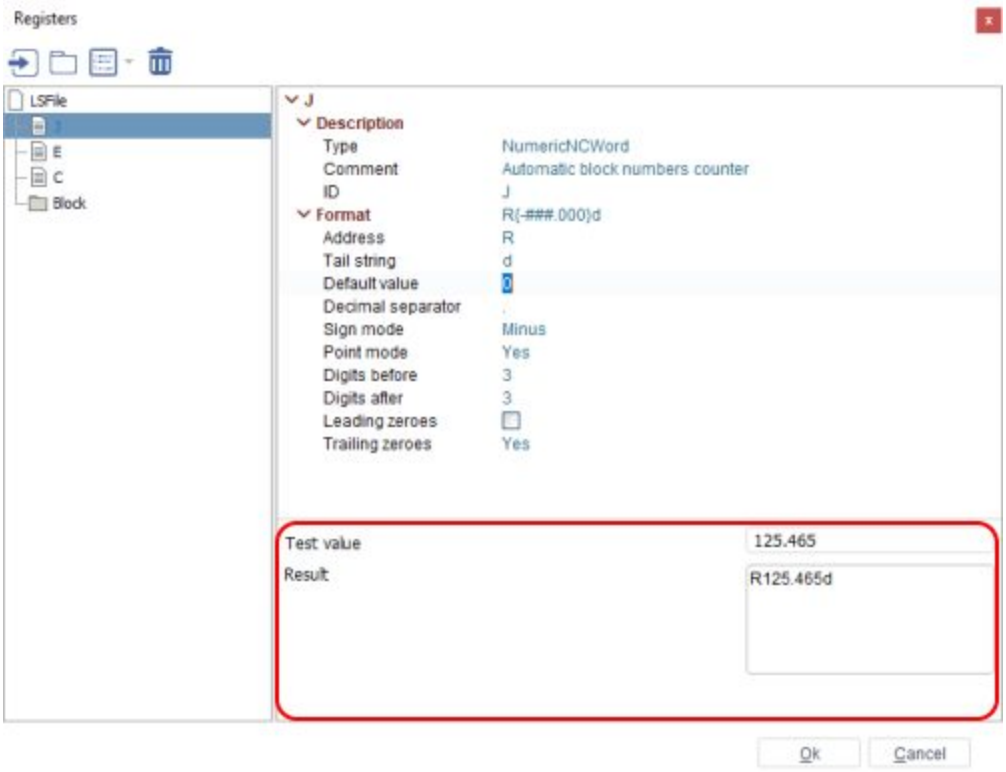
Параметры формата регистра TextNCWord

Регистр TextNCWord имеет собственные параметры:

1. Address — символы, которые будут записаны в кадр УП перед значением регистра;
2. Tail string — символы, которые будут записаны в кадр УП после значения регистра;
3. Default value — значение по умолчанию текущего регистра;
4. Text case — регистр букв значения. Этот параметр имеет три значения:
    1. Original — вывод значения в исходном виде;
    2. Lower — вывод значения в нижнем регистре;
    3. Upper — вывод значения в верхнем регистре.

Панель тестирования

Панель тестирования расположена под панелью параметров элементов. Панель тестирования отображается только для элементов регистров.

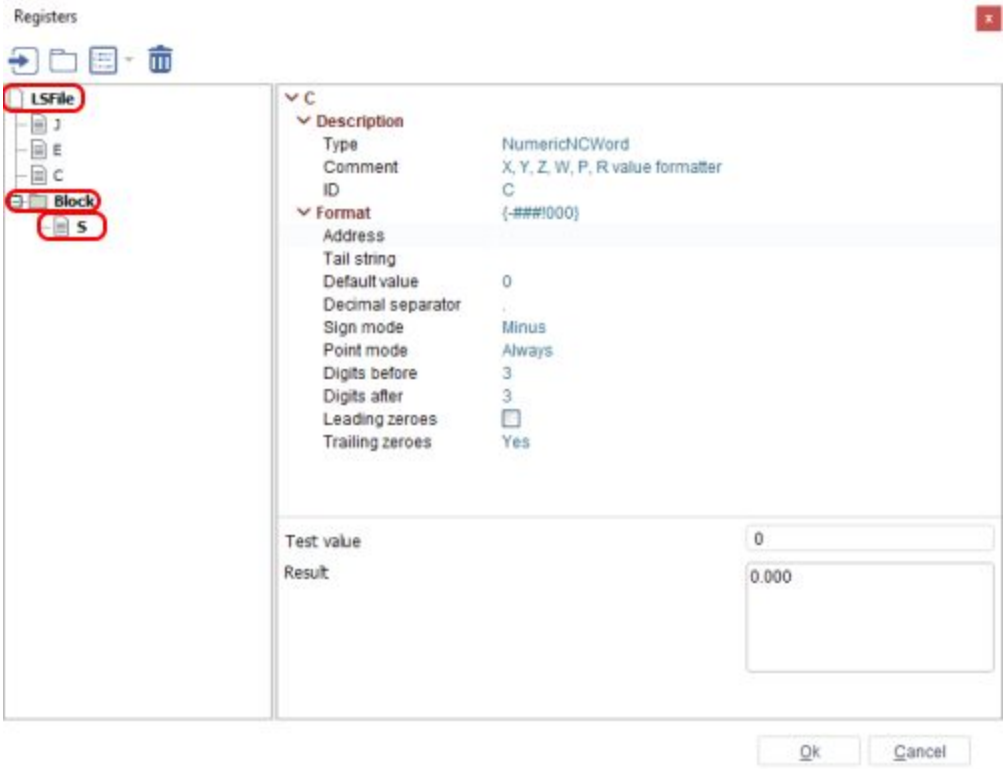


На этой панели есть 2 поля:

1. Test value — значение, которое будет выведено на основе заданного формата;
2. Result — пример отображения введённого значения на основе заданного формата. Для регистра отображается пять значений, так как этот регистр инкрементируется с каждым значением.

Сохранение изменений

Каждое изменение, добавление, удаление и перемещение элементов отображается на панели. Этот элемент и его родители будут выделены полужирным.



Чтобы сохранить все введённые изменения в окне регистров, нужно нажать кнопку Ok. После этого файл будет пересобран, и все данные из окна будут внесены в этот файл.

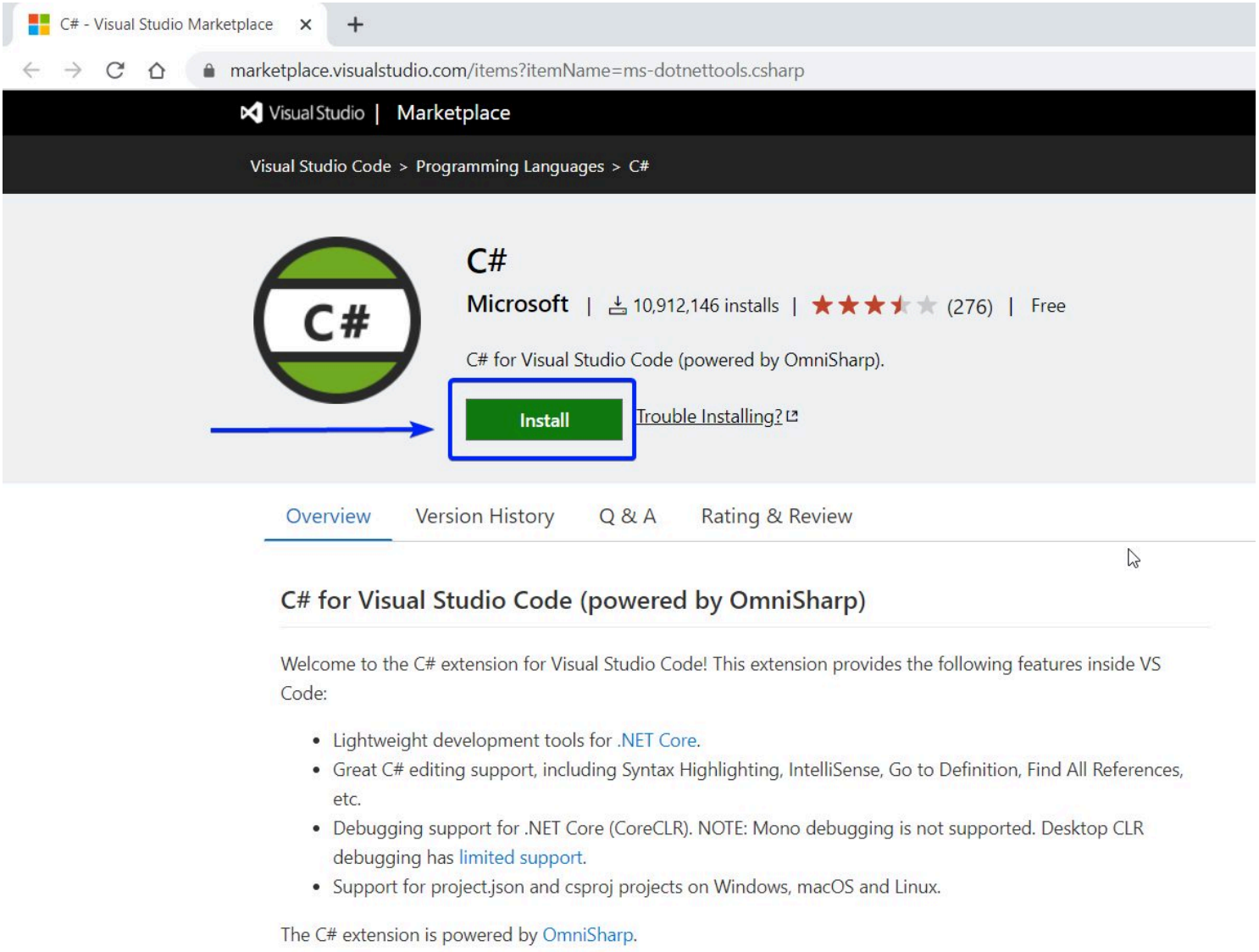
```
// Этот файл может генерироваться автоматически, не размещайте здесь значимый код.
// Используйте его только для определения списка nc-слов (регистров), которые могут
// появляться в блоках nc-файла.
namespace DotnetPostprocessing.SDK
{
    ///<summary>Класс, определяющий nc-файл — основной выходной файл, который должен
генерироваться постпроцессором.</summary>
    public partial class LSFile : TTextNCFile
    {
        ///<summary>Блок nc-файла — это упорядоченный список nc-слов</summary>
        public NCBlock Block;
        ///<summary></summary>
        public NumericNCWord S = new NumericNCWord("S{-###.###}", 0);
        ///<summary>Счётчик автоматических номеров блоков</summary>
        public NumericNCWord J = new NumericNCWord("{-###.000}", 0);
        ///<summary>Форматтер значения внешней оси</summary>
        public NumericNCWord E = new NumericNCWord("{-####.000}", 0);
        ///<summary>Форматтер значений X, Y, Z, W, P, R</summary>
```

```
        public NumericNCWord C = new NumericNCWord("{-###!000}", 0);
        public LSFile(): base()
        {
            Block = new NCBlock(
                  this,
                  S);
            OnInit();
        }
    }
}
```

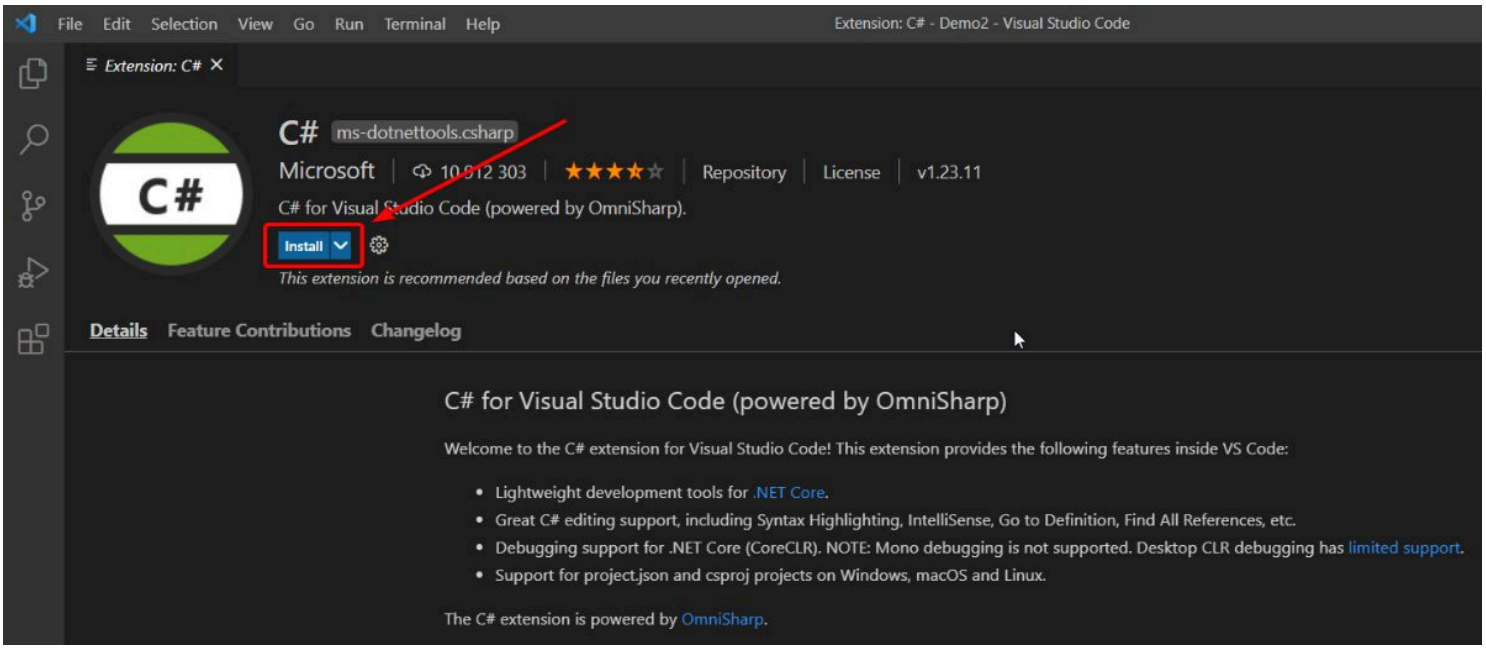
Подготовка компьютера к разработке .NET-постпроцессоров

Чтобы иметь возможность собирать двоичный файл *.dll постпроцессора из его исходных файлов (Postprocessor.cs, Registers.cs, *.csproj и т. д.), следуйте инструкциям ниже.

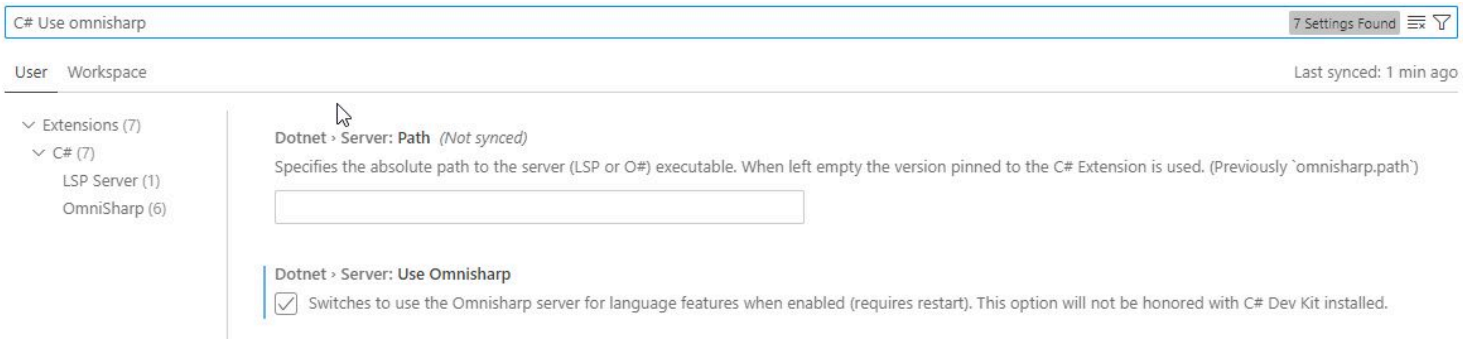
1. Скачайте и установите .NET SDK 8.0 x64 с официального сайта (https://dotnet.microsoft.com/download, размер ~200 Мб). Там есть простое пошаговое руководство.
2. Чтобы иметь возможность писать исходные файлы, скачайте и установите последнюю доступную версию Visual Studio Code (https://code.visualstudio.com/download, размер ~100 Мб).
3. Установите расширение C# из магазина Visual Studio Code.



4. Это запустит экземпляр Visual Studio Code (сокращённо VSCode) на вкладке Extensions. Просто нажмите кнопку "Install" рядом с именем расширения.

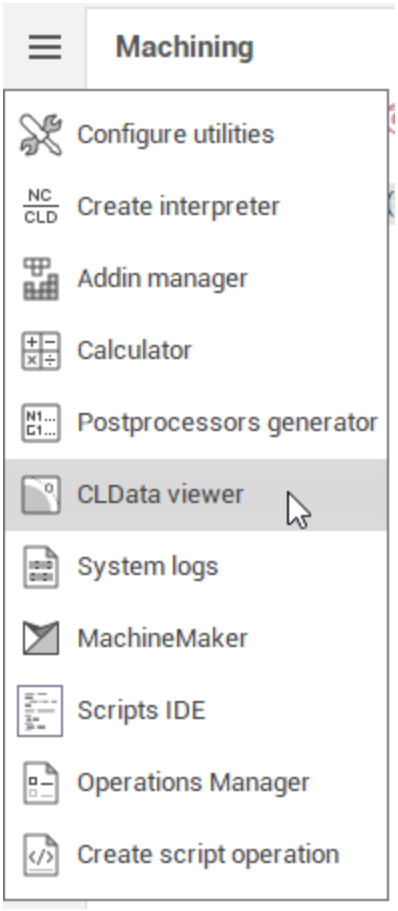


В последних версиях VS Code при установке расширения для `C#` предлагается сразу установить "C# Dev Kit". Однако по условиям использования этот набор требует лицензии Visual Studio. Использование расширения "C# Dev Kit" не обязательно для разработки постпроцессоров; всё прекрасно работает и без него, и оно не требует покупки Visual Studio.

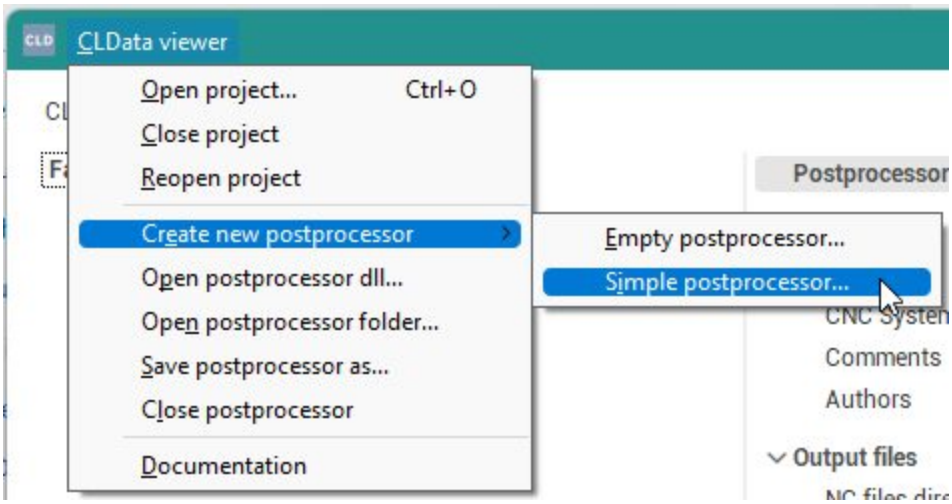


Иногда версии расширения `C#` на базе Omnisharp работают лучше, поэтому можно попробовать включить настройку "C#: Use Omnisharp", которую можно найти, нажав кнопку с шестерёнкой рядом с кнопкой установки расширения.

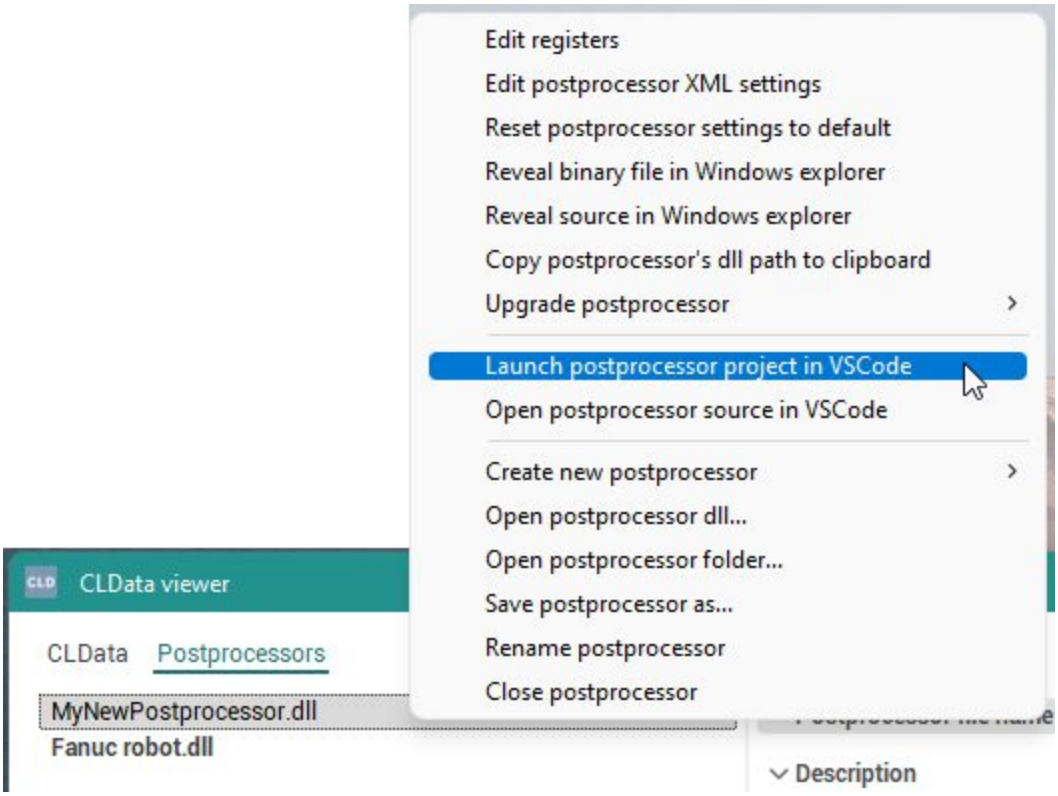
5. Рекомендуется (но не обязательно) установить расширение для редактирования xml-файлов "XML Language Support by Red Hat" (https://marketplace.visualstudio.com/items?itemName=redhat.vscode-xml).
6. Если вы ещё этого не сделали, скачайте и установите последнюю версию CAM-системы с официального ресурса.
7. Запустите CAM-систему и запустите CLData Viewer из меню Utilities.



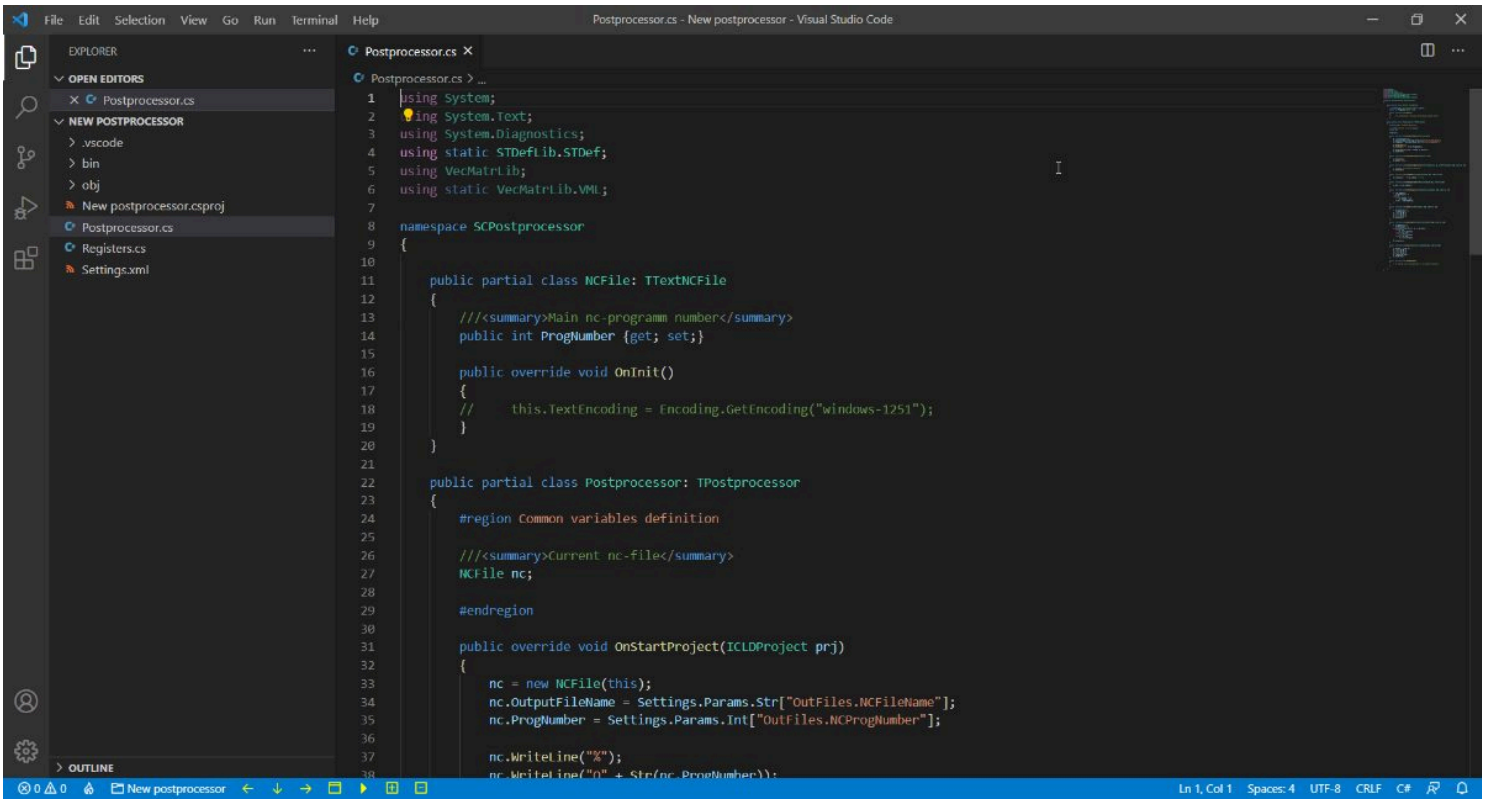
8. Это добавит переменную окружения "InpCoreDir" к переменным Windows. Чтобы иметь возможность запускать Visual Studio Code из CAM-системы, системная переменная окружения "Path" должна включать папку установки VS Code. Обычно установщик VSCode добавляет всё сам, и вам нужно лишь перезагрузить компьютер, чтобы переменные окружения вступили в силу. Мы рекомендуем перезагрузить компьютер после первого запуска CLData Viewer, чтобы переменные окружения вступили в силу.
9. Запустите CAM-систему и снова запустите CLData Viewer. Теперь можно попробовать создать новый постпроцессор. Нажмите в главном меню CLData viewer/Create new postprocessor/Simple postprocessor…



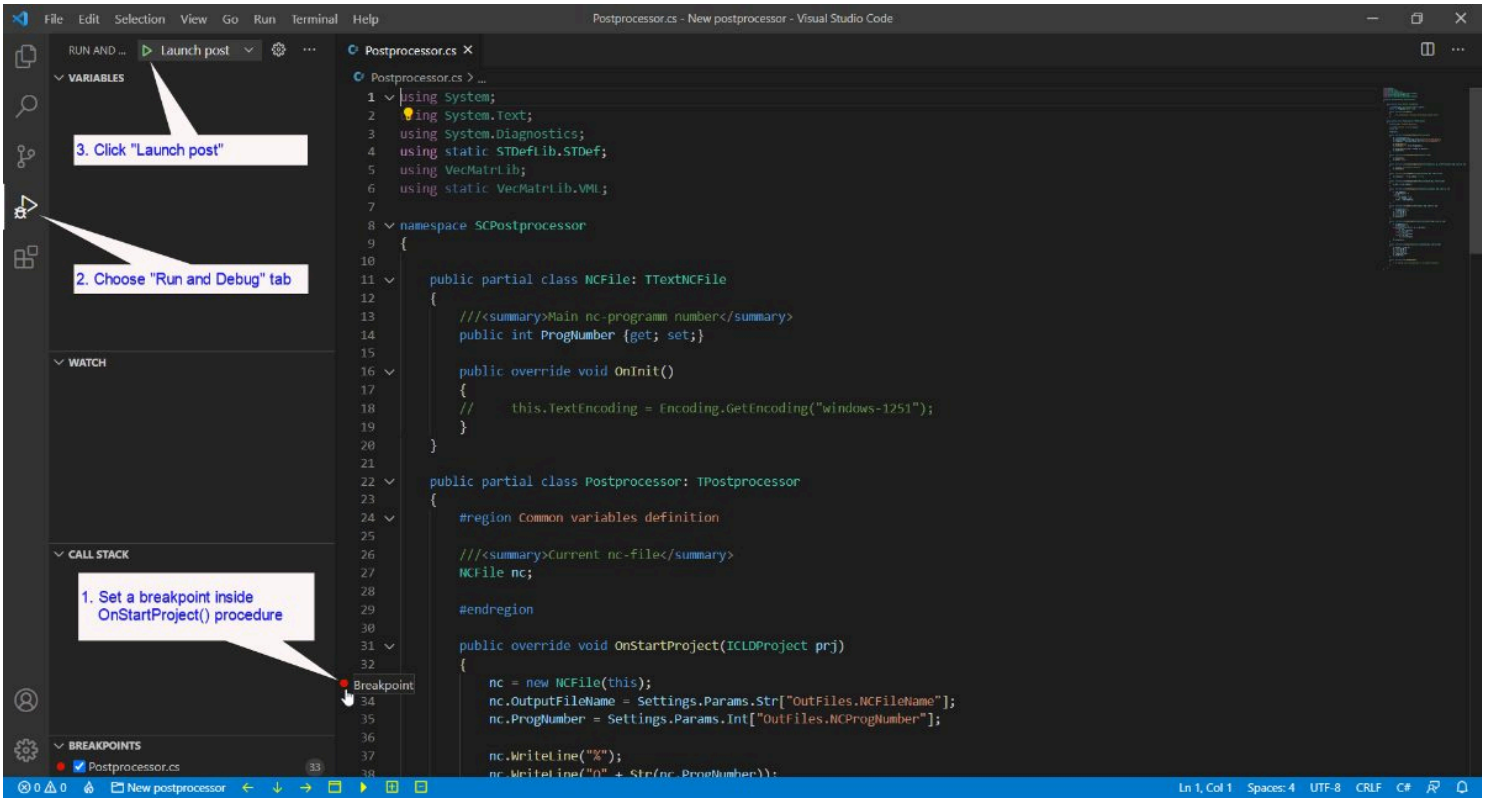
10. Откроется стандартное диалоговое окно выбора папки. Здесь нужно создать новую пустую папку, которая будет содержать все файлы постпроцессора. Имя папки будет совпадать с именем нового постпроцессора.
11. В контекстном меню постпроцессора выберите пункт "Launch postprocessor project in VS Code" или просто дважды щёлкните по постпроцессору в списке.



12. Появится новое окно VSCode с главным исходным файлом Postprocessor.cs выбранного постпроцессора. При первом открытии постпроцессора расширению Omnisharp (C#) может понадобиться дополнительное время, чтобы автоматически скачать последние обновления и зависимости.

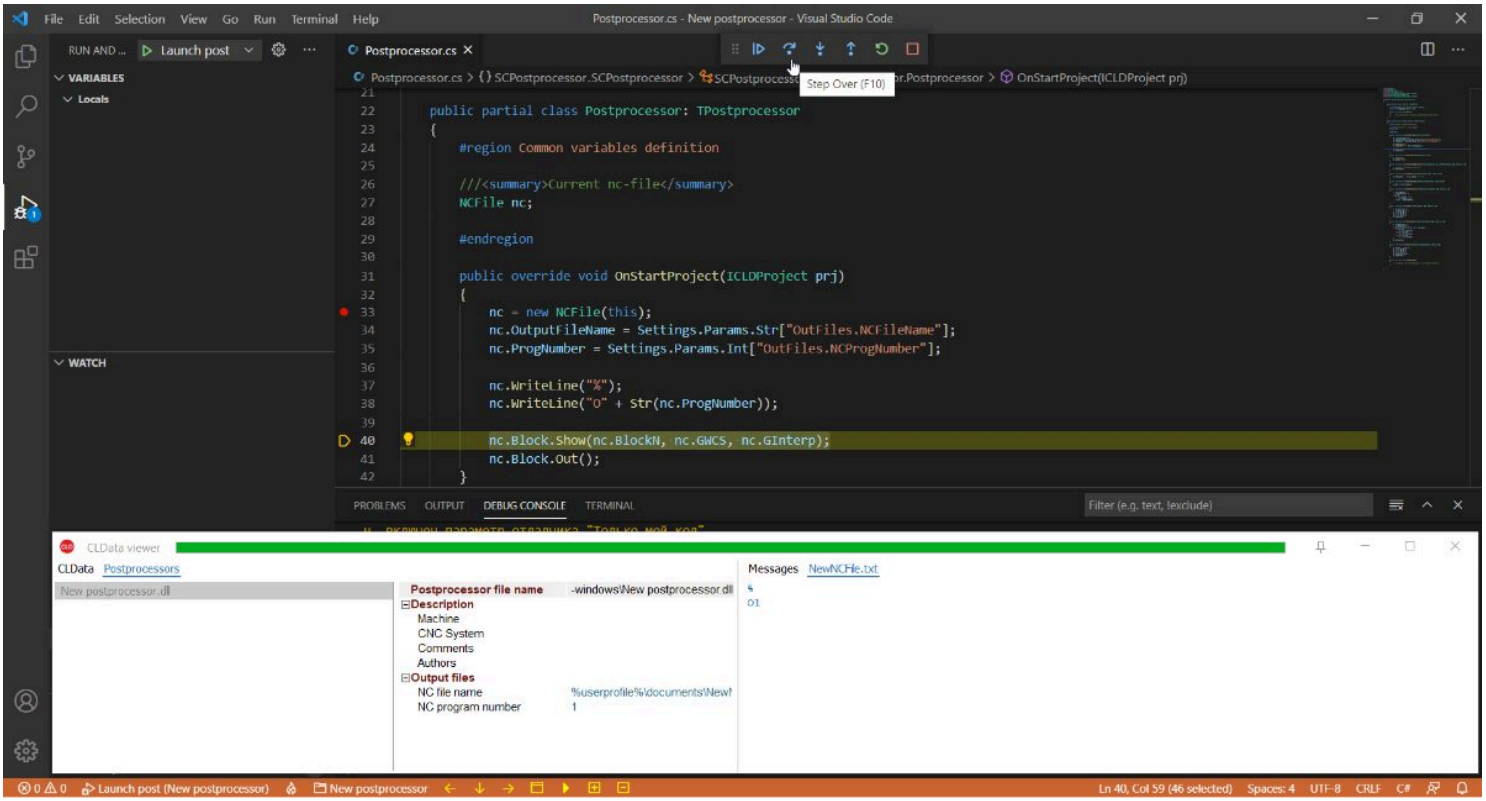


13. Затем можно установить точку останова внутри процедуры OnStartProject, выбрать вкладку "Run and Debug" и нажать "Launch post" (клавиша F5).

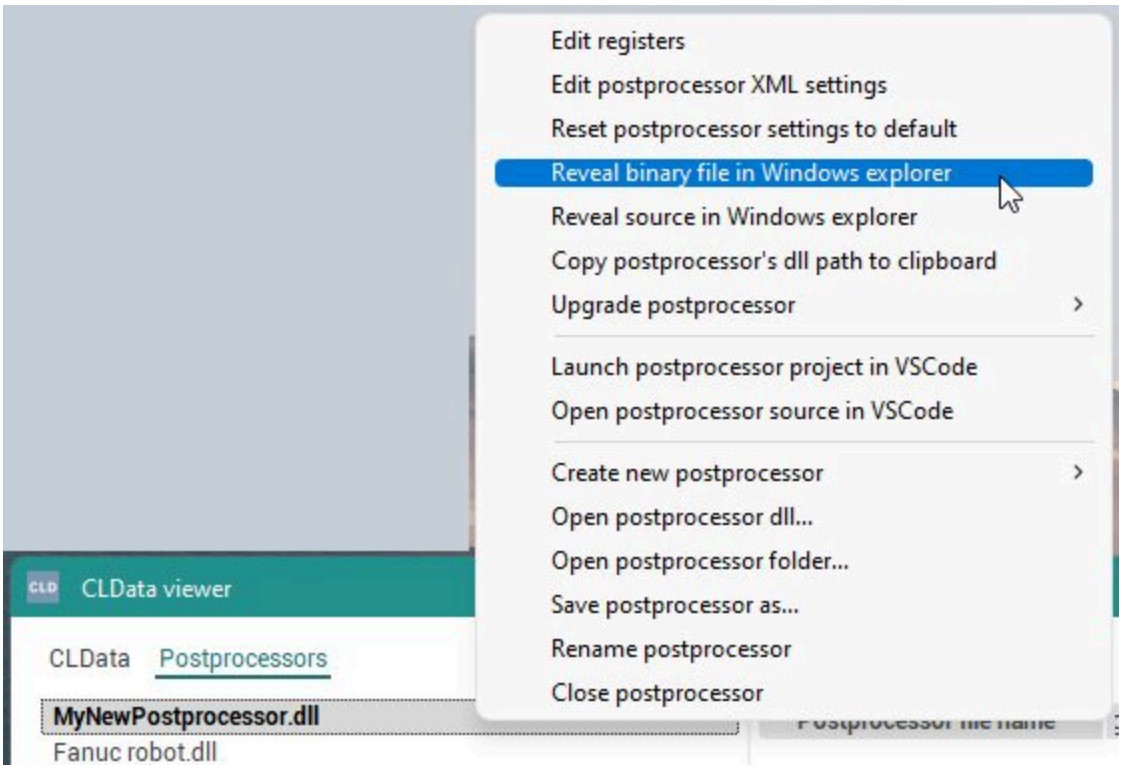


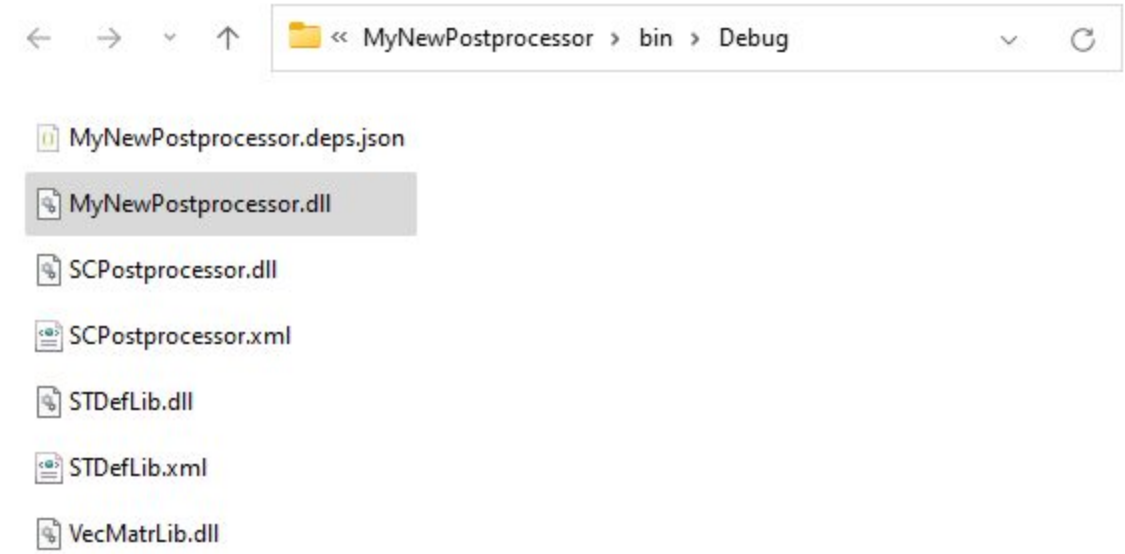
14. Это соберёт постпроцессор из исходников и запустит отладку. Должно появиться окно CLData Viewer (если ранее оно было свёрнуто). Выполнение должно остановиться на заданной точке останова. Теперь можно использовать Step Over (F10) на панели

инструментов, чтобы пройти по строкам кода постпроцессора. По мере записи строк в выходной файл CLData Viewer будет показывать промежуточные результаты.



15. Наконец, вы можете изменять код постпроцессора в файлах Postprocessor.cs, Registers.cs и Settings.xml по своему усмотрению и отлаживать его, используя все доступные возможности Visual Studio Code. Если вы ещё не знакомы с возможностями VSCode, рекомендуем посетить раздел "Get started" его официальной документации (https://code.visualstudio.com/docs/getstarted/introvideos).
16. После того как вы написали и отладили исходный код постпроцессора, можно получить и использовать двоичный выходной файл *.dll постпроцессора. Просто выберите "Reveal binary file in Windows explorer" в контекстном меню постпроцессора в CLData Viewer. Это выделит двоичный файл постпроцессора в проводнике, откуда его можно скопировать или отправить заказчику.


Edit registers
Edit postprocessor XML settings
Reset postprocessor settings to default
Reveal binary file in Windows explorer
Reveal source in Windows explorer
Copy postprocessor's dll path to clipboard
Upgrade postprocessor
Launch postprocessor project in VSCode
Open postprocessor source in VSCode
Create new postprocessor
Open postprocessor dll...
Open postprocessor folder...
Save postprocessor as...
Rename postprocessor
Close postprocessor
CLData viewer
CLData
Postprocessors
MyNewPostprocessor.dll
Fanuc robot.dll


MyNewPostprocessor › bin › Debug
MyNewPostprocessor.deps.json
MyNewPostprocessor.dll
SCPostprocessor.dll
SCPostprocessor.xml
STDefLib.dll
STDefLib.xml
VecMatrLib.dll

CLData

CLData — это промежуточные данные, которые CAM-система передаёт постпроцессору: упорядоченный список технологических команд (перемещения инструмента, управление шпинделем и СОЖ, смены инструмента, стандартные циклы, вспомогательные функции и т. д.), каждая из которых несёт набор параметров. Постпроцессор читает CLData и формирует управляющую программу для конкретного станка и системы ЧПУ.

Обе подсистемы постпроцессирования — Генератор постпроцессоров (sppx-постпроцессоры) и .NET-постпроцессоры — работают с одними и теми же CLData. Модуль, который читает и пишет CLData, общий для обеих; низкоуровневое взаимодействие с нативным кодом скрыто, и каждая подсистема предоставляет доступ к данным через удобные обёртки. Между подсистемами различается только синтаксис доступа — сами команды и их параметры идентичны.

В этом разделе команды CLData описаны один раз, для обеих подсистем. Для каждой команды параметры сначала описываются в синтаксически нейтральном виде, затем отдельными абзацами показывается, как их читать из sppx и из .NET. Используйте форму доступа своей подсистемы и не смешивайте два синтаксиса.

Из чего состоит CLData

CLData организованы в виде небольшой иерархии — проект, его файлы, команды внутри каждого файла и параметры каждой команды:

- Проект — весь проект CAM-системы. Помимо операций он несёт данные уровня проекта — схему станка, детали и заготовки, единицы измерения, установочные позиции и т. д.
- Файлы — CLData проекта разбиты на файлы. На каждую операцию CAM приходится один файл, для каждой подпрограммы — отдельный необязательный файл, и один файл уровня проекта, который хранит информацию о проекте и не содержит команд.
- Команды — каждый файл представляет собой упорядоченную последовательность технологических команд: перемещения инструмента, управление шпинделем и СОЖ, смены инструмента, стандартные циклы, маркеры секций, вспомогательные функции и т. д. Порядок соответствует порядку выполнения операции.
- Параметры — каждая команда несёт набор описывающих её параметров: координаты перемещения, частоту вращения шпинделя, номер инструмента, глубины и смещения цикла и т. д. У каждого типа команды свой набор параметров.

Постпроцессор обходит эту иерархию — файл за файлом, команда за командой — и для каждой команды читает нужные ему параметры, чтобы сформировать соответствующий кадр управляющей программы. В следующем разделе приведён реальный листинг такого проекта,

а разделы после него объясняют, как добираться до файлов, команд и параметров из кода постпроцессора.

Структура листинга CLData

CLData проекта — это не один плоский список команд, а набор файлов, и внутри каждого файла команды сгруппированы во вложенные секции. Знание этой раскладки делает список команд читаемым и позволяет обработчику реагировать на структуру траектории. Приведённый ниже пример — реальный листинг простого токарно-фрезерного проекта, полученный небольшим постпроцессором, который обходит каждый файл и каждую команду (тот же приём обхода файлов, что и в примере из раздела Файлы CLData).

Файлы

- CLData проекта — это список файлов, нумерация начинается с `0`. Нумерация файлов 0-based в обеих подсистемах. А вот нумерация команд *внутри* файла различается: она 1-based в sppx (`for c = 1 to CldFile[f].CmdCount`) и 0-based в .NET (`for (int c = 0; c < file.CmdCount; c++)`).
- Файл **`0`** — всегда файл уровня проекта (тип `CAMProject`). Он не содержит команд — в нём хранится информация о проекте, читаемая оператором Project.
- Каждая операция CAM становится одним файлом типа `TechOperation`, в порядке списка операций.
- Файлы типа `NCSub` необязательны — они существуют, только если в проекте используются подпрограммы или циклы с подпрограммами. В отличие от файлов операций, они не транслируются автоматически; постпроцессор выводит их явным вызовом операторов работы с подпрограммами (см. Операторы работы с подпрограммами УП).

```
File 0   CAMProject      - данные уровня проекта, без команд
File 1   NCSub           - необязательно: тела подпрограмм / циклов
File 2   NCSub
File 3   TechOperation   - по одному файлу на операцию CAM, в порядке
списка операций
File 4   TechOperation
 ...
File 15  TechOperation
```

Секции внутри операции

- Каждый файл операции обёрнут парой `PPFUN TECHINFO(58)` … `PPFUN ENDTECHINFO(59)` — начало / конец операции, несущие технологические данные операции.
- Команды группируются во вложенные секции командой `STRUCTURE` (`ON(71)` открывает секцию, `OFF(72)` закрывает). Три секции верхнего уровня — Header, Body и Tail; Header содержит подсекцию Approach, а Tail — подсекцию Return. `STRUCTURE` также используется

для группировки рабочих перемещений по регионам / проходам инструмента — например, у фрезерной операции Body делится на группы по глубине Level, а каждый уровень — на проходы Offset (см. `File 15` ниже). В списке команд эти вложенные секции отображаются отступами.

- В Header находятся команды настройки, обычно в таком порядке:
  - первый `COMMENT` — имя операции CAM;
  - `ORIGIN` — базовая система координат, в которой выводится траектория операции;
  - `LOADTL` — инструмент, которым ведётся обработка в операции;
  - `COMMENT` сразу после инструмента, начинающийся с `@`, — имя инструмента;
  - `CUTCOM` — активирует коррекцию инструмента;
  - `PLANE` — активирует рабочую плоскость;
  - `SPINDL` — запускает шпиндель;
  - `FROM` — начальное положение инструмента; перемещения не выполняет, лишь обозначает, где инструмент уже находится к началу операции.
- Набор команд перемещения зависит от типа оборудования: `GOTO` / `CIRCLE` для 2D/3D токарно-фрезерных траекторий (токарные и сверлильные циклы дополнительно выводят `EXTCYCLE` и `MULTIGOTO` для позиционирования, как в примере ниже), `MULTIGOTO` / `MULTIARC` для пятиосевых станков и роботов, и `EDMMOVE` для электроэрозионных станков. В остальном структура остаётся в основном такой же.
- Ключевые точки траектории помечаются тегами — `PPRINT`, текст которого начинается с `#` (например, `#KeyPoint: StartCutting`, `#Approach:…`, `#Return:…`). Постпроцессор реагирует на тег или пропускает его, чтобы метка не попала в управляющую программу; см. Теги траектории. В листинге ниже такие строки помечены `! toolpath tag`.

Ниже — сквозной обход всего проекта: файл проекта, файлы подпрограмм и файлы операций; длинные списки параметров сокращены до …, повторяющиеся проходы свёрнуты; отступы показывают вложенность `STRUCTURE` (индексы команд опущены):

```
File 0 — CAMProject   (уровень проекта; команд нет)

File 1 — NCSub   (тело подпрограммы / цикла; выводится только при явном вызове
из постпроцессора)
COMMENT    "Geometry"
PPFUN      STARTSUB(50), 1
GOTO       X 67.8, Y 0, Z 0
PPRINT     "#KeyPoint: StartCutting"                    ! toolpath tag
FEDRAT     F 0.5, MMPR(316), K 0
GOTO       X 30.703, Y 0, Z 0
GOTO       X 33.362, Y 0, Z 2.659
PPFUN      ENDSUB(51), 1

File 2 — NCSub   (ещё одно тело подпрограммы; та же форма, что и File 1 — опущено)
```

```
File 3 — TechOperation "Lathe facing"   (точение)
STRUCTURE  ON(71)  TYPE "HeaderSection", Comment "$Header$"
  PPFUN      TECHINFO(58), 250, 30.703, 0, 0, 979.484, …    ! операция начинается
  STRUCTURE  ON(71)  TYPE "String", Comment "Header"
    COMMENT    "Lathe facing"                                ! имя операции
    ORIGIN     X 0, Y 0, Z 265, PPFUN 0, N 54, …             ! базовая система
координат (G54)
    LOADTL     N 2, ToolChanged 1, X 15, Y 125, …            ! инструмент
    COMMENT    "@IC16 Re0.2 R OD cutting tool"               ! имя инструмента
(начинается с @)
    CUTCOM     ON(71), LENGTH(9)2, …, LEFT(8)                ! коррекция инструмента
    FROM       COUNT 10, MACHINE, X 100, Y 0, Z 500, …       ! начальное положение
(без перемещения)
  STRUCTURE  OFF(72) TYPE "String", Comment "Header"
  STRUCTURE  ON(71)  TYPE "String", Comment "Approach"
    STRUCTURE  ON(71)  TYPE "Approach", Comment "Approach"
      PPRINT     "#Approach:[Rule 1]=C; Z10; X; Z"           ! toolpath tag
      SELWORKPIECE"Table"
      SPINDL     ON(71), NO -1000, K 0, MODE CSS(2), SPEED -150   ! шпиндель
      RAPID      N 10000
      MULTIGOTO  COUNT 1, MACHINE, Z 10, …
      MULTIGOTO  COUNT 1, MACHINE, X 67.8, …
      MULTIGOTO  COUNT 1, MACHINE, Z 2.659, …
    STRUCTURE  OFF(72) TYPE "Approach", Comment "Approach"
  STRUCTURE  OFF(72) TYPE "String", Comment "Approach"
STRUCTURE  OFF(72) TYPE "HeaderSection", Comment "$Header$"
STRUCTURE  ON(71)  TYPE "BodySection", Comment "$Body$"
  GOTO       X 67.8, Y 0, Z 2.659
  COOLNT     ON(71), N 1, #1
  FEDRAT     F 0.5, MMPR(316), K 0
  EXTCYCLE   ON(71)   LATHEFINISH(400), 1, 0, 3, 0, 0.3, …
  EXTCYCLE   CALL(52) LATHEFINISH(400), 1, 0, 3, 0, 0.3, …
  EXTCYCLE   OFF(72)  LATHEFINISH(400), 1, 0, 3, 0, 0.3, …
STRUCTURE  OFF(72) TYPE "BodySection", Comment "$Body$"
STRUCTURE  ON(71)  TYPE "TailSection", Comment "$Tail$"
  STRUCTURE  ON(71)  TYPE "String", Comment "Return"
    STRUCTURE  ON(71)  TYPE "Return", Comment "Return"
      PPRINT     "#Return:[Rule 1]=Z10; X; Z; C"             ! toolpath tag
      RAPID      N 10000
      MULTIGOTO  COUNT 1, MACHINE, Z 10, …
    STRUCTURE  OFF(72) TYPE "Return", Comment "Return"
  STRUCTURE  OFF(72) TYPE "String", Comment "Return"
  PPFUN      ENDTECHINFO(59), 250, 30.703, 0, 0, 979.484, … ! операция заканчивается
STRUCTURE  OFF(72) TYPE "TailSection", Comment "$Tail$"
```

```
…  (Files 4…14 — остальные операции, каждая той же формы Header / Body / Tail
— опущены)

File 15 — TechOperation "2D contouring"   (фрезерование)
STRUCTURE  ON(71)  TYPE "HeaderSection", Comment "$Header$"
  PPFUN      TECHINFO(58), 250, -47.996, -289.976, -16, …  ! операция начинается
  STRUCTURE  ON(71)  TYPE "String", Comment "Header"
    COMMENT    "2D contouring"                              ! имя операции
    ORIGIN     X 0, Y 0, Z 274, PPFUN 0, N 55, …            ! базовая система
координат (G55)
    LOADTL     N 1, ToolChanged 1, …, D 8, L 40, …          ! инструмент
    COMMENT    "@Cylindrical mill L40, D8"                  ! имя инструмента
(начинается с @)
    PLANE      XY                                           ! рабочая плоскость
    SPINDL     ON(71), NO 10544.01, K 0, MODE RPM(0)        ! шпиндель
    CUTCOM     ON(71), LENGTH(9)1, …, LEFT(8)               ! коррекция инструмента
    FROM       COUNT 10, MACHINE, X 100, Y 0, Z 500, …      ! начальное положение
(без перемещения)
  STRUCTURE  OFF(72) TYPE "String", Comment "Header"
  STRUCTURE  ON(71)  TYPE "String", Comment "Approach"
    STRUCTURE  ON(71)  TYPE "Approach", Comment "Approach"
      PPRINT     "#Approach:[Rule 1]=C; Z10; X; Z"          ! toolpath tag
      RAPID      N 10000
      AXESBRAKE  COUNT 1, AxisCPos(3) OFF(72)
      MULTIGOTO  COUNT 1, MACHINE, C 90, …
      MULTIGOTO  COUNT 1, MACHINE, Z 10, …
      MULTIGOTO  COUNT 1, MACHINE, X -28.715, …
      MULTIGOTO  COUNT 1, MACHINE, Z 0, …
      MULTIGOTO  COUNT 1, MACHINE, Y -1.804, …
    STRUCTURE  OFF(72) TYPE "Approach", Comment "Approach"
  STRUCTURE  OFF(72) TYPE "String", Comment "Approach"
STRUCTURE  OFF(72) TYPE "HeaderSection", Comment "$Header$"
STRUCTURE  ON(71)  TYPE "BodySection", Comment "$Body$"
  STRUCTURE  ON(71)  TYPE "String", Comment "Level: -4"      ! первый уровень
по глубине
    STRUCTURE  ON(71)  TYPE "String", Comment "Offset 2"     ! первый проход на
этом уровне
      RAPID      N 10000
      GOTO       X -28.715, Y -1.804, Z 0
      GOTO       X -28.715, Y -1.804, Z -3
      AXESBRAKE  COUNT 1, AxisCPos(3) ON(71)
      COOLNT     ON(71), N 1, #1
      FEDRAT     F 100, MMPM(315), K 256
      GOTO       X -28.715, Y -1.804, Z -4
      FEDRAT     F 100, MMPM(315), K 4
      GOTO       X -28.716, Y -3.004, Z -4
```

```
      GOTO       X -32.716, Y -3.001, Z -4
      FEDRAT     F 200, MMPM(315), K 0
      GOTO       X -39.887, Y -2.994, Z -4
      CIRCLE     XC -0.086, YC -0.024, ZC -4, R -39.911, XE -39.882, YE 3, ZE -4
      GOTO       X -32.725, Y 3.001, Z -4
      …
    STRUCTURE  OFF(72) TYPE "String", Comment "Offset 2"
    STRUCTURE  ON(71)  TYPE "String", Comment "Offset 1"     ! следующий проход — та
же форма — опущено
      …
    STRUCTURE  OFF(72) TYPE "String", Comment "Offset 1"
    STRUCTURE  ON(71)  TYPE "String", Comment "Offset 0"
      …
    STRUCTURE  OFF(72) TYPE "String", Comment "Offset 0"
  STRUCTURE  OFF(72) TYPE "String", Comment "Level: -4"
  …  (уровни -8, -12, -16 — те же проходы на каждой глубине; затем весь набор
повторяется после
      каждого поворота оси C: MULTIGOTO C 180 / 270 / 360 — много повторяющихся
проходов, опущено)
STRUCTURE  OFF(72) TYPE "BodySection", Comment "$Body$"
STRUCTURE  ON(71)  TYPE "TailSection", Comment "$Tail$"
  COOLNT     OFF(72), N 1, #1
  STRUCTURE  ON(71)  TYPE "String", Comment "Return"
    STRUCTURE  ON(71)  TYPE "Return", Comment "Return"
      PPRINT     "#Return:[Rule 1]=Z10; X; Z; C"           ! toolpath tag
      RAPID      N 10000
      MULTIGOTO  COUNT 1, MACHINE, Z 10, …
      GOHOME     COUNT 1, MACHINE, X 100, …
      GOHOME     COUNT 1, MACHINE, Z 500, …
      GOHOME     COUNT 1, MACHINE, Y 0, …
    STRUCTURE  OFF(72) TYPE "Return", Comment "Return"
  STRUCTURE  OFF(72) TYPE "String", Comment "Return"
  CUTCOM     OFF(72), LENGTH(9)0, …                        ! коррекция выключается
  SPINDL     OFF(72), NO 0, K 0                            ! шпиндель выключается
  PPFUN      ENDTECHINFO(59), 250, -47.996, -289.976, …    ! операция заканчивается
STRUCTURE  OFF(72) TYPE "TailSection", Comment "$Tail$"
```

Модель доступа

Параметры команды бывают двух видов:

- Числовые параметры — плоский массив вещественных чисел, адресуемых по позиции. Это исходная, унаследованная форма.
- Именованные / структурированные параметры — типизированные значения, адресуемые по имени, возможно вложенные (объекты, массивы). Это современная форма,

применяемая в новых командах (и доступная для большинства параметров старых). Модель именования — та же, что описана в XML-свойства — доступ из кода.

В таблице сведены способы их чтения. Подробности и примеры даются для каждой команды в справочнике команд; шаблон доступа всегда один и тот же.

Форма доступа	sppx	.NET	Примечания
Числовой массив, по индексу	`cld[i]`	`cld[i]`	Общее для обеих. Массив заполняется параметрами текущей команды перед её обработкой.
Числовой элемент, по короткому имени	`cld.X`, `cld.N` …	—	Только sppx. Плоский дублирующий доступ к некоторым числовым элементам. В .NET недоступен.
Именованный / структурированный, по имени	`cmd.Str["Name"]`, `cmd.Flt["Name"]`, `cmd.Int["Name"]`, `cmd.Ptr["Name"]`	`cmd.Str["Name"]`, `cmd.Flt["Name"]`, `cmd.Int["Name"]`, `cmd.Ptr["Name"]`	Одинаковая грамматика в обеих (пути через точку, ключи массива `Name(Key)`, цепочки указателей). См. XML-свойства — доступ из кода.
По стабильному коду параметра	`CmdPrm.Flt[code]`	`cmdPrm.Flt[code]`	Используется расширенными / измерительными циклами, где параметры адресуются фиксированным кодом, чтобы их порядок мог меняться без

Форма доступа	sppx	.NET	Примечания
			поломки постпроцессоров.
Строго типизированная обёртка команды	—	`cmd.<Property>` (напр. `cmd.R`, `cmd.IsCSS`)	Только .NET. Каждый обработчик команды получает типизированный объект `cmd` с именованными документированным и свойствами. Современный способ для .NET; см. справочник API.

sppx

В sppx числовые параметры текущей команды помещаются в предопределённый массив `cld` до запуска программы команды, поэтому `cld[1]`, `cld[2]`, … адресуют их по позиции; у некоторых элементов есть также короткие имена-члены, например `cld.X`. Структурированные параметры читаются по имени оператором `cmd` (`cmd.Str["…"]`, `cmd.Flt["…"]`, `cmd.Ptr["…"]`), а расширенные циклы используют доступ по коду `CmdPrm.Flt[code]`. Эти средства доступа описаны в разделе Функции и операторы доступа к CLData.

.NET

В .NET постпроцессор реализует типизированный обработчик на каждую команду, например `OnSpindle(ICLDSpindleCommand cmd, CLDArray cld)`. Внутри обработчика можно:

- использовать строго типизированный объект `cmd` — именованные документированные свойства, такие как `cmd.IsCSS`, `cmd.RPMValue`, `cmd.R`, `cmd.Center` (это рекомендуемая, современная форма);
- читать именованные / структурированные параметры через `cmd.Str["Name"]`, `cmd.Flt["Name"]` и т. д.;
- читать «сырой» числовой массив `cld[i]` (те же индексы, что и в sppx);
- использовать доступ по коду `cmdPrm.Flt[code]` для расширенных циклов.

Типизированные интерфейсы команд (`ICLD…Command`) и все их свойства описаны в справочнике API .NET SDK — см. Справочные материалы .NET.

Поиск команд (обход CLData)

Помимо текущей команды, постпроцессор может просматривать другие команды файла (заглядывать вперёд/назад, находить следующую команду заданного типа, читать контекст проекта / операции).

- sppx: `CurCode`, `NextCode`, `CodeOfCmd`, `GetCld`, `FindCld` / `GFindCld`, оператор `CLDFile` и оператор `Project`.
- .NET: `CurrentFile.Cmd[i]`, `CurrentFile.FindCommand(...)`, `CurrentFile.IndexOfCmd(...)`, `CurrentCmd.Next` / `Prev` / `NextMotion` / `PrevMotion`, `CurrentOperation.*`, `CLDProject.*`.

Обе формы описаны рядом в разделе Функции и операторы доступа к CLData.

Справочник команд

Каждая команда описана один раз для обеих подсистем (общие параметры → доступ из sppx → доступ из .NET). Полный список — в разделе Описание технологических команд.

Информация о проекте в CLData

Статические данные уровня проекта, встроенные в CLData (схема станка, детали, заготовки, установочные позиции, системы координат, единицы измерения и т. д.), описаны в разделе Информация о проекте в CLData.

В этом разделе

- Функции и операторы доступа к CLData — чтение CLData из кода постпроцессора
- Информация о проекте в CLData — статические данные уровня проекта в CLData
- Описание технологических команд — список всех команд CLData

Функции и операторы доступа к CLData

Язык программирования генератора постпроцессоров реализует несколько функций, переменных и операторов, дающих сведения о содержимом CLData. Идея одинакова в sppx и в .NET, но конкретные примитивы различаются, поэтому каждый раздел ниже описывает форму sppx и форму .NET рядом — параметры текущей команды, остальные команды файла и данные уровня проекта — и указывает, где у одной подсистемы нет прямого аналога.

Общую картину того, как постпроцессор добирается до CLData (модель команд, текущая команда, массив `cld`), см. в разделе Модель доступа к CLData.

В этом разделе

- Именованные параметры CLData — общая модель именованных параметров (`Str` / `Int` / `Flt` / `Ptr`, доступ по имени через точку/скобки); общая для средств доступа к команде, файлу и проекту.
- Предопределённый массив CLD — массив числовых параметров текущей команды (`CLD[i]` / `CLD.Name` в sppx, `CLDArray` в .NET).
- Текущая команда (оператор Cmd) — чтение кода, имени, данных и именованных параметров текущей команды, включая параметры, адресуемые уникальным кодом.
- Просмотр команд CLData — обращение к другим командам файла (заглядывание вперёд/назад, поиск следующей команды заданного типа, коды команд).
- Файлы CLData (оператор CLDFile) — доступ к любой команде любого загруженного файла CLData.
- Информация о проекте (оператор Project) — доступ к данным уровня проекта CAM-системы, встроенным в CLData.
- Массив GMA — удобный массив sppx для обработки многоосевых перемещений MULTIGOTO.

Сопутствующие ссылки

- Модель доступа к CLData
- Описание технологических команд
- Информация о проекте в CLData

Именованные параметры CLData

Из CAM-системы в постпроцессор приходит набор данных, называемый CLData, на основе которого постпроцессор формирует G-код. Данные внутри CLData могут описывать множество различных параметров, относящихся к совершенно разным сложным объектам: технологической операции, траектории, инструменту, проекту, заготовке и т. д. Поэтому внутри CLData они непременно структурированы и фактически являются не простым списком, а деревом параметров. Доступ к этим параметрам из кода постпроцессора выполняется по строковому имени. Поскольку мы имеем дерево параметров, обычно полное имя параметра — это составная строка из нескольких идентификаторов, разделённых точками, а иногда и скобками, то есть

GroupName.SubGroupName.ParameterName.SubParameterName и т. д. Уровень вложенности параметров может быть разным — от одного до нескольких идентификаторов, разделённых точками. Пример реального имени параметра, отвечающего за радиус сглаживания траектории внутри операции: "Operation.Smoothing.Radius".

Доступ из sppx

Синтаксис доступа к параметру из кода программы постпроцессора следующий.

```
ObjectKeyWord.TypeKeyWord["ParameterName"]
```

Здесь ObjectKeyWord — ключевое слово, указывающее, у какого объекта из CLData мы запрашиваем параметры. Сейчас в качестве ObjectKeyWord могут быть:

- Cmd — текущая команда CLData.
- CLDFile[*index1*].Cmd[*index2*]. Команда CLData с индексом index2 из cld-файла с индексом index1. Здесь index1 и index2 — любые целые числа.
- Project — проект CAM-системы, в котором сгруппированы параметры, относящиеся ко всем операциям, а не к отдельной операции, например заготовка.

TypeKeyWord — ключевое слово, определяющее форму (строка, число и т. д.) получения значения параметра. Параметры могут быть разных типов: string (строка), integer (целое), floating-point (число с плавающей точкой), complex (параметры этого типа не имеют собственного значения, но могут иметь дочерние свойства любого из описанных здесь типов; нужны для объединения нескольких параметров в группу), array (как и сложные, не имеют собственных значений и могут иметь дочерние свойства, однако дочерние свойства массивов, как правило, имеют общий тип и, следовательно, одинаковое имя, поэтому доступ к дочерним свойствам массивов выполняется не по имени, а по индексу).

Доступны следующие ключевые слова.

- `Str["ParameterName"]` — возвращает значение параметра команды с именем ParameterName в виде текстовой строки String.
- `Int["ParameterName"]` — возвращает значение параметра команды с именем ParameterName в виде целого числа Integer.
- `Flt["ParameterName"]` — возвращает значение параметра команды с именем ParameterName в виде вещественного числа Real (числа с плавающей точкой).
- `Ptr["ParameterName"]` — возвращает ссылку (не значение) на параметр команды с именем ParameterName. Удобно использовать для доступа к параметрам типа complex и array. Если параметра с таким именем в этой команде не существует, оператор возвращает значение 0, иначе — ненулевое число. Таким образом, проверку на равенство нулю можно использовать для проверки существования параметра.

```
if cmd.Ptr["Axes(AxisCPos)"]<>0 then begin
  C = cmd.Flt["Axes(AxisCPos).Value"]
end
```

От полученной ссылки через точку можно обратиться либо к значению этого параметра, либо к одному из дочерних параметров. Для этого после точки снова нужно указать одну из команд, описанных здесь: Str, Int, Flt, Ptr, Item, ItemCount, Name. Если в последующих инструкциях имя параметра не указано, это означает, что речь идёт о самом этом параметре. Например, инструкция `Cmd.Ptr["Parameter1"].Int` означает взять значение параметра с именем `Parameter1` как целое число. Запись `Cmd.Ptr["Parameter1"].Int["SubParameter1"]` означает взять целочисленное значение дочернего параметра `SubParameter1` внутри параметра `Parameter1`.

`Cmd.Item[Index]` — возвращает ссылку на дочерний параметр по индексу. Индексация параметров начинается с 1.

`Cmd.ItemCount` — возвращает число дочерних параметров в списке.

Для параметров типа array все дочерние параметры на один уровень ниже могут иметь одинаковое имя. В этом случае доступ по имени невозможен; вместо него нужно использовать доступ по индексу.

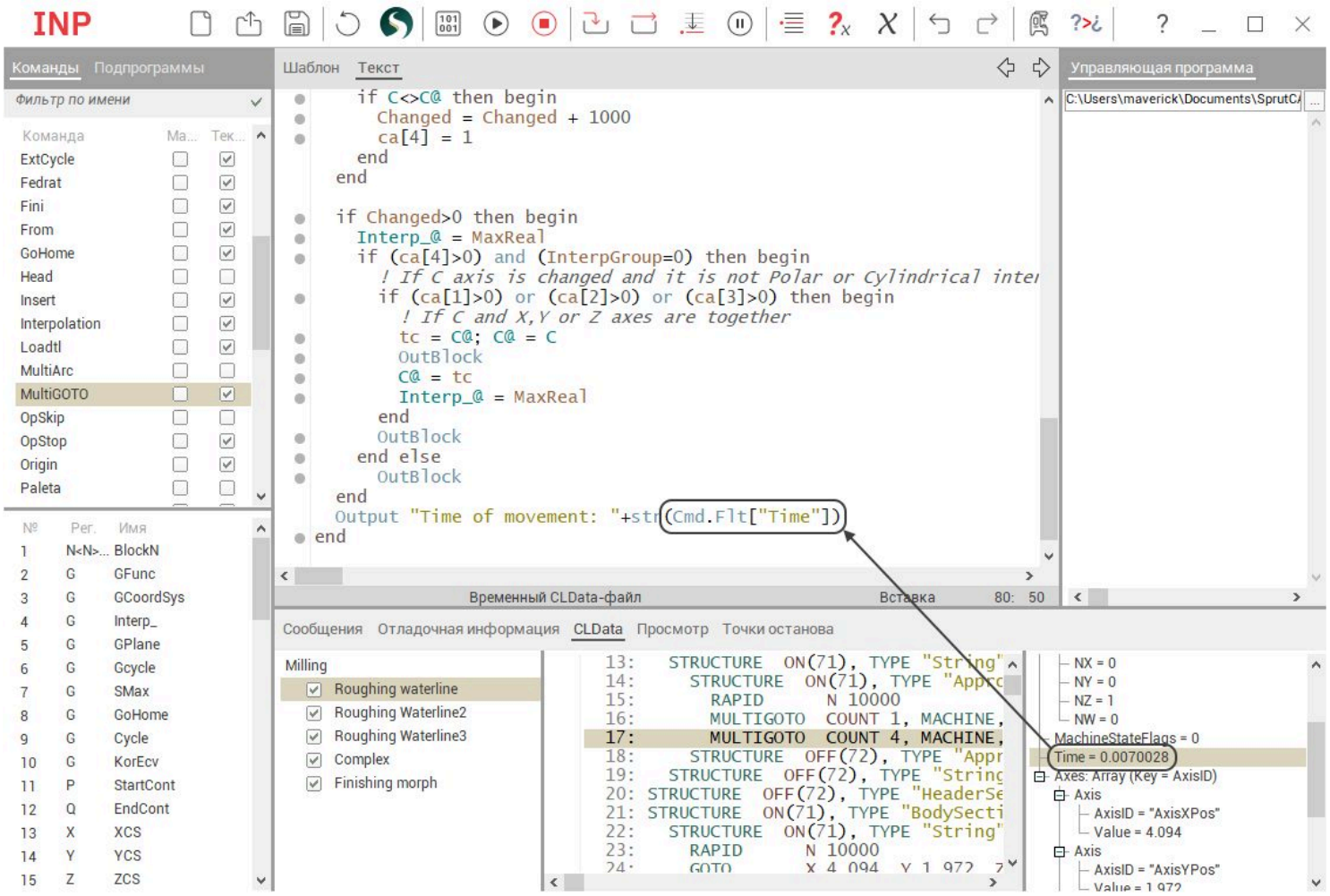
- `ObjectKeyWord.Ptr["NameOfArrayParameter"].Item[Index]` — возвращает ссылку на дочерний параметр по индексу. Индексация параметров начинается с 1.
- `ObjectKeyWord.Ptr["NameOfArrayParameter"].ItemCount` — возвращает число дочерних параметров в списке.

Помимо доступа по индексу, доступ к дочерним параметрам массива в некоторых случаях может выполняться по ключу. Если у массива определён параметр Key, задающий имя ключевого поля в дочерних параметрах, то для доступа к конкретному элементу массива

после имени массива нужно указать значение ключа в скобках. Например, инструкция `Cmd.Ptr["ArrayName(KeyValue)"]` означает взять параметр ArrayName из текущей команды, где найти дочернее свойство с ключевым полем, равным KeyValue.

В качестве имени параметра в них может быть не только строковая константа, как показано здесь, но и строковые переменные и выражения.

Есть удобный способ сформировать строку доступа к нужному параметру. Все параметры объектов CLData показаны отдельным деревом на вкладке CLData в её правой части. Нужно выбрать необходимую команду в списке или проект в списке операций. При этом в дереве справа будут показаны параметры выбранного объекта. Затем нужно найти в этом дереве нужный параметр и дважды щёлкнуть по нему. Строка доступа к указанному параметру будет вставлена в текущую позицию курсора в окне редактирования кода.



Пример обработки команды MultiGoto с использованием различных способов доступа к элементам параметра-массива Axes (по индексу, по ключу и по составному имени):

```
program MultiGoto
  i: Integer
  AxisName: String
```

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

Больше примеров доступа к именованным параметрам приведено в разделе Текущая команда.

Доступ из .NET

.NET использует ту же модель именованных параметров, предоставляемую интерфейсом `INamedProperty`. Её реализует каждый объект CLData, несущий именованные параметры — текущая команда (`ICLDCommand`), произвольная команда, взятая через CLDFile, и проект (`ICLDProject`). Ключевые слова соответствуют sppx-овским один к одному:

sppx	.NET	Описание
`Str["Name"]`	`Str["Name"]`	значение параметра как строка
`Int["Name"]`	`Int["Name"]`	значение параметра как целое число
`Flt["Name"]`	`Flt["Name"]`	значение параметра как число с плавающей точкой
-	`Bol["Name"]`	значение параметра как логическое значение
`Ptr["Name"]`	`Ptr["Name"]`	ссылка на (сложный) параметр как `INamedProperty`

sppx	.NET	Описание
-	`Arr["Name"]`	ссылка на параметр-массив как `INamedPropertiesCollection`
`Item[i]`	`Arr["Name"][i]`	i-й дочерний элемент параметра-массива (0-based в .NET)
`ItemCount`	`Arr["Name"].TopItem`	максимальный индекс в параметре-массиве
`Name`	`Name`	собственное имя параметра

> База индексации. Когда дочерние элементы параметра-массива адресуются по индексу, нумерация различается между двумя подсистемами: она 1-based в sppx (`Item[1]` … `Item[ItemCount]`) и 0-based в .NET (`Arr["Name"][0]` … `Arr["Name"][TopItem]`). Соответственно, sppx-овский `ItemCount` — это число дочерних элементов, тогда как .NET-овский `TopItem` — максимальный индекс, на единицу меньше числа, поэтому цикл в .NET выполняется как `for (int i = 0; i <= coll.TopItem; i++)`. (Это касается только дочерних элементов именованных параметров-массивов; числовой массив CLD 1-based в обеих подсистемах.)

Как и в sppx, имя может быть составной строкой с точками и скобками — `Ptr["StartPoint.X"]`, `Ptr["Axes(AxisCPos).Value"]` — разрешаемой на любую глубину вложенности; регистр символов не имеет значения. Кроме того, `INamedProperty` предлагает `FindChild(name)`, `GetNextChild(child)`, `Parent`, `SimpleType` (тип `NamedParamType` значения) и типизированные средства доступа к значению `ValueAsString` / `ValueAsInteger` / `ValueAsDouble` / `ValueAsBoolean`.

```
public override void OnMultiGoto(ICLDMultiGotoCommand cmd, CLDArray cld)
{
    var axes = cmd.Arr["Axes"];
    for (int i = 0; i <= axes.TopItem; i++) {
        string axisName = axes[i].Str["AxisID"];
        double value = axes[i].Flt["Value"];
        // ... вывод оси
    }
    // или обратиться к отдельной оси напрямую по её ключу:
    if (cmd.Ptr["Axes(AxisCPos)"] != null)
```

```
        C = cmd.Flt["Axes(AxisCPos).Value"];
}
```

Сопутствующие ссылки

- Текущая команда (оператор Cmd)
- Файлы CLData (оператор CLDFile)
- Информация о проекте (оператор Project)
- Функции и операторы доступа к CLData

Предопределённый массив CLD

Массив CLD хранит числовые параметры выполняемой в данный момент команды CLData. Перед вызовом обработчика команды параметры команды помещаются в этот массив; внутри обработчика параметры анализируются чтением отдельных элементов массива, и на основе этого анализа формируются кадры управляющей программы.

Параметры команды доступны либо индексированием соответствующего элемента массива, либо по уникальному имени параметра.

Доступ из sppx

Массив CLD — это массив вещественных чисел. Он предназначен для хранения числовых параметров выполняемой в данный момент технологической команды CLData. Перед вызовом обработчика текущей команды её параметры заполняются в этот массив. Внутри обработчика параметры команды можно анализировать, обращаясь к отдельным элементам массива, и на основе анализа формировать кадры управляющей программы.

Параметры команды доступны либо индексированием соответствующего элемента массива CLD, либо по уникальному имени параметра.

Для доступа по индексу используйте обычный синтаксис массива: `CLD[i]` — i-й элемент массива, вещественное число, i — индекс элемента, положительное целое число. Индекс элемента массива может быть переменной или выражением целого типа. Например: `CLD[3]`, `CLD[n]`, `CLD[2*n+1]` и т. д. Общее число элементов массива CLD содержится в предопределённой переменной RecNum.

Для доступа к параметрам по имени используйте следующий синтаксис: `CLD.Имя параметра`, где Имя параметра — уникальный идентификатор параметра. Например, `CLD.X`, `CLD.Mode` и так далее. Имена параметров специфичны для каждой команды. Подробное описание технологических команд CLData и их параметров приведено в описании технологических команд. Также можно увидеть список параметров на панели Текущие параметры страницы Шаблон главного окна и на вкладке CLData внизу главного окна.

Пример использования массива CLD:

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

```
  J = CLD.Yc
  R = abs(cld.R)
  OutBlock
end
```

Приведённый выше обработчик команды CIRCLE выводит в кадры управляющей программы движение по окружности G2 или G3 (по часовой стрелке или против) с конечной точкой (X, Y, Z), центром (I, J) и радиусом (R).

Доступ к параметрам команды CLData, в том числе нечисловым, можно также получить оператором Cmd.

Доступ из .NET

Обработчик команды получает тот же массив числовых параметров вторым аргументом — экземпляр класса `CLDArray` (по соглашению именуемый `cld`). Его члены отражают массив CLD из sppx:

sppx	.NET	Описание
`CLD[i]`	`cld[i]`	i-й параметр, вещественное число; индекс 1-based ([1..Count]), как и в sppx
`CLD.Name`	`cld["Name"]`	параметр, адресуемый по его уникальному текстовому имени (без учёта регистра)
`RecNum`	`cld.Count`	число параметров в массиве

Тот же обработчик CIRCLE в .NET:

```
public override void OnCircle(ICLDCircleCommand cmd, CLDArray cld)
{
    int interp = cld["R"] > 0 ? 3 : 2;   // G3 / G2
    double x = cld[5];
    double y = cld[6];
    double z = cld[7];
    double i = cld["Xc"];
    double j = cld["Yc"];
    double r = Math.Abs(cld["R"]);
    // ... вывод кадра
}
```

В .NET параметры обычно читаются вместо этого из строго типизированного объекта `cmd` (`cmd.EP.X`, `cmd.R`, ...) — см. Текущая команда; «сырой» массив `cld` доступен для случаев, которые не покрываются типизированными членами.

Сопутствующие ссылки

- Текущая команда (оператор Cmd)
- Именованные параметры CLData
- Функции и операторы доступа к CLData

Текущая команда (оператор Cmd)

Текущая команда — это команда CLData, обработчик которой выполняется. Помимо числового массива CLD, постпроцессор может прочитать код, имя, текстовые данные команды и любой из её именованных параметров (включая вложенные). Это обеспечивают оператор Cmd в sppx и строго типизированный объект `cmd` в .NET.

Доступ из sppx

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

`Cmd.Code` — возвращает уникальный числовой код команды CLData. Каждая команда однозначно определяется этим кодом. Список кодов команд приведён в описании технологических команд.

`Cmd.Name` — возвращает текстовое имя команды или параметра команды.

`Cmd.Data` — для отдельных команд CLData, имеющих единственный строковый параметр (таких как Comment, PartNo и т. п.), возвращает строковое значение. Для остальных команд возвращает строковое представление команды — в точности то, что отображается на вкладке CLData главного окна.

`Cmd.Data[Index]` — альтернативный способ доступа к массиву числовых параметров команды CLD. Возвращает вещественное значение элемента массива параметров с индексом Index.

Ниже приведено несколько языковых конструкций, дающих доступ к параметрам команды CLData по имени параметра. Подробнее см. раздел Именованные параметры CLData. Каждая команда CLData может характеризоваться собственным набором именованных параметров. Имена параметров с описанием для каждой команды перечислены в описании технологических команд. Имена параметров видны на вкладке CLData внизу главного окна.

`Cmd.Str["ParameterName"]` — возвращает значение параметра команды с именем ParameterName в виде текстовой строки String.

`Cmd.Int["ParameterName"]` — возвращает значение параметра команды с именем ParameterName в виде целого числа Integer.

`Cmd.Flt["ParameterName"]` — возвращает значение параметра команды с именем ParameterName в виде вещественного числа Real (числа с плавающей точкой).

`Cmd.Ptr["ParameterName"]` — возвращает ссылку на параметр команды с именем ParameterName.

Рассмотрим несколько примеров использования параметров команды CLData.

Пример 1. Внутри обработчика команды загрузки инструмента LoadTl вывести в управляющую программу строку с адресом T, номером инструмента и именем револьверной головки в скобках.

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

Возможный результат вывода в управляющую программу: T3 (TopTurret)

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

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

Пример 3. Пример обработки команды настройки системы координат Origin. Анализируется значение свойства OriginType. Если оно равно нулю, выводится стандартная команда выбора системы координат заготовки G54-G59. Номер системы координат берётся из параметра CSNumber. Иначе формируется команда переноса локальной системы координат G92 X Y Z. Значения смещений по осям системы координат берутся разными способами из соответствующих дочерних свойств параметра `MCS.OriginPoint`.

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

Оператор Cmd (параметры с уникальным кодом)

Для измерительных циклов параметры могут записываться в массив CLD в другом порядке. Каждый параметр имеет уникальный код, по которому можно получить значение параметра.

Синтаксис доступа к параметрам команды CLData по числовому коду:

- `CmdPrm.ItemCount` — число параметров команды;
- `CmdPrm.Flt[Code]` — возвращает значение параметра по его коду в виде Double;
- `CmdPrm.Int[Code]` — возвращает значение параметра по его коду в виде Integer;
- `CmdPrm.Str[Code]` — возвращает значение параметра по его коду в виде String;
- `CmdPrm.Bol[Code]` — возвращает значение параметра по его коду в виде Boolean;
- `CmdPrm.IndexOfCode(Code)` — найти индекс параметра по его коду;
- `CmdPrm.Item[i].ItemType` — тип параметра по индексу (String, Integer, Double, Boolean);
- `CmdPrm.Item[i].Flt` — значение параметра по индексу в виде Double;
- `CmdPrm.Item[i].Int` — значение параметра по индексу в виде Integer;
- `CmdPrm.Item[i].Str` — значение параметра по индексу в виде String;
- `CmdPrm.Item[i].Bol` — значение параметра по индексу в виде Boolean;
- `CmdPrm.Item[i].Code` — числовой код параметра по индексу.

Доступ из .NET

В .NET текущая команда — это первый аргумент обработчика, строго типизированный интерфейс (`ICLDCircleCommand`, `ICLDGotoCommand`, ...), производный от `ICLDCommand`. Чтение типизированных свойств (`cmd.EP.X`, `cmd.R`, ...) — рекомендуемый способ; перечисленные ниже члены покрывают ту же информацию, что и оператор Cmd в sppx:

sppx	.NET (**`ICLDCommand`**)	Описание
`Cmd.Code`	`cmd.CmdTypeCode`	уникальный числовой код типа команды

sppx	.NET (`ICLDCommand`)	Описание
-	`cmd.CmdType`	тип команды как перечисление `CLDCmdType`
`Cmd.Name`	`cmd.Caption`	имя команды, как оно отображается в окнах
`Cmd.Data`	`cmd.CLDataS`	текстовые данные команды (или её строковое представление)
`Cmd.Data[i]`	`cmd.CLD[i]`	i-й числовой параметр (см. Массив CLD)
-	`cmd.Index`	индекс команды в её файле CLData

Доступ к именованным параметрам (`Str` / `Int` / `Flt` / `Ptr`) идентичен, поскольку `ICLDCommand` реализует общую модель именованных свойств:

```
public override void OnLoadTl(ICLDLoadTlCommand cmd, CLDArray cld)
{
    nc.OutWithN("T" + cmd.Int["ToolID"] + "(" + cmd.Str["RevolverID"] + ")");
}
```

Вложенные и массивные параметры используют `cmd.Ptr["Name"]`, `.Arr["Name"]`, `.Item[i]` и составной синтаксис имён `"Group.Sub.Param"` / `"Array(Key)"`, в точности как в sppx — см. Именованные параметры CLData.

Параметры с уникальным кодом. В .NET-постпроцессорах почти всё так же, но доступ выполняется через ключевое слово `cmd.Prm` (`cmd.Prm.Flt[code]`, `cmd.Prm.Int[code]`, `cmd.Prm.IndexOfCode(code)`, `cmd.Prm.Item[i]...`).

Сопутствующие ссылки

- Именованные параметры CLData
- Предопределённый массив CLD
- Просмотр команд CLData
- Функции и операторы доступа к CLData

Просмотр команд CLData

Помимо текущей команды, постпроцессор может обращаться к другим командам файла — прочитать код текущей или следующей команды, заглянуть вперёд в поисках команды заданного типа или обойти весь файл. sppx предлагает для этого набор специальных функций; .NET делает то же самое через навигационные члены `ICLDCommand` и объект CLDFile. Коды/типы команд — связующая нить: sppx определяет тип команды числовым кодом (получаемым из имени функцией `CodeOfCmd`), .NET — перечислением `CLDCmdType`.

Доступ из sppx

Функция CurCode

Функция CurCode возвращает код текущей команды CLData. Каждая команда CLData имеет уникальный числовой идентификатор — код. Коды команд CLData перечислены в описании технологических команд. Пример функции внутри обработчика команды CIRCLE ниже.

```
program Circle
  Print CurCode
end
```

В результате выполнения программы в окно отладки выводится число 15000.

Код следующей команды CLData можно получить функцией NextCode. Для получения кода произвольной команды по её текстовому имени следует использовать функцию CodeOfCmd.

Функция NextCode

Функция NextCode возвращает код технологической команды, расположенной непосредственно после текущей команды. Каждая команда CLData имеет уникальный числовой идентификатор — код. Коды команд CLData перечислены в описании технологических команд. Код текущей команды CLData можно получить функцией CurCode. Для получения кода произвольной команды по её текстовому имени следует использовать функцию CodeOfCmd.

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

Функция CodeOfCmd

`CodeOfCmd(CommandName: String)` возвращает код команды CLData, имя которой передано параметром. Каждая команда CLData имеет уникальный числовой идентификатор — код. Коды

команд CLData перечислены в описании технологических команд. Функция имеет следующий формат.

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

CommandName — текстовое имя команды. Оно должно быть одним из имён команд, которые видны в списке CLData в левой части главного окна. Возвращаемое значение можно использовать с операторами вроде CurCode, NextCode, GetCLD, `CLDFile[i].Cmd[j].Code` (см. оператор CLDFile).

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

Функция CLDCounter

Функция CLDCounter возвращает номер текущей технологической команды в файле списка технологических команд. Для гибкого доступа к произвольной команде CLData по её индексу в файле можно использовать оператор CLDFile.

Функция GetCLD

Оператор GetCLD используется для просмотра команд CLData в порядке их следования в файле. Просмотр возможен только в прямом направлении. Синтаксис:

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

Здесь:

- CldCode — GetCLD необязательно возвращает код полученной команды CLData.
- i — относительный индекс запрашиваемой команды CLData:
  - 0 — текущая команда,
  - 1 — следующая команда,
  - 2 — вторая команда от текущей и так далее.
- a — динамический массив вещественных чисел, который получит данные полученной команды CLData. Он должен быть определён до вызова оператора.

Например, пусть список команд CLDATA следующий и пусть текущая команда — номер 9:

```
5: CUTCOM ON(71),LENGTH(9) 2,X 0,Y 0,Z 0,N 0,K 0,M 0,LEFT(8)
6: RAPID N 10000
```

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

Программа технологической команды GOTO.abs следующая:

```
program AbsMov
  a: Array of Real ! Массив должен быть объявлен заранее
  c: Integer ! В эту переменную будет записан код операции
  c = GetCld(1, a) ! Получение данных следующей команды
  if c = CodeOfCmd("FEDRAT") then
    ! Вывод значения первого элемента массива CLD
  Print "FEDRAT.CLD[1]=", Str(a[1]) ! следующей команды
end
```

Результатом выполнения программы будет:

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

Для доступа к параметрам произвольной технологической команды можно также использовать оператор CLDFile.

Функция GetCLDStr

GetCLDStr возвращает строковое представление текущей команды CLData с её параметрами, аналогичное тому, что показано в окне текстового представления CLData.

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

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

Функция FindCld

Функция FindCld используется для поиска команды с именем CmdName в текущем файле CLData. Параметр StartIndex задаёт первую просматриваемую команду. StartIndex отсчитывается от первой команды в текущем файле CLData. Если параметр StartIndex опущен, оператор просматривает команды начиная с команды, непосредственно следующей за

текущей. Возвращаемое значение — индекс найденной команды. Если команда не найдена, оператор возвращает -1. Параметр Data должен быть объявленным массивом вещественных чисел. Data получит значения из массива CLData найденной команды. Синтаксис:

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

Например, пусть дана следующая CLDATA и пусть номер текущей команды равен 0.

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

Программа PartNo следующая:

```
program PartNo
  V: Array of Real
  NCircle: Integer
  NGoto: Integer
  NCircle = FindCld("CIRCLE", V)
  NGoto = FindCld("GOTO.abs", V)
  if NGoto > NCircle then
    Print "The first movement is an arc"
  else
    Print "The first movement a cut"
End
```

В результате выполнения программы будет напечатано сообщение "The first movement a cut".

Для доступа к параметрам произвольной технологической команды можно также использовать оператор CLDFile.

Функция GFindCld

Функция GFindCld аналогична функции FindCLD, но она ищет команду с именем CmdName во всех файлах CLData. Параметр StartIndex задаёт первую просматриваемую команду. StartIndex отсчитывается от первой команды в первом файле CLData. Если параметр StartIndex опущен, оператор просматривает команды начиная с команды, непосредственно следующей за

текущей. Возвращаемое значение — индекс найденной команды. Если команда не найдена, оператор возвращает -1. Параметр Data должен быть объявленным массивом вещественных чисел. Data получит значения из массива CLData найденной команды. Синтаксис:

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

Для доступа к параметрам произвольной технологической команды можно также использовать оператор CLDFile.

Доступ из .NET

В .NET нет функций просмотра из sppx; вместо этого каждая команда (`ICLDCommand`) знает своих соседей и может выполнять поиск от себя, а объект CLDFile ищет внутри файла. Типы команд задаются перечислением `CLDCmdType`, а не числовым кодом, полученным из имени.

sppx	.NET	Описание
`CurCode`	`cmd.CmdTypeCode` / `cmd.CmdType`	код типа / тип текущей команды
`CLDCounter`	`cmd.Index`	индекс текущей команды в её файле
`CodeOfCmd("NAME")`	`CLDCmdType.Name`	тип команды — именованное значение перечисления вместо искомого кода
`GetCLD(0, a)` / `GetCLD(1, a)`	`cmd` / `cmd.Next`	текущая / следующая команда (`cmd.Prev` для предыдущей)
-	`cmd.NextMotion` / `cmd.PrevMotion`	следующая / предыдущая команда перемещения
`NextCode`	`cmd.Next?.CmdType`	тип следующей команды
`FindCld(name, data)`	`cmd.FindNextCommand(type[, offset[, count]])`	найти следующую команду типа в том же файле
`GFindCld(name, data)`	`cmd.FindNextCommandOfProject(type[, ...])`	найти следующую команду типа по всем файлам

sppx	.NET	Описание
`GetCLDStr`	`cmd.CLDataS`	текстовое представление команды

`cmd.Next` / `Prev` возвращают `null` за пределами файла; методы `FindNext...` возвращают `null`, когда ничего не найдено. Массив `CLData` найденной команды читается через её член `CLD` (см. Массив CLD), поэтому отдельный аргумент-массив для вывода не нужен.

```
// sppx: if NextCode = CodeOfCmd("Fini") then Output "M02"
if (cmd.Next != null && cmd.Next.CmdType == CLDCmdType.Fini)
    nc.OutWithN("M02");

// sppx: NGoto = FindCld("GOTO.abs", V)
ICLDCommand goto = cmd.FindNextCommand(CLDCmdType.Goto);
if (goto != null) {
    double firstParam = goto.CLD[1];
    // ...
}
```

Чтобы обратиться к произвольной команде по индексу или обойти весь файл или проект, используйте объекты CLDFile / проект (`FindCommand`, `IndexOfCmd`).

Сопутствующие ссылки

- Текущая команда (оператор Cmd)
- Файлы CLData (оператор CLDFile)
- Функции и операторы доступа к CLData

Файлы CLData (оператор CLDFile)

CLData проекта CAM разбита на файлы — по одному на технологическую операцию, плюс файлы подпрограмм УП и файл уровня проекта. Постпроцессор может обратиться к любой команде любого файла (не только текущего), чтобы заглянуть вперёд, собрать списки инструментов и т. д. В sppx это делается оператором CLDFile; в .NET — списком `ICLDProject.CLDFiles` объектов `ICLDFile`.

Доступ из sppx

Оператор CldFile предназначен для получения доступа к данным файлов CLData из кода постпроцессора. Он предоставляет информацию только для чтения, то есть может находиться только справа от оператора присваивания. Синтаксис:

- `CldFile.FileCount` — возвращает число загруженных файлов CLData;
- `CldFile.CurrentFile` — индекс текущего файла CLData (файлы CLData индексируются начиная с номера 0);
- `CldFile.FirstEnabledFile` — индекс первого файла CLD (операции), включённого в списке на вкладке CLData;
- `CldFile.LastEnabledFile` — индекс последнего файла CLD (операции), включённого в списке на вкладке CLData;
- `CldFile.CurrentCmd` — индекс текущей команды CLData в текущем файле CLData (индексы 1-based);
- `CldFile[<FileIndex>].Enabled` — возвращает 1, когда файл номер FileIndex включён, 0 — файл отключён;
- `CldFile[<FileIndex>].FileType` — возвращает тип файла CLData: 0 — технологическая операция, 1 — подпрограмма УП, 2 — настройки проекта CAM;
- `CldFile[<FileIndex>].IsNCSub` — возвращает 1, когда файл является файлом подпрограммы УП, 0 — когда файл не является подпрограммой УП;
- `CldFile[<FileIndex>].CmdCount` — число команд в файле CLData номер FileIndex;
- `CldFile[<FileIndex>].Cmd[<CmdIndex>]` — возвращает ссылку на команду CLData с номером CmdIndex в файле с индексом FileIndex. После этой команды через точку допустимо указать любую из инструкций, определённых для оператора Cmd. Например: `CldFile[<FileIndex>].Cmd[<CmdIndex>].Code` — код команды CLData с номером CmdIndex;
- `CldFile[<FileIndex>].Cmd[<CmdIndex>].Data` — возвращает строковые данные команды CLData с номером CmdIndex (например, строку комментария). Если команда не содержит строковых данных, возвращает её строковое представление.
- `CldFile[<FileIndex>].Cmd[<CmdIndex>].Data[<CldIndex>]` — значение элемента массива CLD с номером CldIndex.
- `CldFile[<FileIndex>].Cmd[<CmdIndex>].Name` — имя команды CLData или параметра команды CLData;

- `CldFile[<FileIndex>].Cmd[<CmdIndex>].Str |.Int |.Flt |.Ptr` — значение параметра команды по его имени в соответствующей форме; подробнее см. раздел Именованные параметры CLData

Пример ниже печатает номера и комментарии всех инструментов, используемых в программе.

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

Доступ из .NET

Список файлов достигается из объекта проекта как `CLDProject.CLDFiles` (`ICLDFileList`); текущий файл доступен напрямую как `CurrentFile`. Каждый элемент — это `ICLDFile`. Обратите внимание, что индекс команды в файле в .NET 0-based (`[0..CmdCount-1]`), тогда как индекс команды `CLDFile[f].Cmd[c]` в sppx 1-based.

`ICLDFileList`:

sppx	.NET	Описание
`CldFile.FileCount`	`CLDProject.CLDFiles.FileCount`	число загруженных файлов
`CldFile[f]`	`CLDProject.CLDFiles[f]`	файл с индексом `f`

sppx	.NET	Описание
`CldFile.CurrentFile`	`CurrentFile`	обрабатываемый файл

`ICLDFile`:

sppx	.NET	Описание
`CldFile[f].CmdCount`	`file.CmdCount`	число команд в файле
`CldFile[f].Cmd[c]`	`file.Cmd[c]` / `file[c]`	команда с индексом `c` (0-based)
`CldFile[f].Cmd[c].Code`	`file.GetCommandType(c)`	тип команды с индексом `c` (`CLDCmdType`)
`CldFile[f].Enabled`	`file.Enabled`	обрабатывается ли файл
`CldFile[f].FileType`	`file.FileType`	тип файла (`CLDFileType.TechOperation` / `NCSub` / `CAMProject`)
`CldFile[f].IsNCSub`	`file.FileType == CLDFileType.NCSub`	является ли файл подпрограммой УП
-	`file.Caption` / `file.ID`	краткое описание / уникальный идентификатор файла
-	`file.Index`	индекс файла в проекте
-	`file.Project`	проект, которому принадлежит файл
-	`file.IndexOfCmd(type, from, to)`	индекс первой команды типа в диапазоне (`-1`, если нет)
-	`file.IndexOfCmdName(name, from, to)`	то же, по имени типа
-	`file.FindCommand(type, from[, to])`	первая команда типа в диапазоне (`null`, если нет)

Тот же обход «печать всех инструментов» в .NET:

```
foreach... // по CLDProject.CLDFiles
for (int f = 0; f < CLDProject.CLDFiles.FileCount; f++) {
    ICLDFile file = CLDProject.CLDFiles[f];
    if (!file.Enabled || file.FileType == CLDFileType.NCSub)
        continue;
    for (int c = 0; c < file.CmdCount; c++) {
        ICLDCommand command = file.Cmd[c];
        if (command.CmdType == CLDCmdType.LoadTl) {
            nc.OutWithN("RevolverID = " + command.Str["RevolverID"]);
            nc.OutWithN("Tool number = " + command.CLD[1]);
            ICLDCommand next = file.Cmd[c + 1];
            if (next != null && next.CmdType == CLDCmdType.Comment)
                nc.OutWithN("Tool comment: " + next.CLDataS);
        }
    }
}
```

Сопутствующие ссылки

- Именованные параметры CLData
- Текущая команда (оператор Cmd)
- Просмотр команд CLData
- Информация о проекте (оператор Project)
- Функции и операторы доступа к CLData

Информация о проекте (оператор Project)

Часть CLData — это данные уровня проекта: сведения, относящиеся ко всему проекту CAM, а не к отдельной операции (схема станка, детали и заготовки, установочные позиции и т. д.). Постпроцессор читает их оператором Project в sppx или объектом `ICLDProject` в .NET. Каталог доступных свойств проекта и их смысл описаны в разделе Информация о проекте в CLData.

Доступ из sppx

Оператор Project позволяет получить из кода постпроцессора доступ к информации о проекте CAM-системы (информации, относящейся ко всему проекту, а не к отдельным операциям), которая лежит в CLData. Синтаксис оператора следующий.

```
Project.Ptr|Str|Int|Flt["ParameterName"]
```

Здесь Project — ключевое слово для доступа к параметрам уровня проекта CAM-системы.

Ptr, Str, Int или Flt — ключевое слово, определяющее форму, в которой требуется представить значение свойства: как строку, как целое число, как число с плавающей точкой или как указатель на свойство. Подробнее об этом см. в разделе Именованные параметры CLData.

Далее в квадратных скобках следует строковый идентификатор свойства, значение которого требуется получить. Это может быть строковая константа, как показано здесь, а также любое строковое выражение или переменная.

Описание возможных свойств и их значений см. в приложении в статье Информация о проекте в CLData.

Рассмотрим пример получения информации о заготовке с помощью этого оператора.

```
                    PrintAllWorkpiece
sub PrintAllWorkpiece
  i: Integer
  j: Integer
  px, py, pz, pv: Real
  for i = 1 to Project.Ptr["Parts"].ItemCount do begin
    Output "( Part" + str(i) + ".Workpiece )"
    for j = 1 to Project.Ptr["Parts"].Item[i].Ptr["Workpiece"].ItemCount do begin
      Output "(     Primitive " + str(j) + " )"
      case
Project.Ptr["Parts"].Item[i].Ptr["Workpiece"].Item[j].Int["PrimitiveType"] of
        0: begin !Неизвестно
          Output "(         Type: Unknown )"
```

```
        end
        1: begin !Пустая заготовка
          Output "(         Type: Empty workpiece )"
        end
        2: begin !Грани
          Output "(         Type: Faces )"
          px =
Project.Ptr["Parts"].Item[i].Ptr["Workpiece"].Item[j].Flt["Box.Min.X"]
          py =
Project.Ptr["Parts"].Item[i].Ptr["Workpiece"].Item[j].Flt["Box.Min.Y"]
          pz =
Project.Ptr["Parts"].Item[i].Ptr["Workpiece"].Item[j].Flt["Box.Min.Z"]
          Output "(         Min point: (" + str(px) + ", " + str(py) + ", " +
str(pz) + " ) )"
          px =
Project.Ptr["Parts"].Item[i].Ptr["Workpiece"].Item[j].Flt["Box.Max.X"]
          py =
Project.Ptr["Parts"].Item[i].Ptr["Workpiece"].Item[j].Flt["Box.Max.Y"]
          pz =
Project.Ptr["Parts"].Item[i].Ptr["Workpiece"].Item[j].Flt["Box.Max.Z"]
          Output "(         Max point: (" + str(px) + ", " + str(py) + ", " +
str(pz) + " ) )"
        end
        3: begin !Отливка
          Output "(         Type: Casting )"
          pv = Project.Ptr["Parts"].Item[i].Ptr["Workpiece"].Item[j].Flt["Stock"]
          Output "(         Stock: " + str(pv) + " )"
          px =
Project.Ptr["Parts"].Item[i].Ptr["Workpiece"].Item[j].Flt["Box.Min.X"]
          py =
Project.Ptr["Parts"].Item[i].Ptr["Workpiece"].Item[j].Flt["Box.Min.Y"]
          pz =
Project.Ptr["Parts"].Item[i].Ptr["Workpiece"].Item[j].Flt["Box.Min.Z"]
          Output "(         Min point: (" + str(px) + ", " + str(py) + ", " +
str(pz) + " ) )"
          px =
Project.Ptr["Parts"].Item[i].Ptr["Workpiece"].Item[j].Flt["Box.Max.X"]
          py =
Project.Ptr["Parts"].Item[i].Ptr["Workpiece"].Item[j].Flt["Box.Max.Y"]
          pz =
Project.Ptr["Parts"].Item[i].Ptr["Workpiece"].Item[j].Flt["Box.Max.Z"]
          Output "(         Max point: (" + str(px) + ", " + str(py) + ", " +
str(pz) + " ) )"
        end
        4: begin !Параллелепипед
          Output "(         Type: Box )"
```

```
          px = Project.Ptr["Parts"].Item[i].Ptr["Workpiece"].Item[j].Flt["Min.X"]
          py = Project.Ptr["Parts"].Item[i].Ptr["Workpiece"].Item[j].Flt["Min.Y"]
          pz = Project.Ptr["Parts"].Item[i].Ptr["Workpiece"].Item[j].Flt["Min.Z"]
          Output "(         Min point: (" + str(px) + ", " + str(py) + ", " +
str(pz) + " ) )"
          px = Project.Ptr["Parts"].Item[i].Ptr["Workpiece"].Item[j].Flt["Max.X"]
          py = Project.Ptr["Parts"].Item[i].Ptr["Workpiece"].Item[j].Flt["Max.Y"]
          pz = Project.Ptr["Parts"].Item[i].Ptr["Workpiece"].Item[j].Flt["Max.Z"]
          Output "(         Max point: (" + str(px) + ", " + str(py) + ", " +
str(pz) + " ) )"
        end
        5: begin !Тело вращения
          Output "(         Type: Turn envelope )"
          px = Project.Ptr["Parts"].Item[i].Ptr["Workpiece"].Item[j].Flt["Origin.X"]
          py = Project.Ptr["Parts"].Item[i].Ptr["Workpiece"].Item[j].Flt["Origin.Y"]
          pz = Project.Ptr["Parts"].Item[i].Ptr["Workpiece"].Item[j].Flt["Origin.Z"]
          Output "(         Origin: (" + str(px) + ", " + str(py) + ", " + str(pz) +
" ) )"
          px = Project.Ptr["Parts"].Item[i].Ptr["Workpiece"].Item[j].Flt["Axis.X"]
          py = Project.Ptr["Parts"].Item[i].Ptr["Workpiece"].Item[j].Flt["Axis.Y"]
          pz = Project.Ptr["Parts"].Item[i].Ptr["Workpiece"].Item[j].Flt["Axis.Z"]
          Output "(         Axis: (" + str(px) + ", " + str(py) + ", " + str(pz) + "
) )"
        end
        6: begin !Цилиндр
          Output "(         Type: Cylinder )"
          px = Project.Ptr["Parts"].Item[i].Ptr["Workpiece"].Item[j].Flt["Origin.X"]
          py = Project.Ptr["Parts"].Item[i].Ptr["Workpiece"].Item[j].Flt["Origin.Y"]
          pz = Project.Ptr["Parts"].Item[i].Ptr["Workpiece"].Item[j].Flt["Origin.Z"]
          Output "(         Origin: (" + str(px) + ", " + str(py) + ", " + str(pz) +
" ) )"
          px = Project.Ptr["Parts"].Item[i].Ptr["Workpiece"].Item[j].Flt["Axis.X"]
          py = Project.Ptr["Parts"].Item[i].Ptr["Workpiece"].Item[j].Flt["Axis.Y"]
          pz = Project.Ptr["Parts"].Item[i].Ptr["Workpiece"].Item[j].Flt["Axis.Z"]
          Output "(         Axis: (" + str(px) + ", " + str(py) + ", " + str(pz) + "
) )"
          pv = Project.Ptr["Parts"].Item[i].Ptr["Workpiece"].Item[j].Flt["HMin"]
          Output "(         Min level: " + str(pv) + " )"
          pv = Project.Ptr["Parts"].Item[i].Ptr["Workpiece"].Item[j].Flt["HMax"]
          Output "(         Max level: " + str(pv) + " )"
          pv = Project.Ptr["Parts"].Item[i].Ptr["Workpiece"].Item[j].Flt["ROut"]
          Output "(         Outer radius: " + str(pv) + " )"
        end
        7: begin !Труба
          Output "(         Type: Tube )"
          px = Project.Ptr["Parts"].Item[i].Ptr["Workpiece"].Item[j].Flt["Origin.X"]
```

```
          py = Project.Ptr["Parts"].Item[i].Ptr["Workpiece"].Item[j].Flt["Origin.Y"]
          pz = Project.Ptr["Parts"].Item[i].Ptr["Workpiece"].Item[j].Flt["Origin.Z"]
          Output "(         Origin: (" + str(px) + ", " + str(py) + ", " + str(pz) +
" ) )"
          px = Project.Ptr["Parts"].Item[i].Ptr["Workpiece"].Item[j].Flt["Axis.X"]
          py = Project.Ptr["Parts"].Item[i].Ptr["Workpiece"].Item[j].Flt["Axis.Y"]
          pz = Project.Ptr["Parts"].Item[i].Ptr["Workpiece"].Item[j].Flt["Axis.Z"]
          Output "(         Axis: (" + str(px) + ", " + str(py) + ", " + str(pz) + "
) )"
          pv = Project.Ptr["Parts"].Item[i].Ptr["Workpiece"].Item[j].Flt["HMin"]
          Output "(         Min level: " + str(pv) + " )"
          pv = Project.Ptr["Parts"].Item[i].Ptr["Workpiece"].Item[j].Flt["HMax"]
          Output "(         Max level: " + str(pv) + " )"
          pv = Project.Ptr["Parts"].Item[i].Ptr["Workpiece"].Item[j].Flt["ROut"]
          Output "(         Outer radius: " + str(pv) + " )"
          pv = Project.Ptr["Parts"].Item[i].Ptr["Workpiece"].Item[j].Flt["RInn"]
          Output "(         Inner radius: " + str(pv) + " )"
        end
        8: begin !Призма
          Output "(         Type: Prism )"
          px = Project.Ptr["Parts"].Item[i].Ptr["Workpiece"].Item[j].Flt["Origin.X"]
          py = Project.Ptr["Parts"].Item[i].Ptr["Workpiece"].Item[j].Flt["Origin.Y"]
          pz = Project.Ptr["Parts"].Item[i].Ptr["Workpiece"].Item[j].Flt["Origin.Z"]
          Output "(         Origin: (" + str(px) + ", " + str(py) + ", " + str(pz) +
" ) )"
          px = Project.Ptr["Parts"].Item[i].Ptr["Workpiece"].Item[j].Flt["Axis.X"]
          py = Project.Ptr["Parts"].Item[i].Ptr["Workpiece"].Item[j].Flt["Axis.Y"]
          pz = Project.Ptr["Parts"].Item[i].Ptr["Workpiece"].Item[j].Flt["Axis.Z"]
          Output "(         Axis: (" + str(px) + ", " + str(py) + ", " + str(pz) + "
) )"
          pv = Project.Ptr["Parts"].Item[i].Ptr["Workpiece"].Item[j].Flt["HMin"]
          Output "(         Min level: " + str(pv) + " )"
          pv = Project.Ptr["Parts"].Item[i].Ptr["Workpiece"].Item[j].Flt["HMax"]
          Output "(         Max level: " + str(pv) + " )"
        end
        9: begin !Многогранная призма
          Output "(         Type: Polygonal prism )"
          px = Project.Ptr["Parts"].Item[i].Ptr["Workpiece"].Item[j].Flt["Origin.X"]
          py = Project.Ptr["Parts"].Item[i].Ptr["Workpiece"].Item[j].Flt["Origin.Y"]
          pz = Project.Ptr["Parts"].Item[i].Ptr["Workpiece"].Item[j].Flt["Origin.Z"]
          Output "(         Origin: (" + str(px) + ", " + str(py) + ", " + str(pz) +
" ) )"
          px = Project.Ptr["Parts"].Item[i].Ptr["Workpiece"].Item[j].Flt["Axis.X"]
          py = Project.Ptr["Parts"].Item[i].Ptr["Workpiece"].Item[j].Flt["Axis.Y"]
          pz = Project.Ptr["Parts"].Item[i].Ptr["Workpiece"].Item[j].Flt["Axis.Z"]
          Output "(         Axis: (" + str(px) + ", " + str(py) + ", " + str(pz) + "
```

```
) )"
          pv =
Project.Ptr["Parts"].Item[i].Ptr["Workpiece"].Item[j].Int["CornerCount"]
          Output "(         Corner count: " + str(pv) + " )"
          pv = Project.Ptr["Parts"].Item[i].Ptr["Workpiece"].Item[j].Flt["RInn"]
          Output "(         Inscribed radius: " + str(pv) + " )"
          pv = Project.Ptr["Parts"].Item[i].Ptr["Workpiece"].Item[j].Flt["Angle"]
          Output "(         Angle around axis: " + str(pv) + " )"
          pv = Project.Ptr["Parts"].Item[i].Ptr["Workpiece"].Item[j].Flt["HMin"]
          Output "(         Min level: " + str(pv) + " )"
          pv = Project.Ptr["Parts"].Item[i].Ptr["Workpiece"].Item[j].Flt["HMax"]
          Output "(         Max level: " + str(pv) + " )"
        end
        else begin
          Output "(         Type: Something else )"
        end
      end
    end
  end
subend
```

Результатом этой подпрограммы может быть следующий текстовый блок.

```
                    Results
( Part1.Workpiece )
(     Primitive 1 )
(         Type: Tube )
(         Origin: (0, 0, 117.5 ) )
(         Axis: (0, 0, 1 ) )
(         Min level: -149.5 )
(         Max level: 149.5 )
(         Outer radius: 66 )
(         Inner radius: 29 )
(     Primitive 2 )
(         Type: Cylinder )
(         Origin: (0, 0, 117.5 ) )
(         Axis: (0, 0, 1 ) )
(         Min level: -147.5 )
(         Max level: 147.5 )
(         Outer radius: 64 )
(     Primitive 3 )
(         Type: Turn envelope )
(         Origin: (0, 0, 117.5 ) )
(         Axis: (0, 0, 1 ) )
(     Primitive 4 )
```

```
(         Type: Box )
(         Min point: (-64, -64, -30 ) )
(         Max point: (64, 64, 265 ) )
(     Primitive 5 )
(         Type: Casting )
(         Stock: 1 )
(         Min point: (-65, -65, -31 ) )
(         Max point: (65, 65, 266 ) )
(     Primitive 6 )
(         Type: Faces )
(         Min point: (-54, -54, -30 ) )
(         Max point: (54, 54, 265 ) )
(     Primitive 7 )
(         Type: Polygonal prism )
(         Origin: (0, 0, 117.5 ) )
(         Axis: (0, 1, 0 ) )
(         Corner count: 6 )
(         Inscribed radius: 160.786 )
(         Angle around axis: 0 )
(         Min level: -64 )
(         Max level: 64 )
(     Primitive 8 )
(         Type: Empty workpiece )
```

Доступ из .NET

Проект доступен обработчику как объект `CLDProject` (`ICLDProject`); он также передаётся в `OnStartProject` / `OnFinishProject`. `ICLDProject` реализует общую модель именованных свойств, поэтому свойства проекта читаются в точности как оператором Project в sppx — `CLDProject.Ptr["Parts"]`, `CLDProject.Flt["..."]`, `CLDProject.Arr["Parts"][i]...` — с теми же именами через точку/скобки. Помимо именованных свойств он предоставляет:

.NET (**`ICLDProject`**)	Описание
`CLDProject.ProjectName`	имя проекта (без пути к файлу)
`CLDProject.FilePath`	полный путь к файлу проекта (пустой, если ещё не сохранён)
`CLDProject.CLDFiles`	список файлов CLData (`ICLDFileList`, см. Файлы CLData)
`CLDProject.Operations`	список технологических операций

.NET (**`ICLDProject`**)	Описание
`CLDProject.CLDSub`	список подпрограмм CLD
`CLDProject.Machine`	сведения о станке (имя, группа типа, список осей)
`CLDProject.FindCommand(type, startFile, startCommand)`	найти команду типа по всему проекту

Тот же обход заготовки в .NET обходит то же именованное дерево (обратите внимание на индексы массивов, начинающиеся с 0, — см. примечание о базе индексации в разделе Именованные параметры CLData). Полный метод, готовый к вставке в постпроцессор — вызывайте его, например, из `OnStartProject`:

```
// Выводит комментарий для каждого элемента заготовки каждой детали в
управляющую программу.
public void PrintAllWorkpiece()
{
    var parts = CLDProject.Arr["Parts"];
    for (int i = 0; i <= parts.TopItem; i++)
    {
        nc.OutWithN("( Part" + (i + 1) + ".Workpiece )");
        var workpiece = parts[i].Arr["Workpiece"];
        for (int j = 0; j <= workpiece.TopItem; j++)
        {
            var wp = workpiece[j];
            nc.OutWithN("(     Primitive " + (j + 1) + " )");
            switch (wp.Int["PrimitiveType"])
            {
                case 0: // Неизвестно
                    nc.OutWithN("(         Type: Unknown )");
                    break;
                case 1: // Пустая заготовка
                    nc.OutWithN("(         Type: Empty workpiece )");
                    break;
                case 2: // Грани
                    nc.OutWithN("(         Type: Faces )");
                    OutPoint(wp, "Min point", "Box.Min");
                    OutPoint(wp, "Max point", "Box.Max");
                    break;
                case 3: // Отливка
                    nc.OutWithN("(         Type: Casting )");
                    nc.OutWithN("(         Stock: " + wp.Flt["Stock"] + " )");
                    OutPoint(wp, "Min point", "Box.Min");
```

```
                    OutPoint(wp, "Max point", "Box.Max");
                    break;
                case 4: // Параллелепипед
                    nc.OutWithN("(         Type: Box )");
                    OutPoint(wp, "Min point", "Min");
                    OutPoint(wp, "Max point", "Max");
                    break;
                case 5: // Тело вращения
                    nc.OutWithN("(         Type: Turn envelope )");
                    OutPoint(wp, "Origin", "Origin");
                    OutPoint(wp, "Axis", "Axis");
                    break;
                case 6: // Цилиндр
                    nc.OutWithN("(         Type: Cylinder )");
                    OutPoint(wp, "Origin", "Origin");
                    OutPoint(wp, "Axis", "Axis");
                    nc.OutWithN("(         Min level: " + wp.Flt["HMin"] + " )");
                    nc.OutWithN("(         Max level: " + wp.Flt["HMax"] + " )");
                    nc.OutWithN("(         Outer radius: " + wp.Flt["ROut"] + " )");
                    break;
                case 7: // Труба
                    nc.OutWithN("(         Type: Tube )");
                    OutPoint(wp, "Origin", "Origin");
                    OutPoint(wp, "Axis", "Axis");
                    nc.OutWithN("(         Min level: " + wp.Flt["HMin"] + " )");
                    nc.OutWithN("(         Max level: " + wp.Flt["HMax"] + " )");
                    nc.OutWithN("(         Outer radius: " + wp.Flt["ROut"] + " )");
                    nc.OutWithN("(         Inner radius: " + wp.Flt["RInn"] + " )");
                    break;
                case 8: // Призма
                    nc.OutWithN("(         Type: Prism )");
                    OutPoint(wp, "Origin", "Origin");
                    OutPoint(wp, "Axis", "Axis");
                    nc.OutWithN("(         Min level: " + wp.Flt["HMin"] + " )");
                    nc.OutWithN("(         Max level: " + wp.Flt["HMax"] + " )");
                    break;
                case 9: // Многогранная призма
                    nc.OutWithN("(         Type: Polygonal prism )");
                    OutPoint(wp, "Origin", "Origin");
                    OutPoint(wp, "Axis", "Axis");
                    nc.OutWithN("(         Corner count: " + wp.Int["CornerCount"] +
" )");
                    nc.OutWithN("(         Inscribed radius: " + wp.Flt["RInn"] +
" )");
                    nc.OutWithN("(         Angle around axis: " + wp.Flt["Angle"] +
" )");
```

```
                    nc.OutWithN("(         Min level: " + wp.Flt["HMin"] + " )");
                    nc.OutWithN("(         Max level: " + wp.Flt["HMax"] + " )");
                    break;
                default:
                    nc.OutWithN("(         Type: Something else )");
                    break;
            }
        }
    }
}

// Читает дочерние свойства .X / .Y / .Z вложенного свойства-точки и печатает их
одним комментарием.
private void OutPoint(INamedProperty item, string label, string basePath)
{
    double x = item.Flt[basePath + ".X"];
    double y = item.Flt[basePath + ".Y"];
    double z = item.Flt[basePath + ".Z"];
    nc.OutWithN("(         " + label + ": (" + x + ", " + y + ", " + z + " ) )");
}
```

Сопутствующие ссылки

- Информация о проекте в CLData
- Именованные параметры CLData
- Файлы CLData (оператор CLDFile)
- Функции и операторы доступа к CLData

Массив GMA

Массив GMA — это удобство sppx для обработки многоосевых перемещений MULTIGOTO: он представляет управляемые координаты станка (с их новыми/предыдущими значениями, флагами вывода и результатами приведения углов) в единой индексируемой структуре.

> Только sppx. Прямого аналога GMA в .NET нет. В .NET-постпроцессоре многоосевое перемещение читается из строго типизированной команды MULTIGOTO (`ICLDMultiGotoCommand`) — её коллекция `Axes` даёт каждую ось и значение — а работа с регистрами, которую GMA выполняет в sppx, делается через API регистров .NET. См. раздел Доступ из .NET команды MULTIGOTO.

Доступ из sppx

Используйте предопределённый массив GMA для удобной обработки команды многоосевого перемещения MULTIGOTO в технологической программе и в шаблоне команды. Массив GMA инициализируется непосредственно перед обработкой технологической команды MULTIGOTO.

Каждый элемент массива GMA соответствует управляемой координате схемы станка. Элементы массива GMA могут добавляться вручную через интерфейс генератора постпроцессоров либо импортироваться напрямую из схемы станка CAM-системы.

При необходимости координаты можно добавлять в массив GMA во время выполнения постпроцессора с помощью следующего синтаксиса оператора:

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

Здесь AxisName — допустимое выражение, возвращающее строку. Имя новой координаты будет равно AxisName, используйте его для доступа к новому элементу. Необязательный параметр RegisterName — имя регистра, связанного с новой координатой.

Пример

```
GMA("AxisXPos")
```

Используйте имя координаты для доступа к её свойствам GMA:

```
GMA[<AxisName>].<Property>
```

Или для шаблонов:

```
<Axisname>.<Property>
```

Здесь AxisName — допустимое строковое выражение, которое должно быть корректным именем координаты. В следующем примере имя координаты AxisXPos используется для доступа к текущему значению координаты:

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

Структура GMA

Элемент массива GMA имеет следующую структуру:

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

- `GMA[<AxisName>].OutFlag` — поле типа Integer, которому присваиваются значения 1 или 0. Используйте этот флаг для определения того, указана ли координата с именем AxisName в данной команде MULTIGOTO. `GMA[<AxisName>].OutFlag` имеет значение 1 (флаг установлен), если в данной команде MULTIGOTO есть координата с именем AxisName, иначе флаг равен 0.
- `GMA[<AxisName>].Vn` — поле типа Real, значение которого — новое значение координаты AxisName, установленное командой MULTIGOTO. Используйте поле Vn для чтения или задания нового значения координаты.
- `GMA[<AxisName>].Vp` — поле типа Real, содержащее значение координаты AxisName, назначенное предыдущей командой MULTIGOTO. Используйте поле Vp для чтения или задания предыдущего значения координаты.
- `GMA[<AxisName>].Axis` — поле типа String, являющееся именем координаты и равное AxisName. Поле доступно только для чтения.
- `GMA[<AxisName>].Reg` — поле типа String, значение которого — имя регистра, связанного с координатой. Поле доступно только для чтения.
- `GMA[<AxisName>].TurnCount` — поле типа Integer, которому присваивается число полных оборотов поворотной оси. Поле доступно только для чтения.
- `GMA[<AxisName>].Dir` — значение типа Integer, которому присваивается +1, если значение координаты возрастает, иначе устанавливается -1. Поле доступно только для чтения.

Используйте поля TurnCount и Dir для обработки перемещения поворотных осей. CAM-система передаёт перемещения поворотных осей в абсолютных значениях, тогда как некоторые системы ЧПУ принимают для этих координат только значения в ограниченных диапазонах, например от 0° до 360° или от 180° до -180°. Используйте диалог Параметры осей станка для задания параметров границ и диапазонов поворотных осей. Если для данной координаты определено приведение углов, то при обработке команды MULTIGOTO система приведёт значение координаты Vn к заданному диапазону, присвоит полю TurnCount число полных оборотов координаты и установит знак поля Dir в зависимости от предыдущего (не приведённого) значения координаты.

Ниже несколько примеров, иллюстрирующих, как система инициализирует свойства элементов массива GMA при обработке технологической команды MULTIGOTO. Пусть поворотная ось C (AxisCPos) привязана к значениям от 0° до 360°.

1. Команда:

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

Массив GMA для команды MULTIGOTO:

Имя координаты	OutFlag	Vn	Vp	Axis	Reg	TurnCount	Dir
AxisXPos	1	80	-//-//-	"AxisXPos"	"X"	-//-//-	-//-//-
AxisYPos	1	19.877	-//-//-	"AxisYPos"	"Y"	-//-//-	-//-//-
AxisZPos	1	-200	-//-//-	"AxisZPos"	"Z"	-//-//-	-//-//-
AxisCPos	1	0	-//-//-	"AxisCPos"	"C"	-//-//-	-//-//-

Знак "-//-//-" означает, что значения свойств координат не могут быть определены, поскольку нет информации о предыдущей команде.

1. Команда:

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

Массив GMA для команды MULTIGOTO:

Имя координаты	OutFlag	Vn	Vp	Axis	Reg	TurnCount	Dir
AxisXPos	0	80	80	"AxisXPos"	"X"	-//-//-	-//-//-
AxisYPos	0	19.877	19.877	"AxisYPos"	"Y"	-//-//-	-//-//-
AxisZPos	0	-200	-200	"AxisZPos"	"Z"	-//-//-	-//-//-
AxisCPos	1	270	0	"AxisCPos"	"C"	0	+1

1. Команда:

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

Массив GMA для команды MULTIGOTO:

Имя координаты	OutFlag	Vn	Vp	Axis	Reg	TurnCount	Dir
AxisXPos	0	80	80	"AxisXPos"	"X"	-//-//-	-//-//-
AxisYPos	0	19.877	19.877	"AxisYPos"	"Y"	-//-//-	-//-//-
AxisZPos	1	250	-200	"AxisZPos"	"Z"	-//-//-	-//-//-
AxisCPos	1	335	270	"AxisCPos"	"C"	2	-1

В последнем примере система применила приведение углов для координаты `AxisCPos(C)`. Значение, переданное CAM-системой, "-745 °", приводится к диапазону от 0° до 360° по следующим правилам:

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

`Vn'` — текущее не приведённое значение координаты, `Vp'` — предыдущее не приведённое значение координаты.

Ниже приведён пример простой программы команды MULTIGOTO, использующей массив GMA:

```
program MultiGoto
  ! Используйте поля GMA, чтобы присвоить
  ! соответствующим регистрам их новые значения
  if GMA["AxisXPos"].OutFlag>0 then
    X = GMA["AxisXPos"].Vn
  if GMA["AxisYPos"].OutFlag>0 then
    Y = GMA["AxisYPos"].Vn
  if GMA["AxisZPos"].OutFlag>0 then
    Z = GMA["AxisZPos"].Vn
  if GMA["AxisCPos"].OutFlag>0 then
    C = GMA["AxisCPos"].Vn
  ! Вывод сформированных значений в управляющую программу
```

```
  OutBlock
end
```

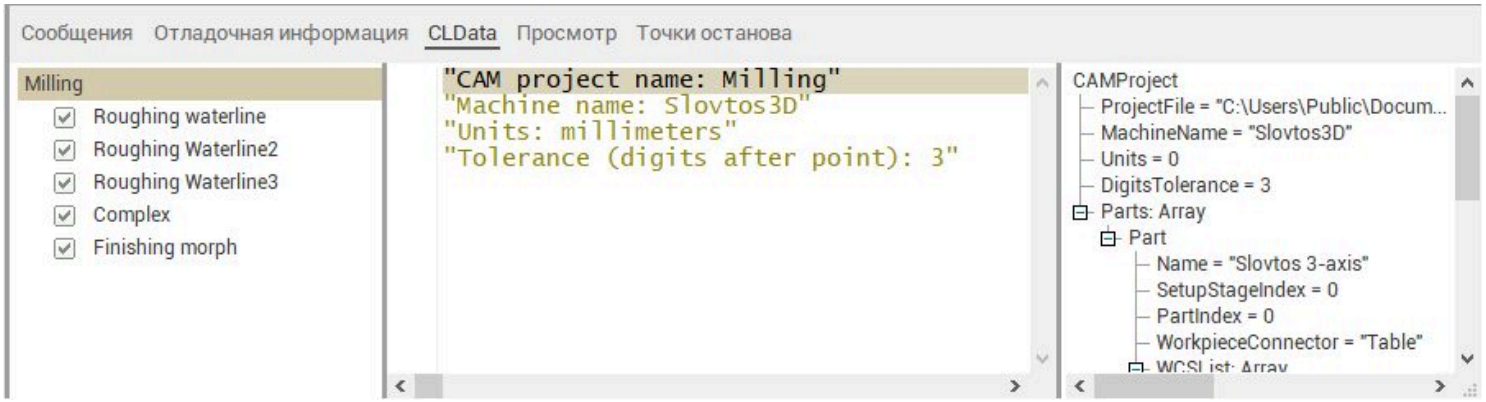
Сопутствующие ссылки

- Предопределённые функции
- Команда MULTIGOTO
- Шаблоны
- Функции и операторы доступа к CLData

Информация о проекте в CLData

Информация о проекте CAM-системы, помещённая внутрь CLData, доступна через оператор Project.

Её можно отобразить на экране, если на вкладке CLData выбрать первый элемент с именем проекта. Свойства будут показаны в дереве параметров справа.



Список возможных свойств описан в таблицах ниже. Информация о структуре свойств, представленная в виде XML-файла, доступна в папке установки CAM-системы в подпапке Supplement\Operations\CLData.xml.

Тип	Описание	Описание вложенного поля
Equipment: ComplexType	"Equipment" — ключевое слово, дающее доступ к свойствам станка, загруженным из файла схемы станка CAM-системы. Список свойств внутри этого узла может различаться в зависимости от файла схемы станка.	
	Schemas: Array	Equipment.Ptr["Schemas"] — список схем станка, используемых в проекте (обычно только одна схема).
MachineSchema: ComplexType	Equipment.Ptr["Schemas(1)"] — корневой элемент узлов свойств схемы станка. Обычно имя этого свойства	

Тип	Описание	Описание вложенного поля
	однозначно идентифицирует схему станка: Equipment.Ptr["Schemas(1)"].Name — уникальный идентификатор схемы станка.	
Name: String	Equipment.Str["Schemas(1).Name"] — понятное пользователю имя станка.	
ProjectFile: String	Project.Str["ProjectFile"] — полный путь к имени файла проекта CAM-системы. Может быть пустым для новых проектов, когда они не сохранены.	
MachineName: String	Project.Str["MachineName"] — имя станка из файла схемы станка.	
Units: Integer	Project.Int["Units"] — система единиц проекта: 0 — метрическая (мм), 1 — дюймовая (дюйм).	
DigitsTolerance: Integer	Project.Int["DigitsTolerance"] — число знаков для точности проекта. Типичное значение — 3 (что означает 0.001). Его можно изменить в окне настроек CAM-системы.	
SetupStages: Array	Project.Ptr["SetupStages"] — список групп установочных позиций.	
	SetupStage: ComplexType	Project.Ptr["SetupStages"].Item[index] — информация о

Тип	Описание	Описание вложенного поля
		группе установочных позиций.
Index: Integer	Project.Ptr["SetupStages"].Item[index].Int["Index"] — индекс группы установочных позиций в списке установочных позиций проекта CAM-системы.	
Name: String	Project.Ptr["SetupStages"].Item[index].Str["Name"] — имя группы установочных позиций в проекте CAM-системы.	
Parts: Array	Project.Ptr["Parts"] — список структур, содержащих информацию о детали. Количество элементов зависит от количества деталей в проекте.	
Part: ComplexType	Project.Ptr["Parts"].Item[index] — структура, содержащая информацию об одной детали.	
Name: String	Project.Ptr["Parts"].Item[index].Str["Name"] — имя детали (имя группы), заданное в CAM-системе.	
SetupStageIndex: Integer	Project.Ptr["Parts"].Item[index].Int["SetupStageIndex"] — индекс группы установочных позиций в списке установочных позиций проекта CAM-системы,	

Тип	Описание	Описание вложенного поля
	внутри которой находится деталь.	
PartIndex: Integer	Project.Ptr["Parts"].Item[index].Int["PartIndex"] — индекс детали в списке деталей проекта CAM-системы	
PartNumber: Integer	Project.Ptr["Parts"].Item[index].Int["PartNumber"] — номер детали, явно заданный пользователем в параметрах детали CAM-системы	
WorkpieceConnector: String	Project.Ptr["Parts"].Item[index].Str["WorkpieceConnector"] — текстовый идентификатор державки заготовки, в которой закреплена деталь. Этот идентификатор зависит от id узла державки заготовки внутри файла схемы станка.	
WCSList: Array	Project.Ptr["Parts"].Item[index].Ptr["WCSList"] — список систем координат заготовки, используемых при обработке этой детали (например, G54, G55 и т. д.).	
WCS: ComplexType	Структура для описания системы координат заготовки	
Number: Integer	Номер системы координат заготовки	
Location: ComplexType	Геометрические параметры WCS	

Тип	Описание	Описание вложенного поля
vX: Complex type	Вектор оси X системы WCS в базовой системе координат	
X: Double	Координата X вектора	
Y: Double	Координата Y вектора	
Z: Double	Координата Z вектора	
vY: Complex type	Вектор оси Y системы WCS в базовой системе координат	
X: Double	Координата X вектора	
Y: Double	Координата Y вектора	
Z: Double	Координата Z вектора	
vZ: Complex type	Вектор оси Z системы WCS в базовой системе координат	
X: Double	Координата X вектора	
Y: Double	Координата Y вектора	
Z: Double	Координата Z вектора	
vT: Complex type	Начало координат WCS в базовой системе координат	
X: Double	Координата X точки	
Y: Double	Координата Y точки	
Z: Double	Координата Z точки	
PartBox: ComplexType	Project.Ptr["Parts(1).PartBox"] — структура для описания габаритных размеров детали.	
Empty: Boolean	Состояние размеров детали: false (0) — размеры заданы и	

Тип	Описание	Описание вложенного поля
	имеют корректные значения, true (1) — размеры пусты (например, когда геометрия детали отсутствует).	
Min: ComplexType	Левая нижняя угловая точка габаритного параллелепипеда детали.	
X: Double	Координата X точки	
Y: Double	Координата Y точки	
Z: Double	Координата Z точки	
Max: ComplexType	Правая верхняя угловая точка габаритного параллелепипеда детали.	
X: Double	Координата X точки	
Y: Double	Координата Y точки	
Z: Double	Координата Z точки	
WorkpieceBox: ComplexType	Project.Ptr["Parts(1).WorkpieceBox"] — структура для описания габаритных размеров заготовки.	
Empty: Boolean	Состояние размеров заготовки: false (0) — размеры заданы и имеют корректные значения, true (1) — размеры пусты (например, когда геометрия заготовки отсутствует).	
Min: ComplexType	Левая нижняя угловая точка габаритного параллелепипеда заготовки.	

Тип	Описание	Описание вложенного поля
X: Double	Координата X точки	
Y: Double	Координата Y точки	
Z: Double	Координата Z точки	
Max: ComplexType	Правая верхняя угловая точка габаритного параллелепипеда заготовки.	
X: Double	Координата X точки	
Y: Double	Координата Y точки	
Z: Double	Координата Z точки	
Workpiece: Array	Project.Ptr["Parts"].Item[index].Ptr["Workpiece"] — список структур, содержащих информацию об элементах заготовки, соответствующих детали с заданным индексом.	
Item: TCLDModelItem	Project.Ptr["Parts"].Item[index].Ptr["Workpiece"].Item[index2] — структура, содержащая информацию об одном элементе заготовки. Состав свойств зависит от типа элемента. Все они наследуются от сложного типа TCLDModelItem, поэтому у всех элементов есть одно общее свойство — тип примитива. См. таблицу Возможные элементы заготовки ниже.	
PrimitiveType: Integer	Project.Ptr["Parts"].Item[index].Ptr["Workpiece"].Item[index2].Int[	

Тип	Описание	Описание вложенного поля
	"PrimitiveType"] — тип геометрического примитива: 0 — Неизвестно, 1 — Пустая заготовка 2 — Грани (ссылка на CAD-модель) 3 — Отливка 4 — Параллелепипед 5 — Тело вращения 6 — Цилиндр 7 — Труба 8 — Призма 9 — Многогранная призма	

Возможные элементы заготовки

Тип	Описание			
EmptyWorkpiece: TCLDModelItem	Элемент заготовки, означающий, что обработка начинается с пустоты. Например, если деталь выращивается с нуля методами аддитивной обработки			
Box: TCLDModelItem	Элемент заготовки, представляющий параллелепипед, заданный двумя точками.			
Min: ComplexType	Левая нижняя угловая точка параллелепипеда.			

Тип	Описание			
X: Double	Координата X точки			
Y: Double	Координата Y точки			
Z: Double	Координата Z точки			
Max: ComplexType	Правая верхняя угловая точка параллелепипеда			
X: Double	Координата X точки			
Y: Double	Координата Y точки			
Z: Double	Координата Z точки			
RevBody: TCLDModelItem	Элемент заготовки, представляющий тело вращения, заданное осью, вокруг которой выполняется вращение, и набором вращающихся геометрических примитивов.			
Origin: ComplexType	Начальная точка оси вращения			

Тип	Описание			
X: Double	Координата X точки			
Y: Double	Координата Y точки			
Z: Double	Координата Z точки			
Axis: ComplexType	Вектор, представляющий ось вращения			
X: Double	Координата X вектора			
Y: Double	Координата Y вектора			
Z: Double	Координата Z вектора			
Cylinder: RevBody	Элемент заготовки, представляющий цилиндр вдоль заданной оси. Является наследником типа RevBody, поэтому содержит все его свойства, описанные выше.			
HMin: Double	Начальный уровень цилиндра. Это расстояние вдоль оси,			

Тип	Описание			
	отсчитываемое от начала координат.			
HMax: Double	Конечный уровень цилиндра. Это расстояние вдоль оси, отсчитываемое от начала координат.			
ROut: Double	Внешний радиус цилиндра.			
Tube: Cylinder	Элемент заготовки, представляющий трубу вдоль заданной оси. Является наследником типа Cylinder, поэтому содержит все его свойства, описанные выше.			
RInn: Double	Внутренний радиус (радиус отверстия) трубы.			
Prism: TCLDModelItem	Элемент заготовки, представляющий призму (элемент			

Тип	Описание			
	выдавливания), заданную осью, вдоль которой выполняется выдавливание, и профилем для выдавливания.			
Origin: ComplexType	Начальная точка оси выдавливания			
X: Double	Координата X точки			
Y: Double	Координата Y точки			
Z: Double	Координата Z точки			
Axis: ComplexType	Вектор, задающий направление выдавливания			
X: Double	Координата X вектора			
Y: Double	Координата Y вектора			
Z: Double	Координата Z вектора			
HMin: Double	Начальный уровень призмы. Это расстояние вдоль оси, отсчитываемое			

Тип	Описание			
	от начала координат.			
HMax: Double	Конечный уровень призмы. Это расстояние вдоль оси, отсчитываемое от начала координат.			
PolygonalPrism: Prism	Элемент заготовки, представляющий правильную многогранную призму вдоль заданной оси. Является наследником типа Prism, поэтому содержит все его свойства, описанные выше.			
RInn: Double	Радиус вписанной окружности многоугольника			
Angle: Double	Угол поворота многоугольника вокруг оси			
CornerCount: Integer	Число углов многоугольника			

Тип	Описание			
Faces: TCLDModelItem	Элемент заготовки, представляющий тело общей формы, заданное ссылкой на CAD-модель.			
Box: ComplexType	Параллелепипед вокруг CAD-модели. Позволяет узнать габаритные размеры.			
Min: ComplexType	Левая нижняя угловая точка параллелепипеда.			
X: Double	Координата X точки			
Y: Double	Координата Y точки			
Z: Double	Координата Z точки			
Max: ComplexType	Правая верхняя угловая точка параллелепипеда			
X: Double	Координата X точки			

Тип	Описание			
Y: Double	Координата Y точки			
Z: Double	Координата Z точки			
Casting: TCLDModelItem	Элемент заготовки, представляющий тело, получаемое добавлением дополнительного припуска к телу общей формы, заданному ссылкой на CAD-модель.			
	Box: ComplexType	Параллелепипед вокруг CAD-модели с припуском. Позволяет узнать габаритные размеры.		
		Min: ComplexType	Левая нижняя угловая точка параллелепипеда.	
			X: Double	Координата X точки
			Y: Double	Координата Y точки

Тип	Описание			
			Z: Double	Координата Z точки
		Max: ComplexType	Правая верхняя угловая точка параллелепипеда	
			X: Double	Координата X точки
			Y: Double	Координата Y точки
			Z: Double	Координата Z точки
	Stock: Double	Величина припуска.		

Сопутствующие ссылки

- CLData

Описание технологических команд

Список технологических команд CLData:

Команда	Код	Описание
AXESBRAKE	2012	Управление тормозами осей станка
CIRCLE	15000	Движение по окружности
CLAMP	1074	Зажим/разжим приспособления
COMMENT	1065	Комментарии
COOLNT	1030	Охлаждение
CUTCOM	1007	Коррекция инструмента
CYCLE	1054	Стандартные циклы обработки отверстий
DELAY	1010	Выдержка времени
EDMMOVE	16001	Перемещение при электроэрозионной обработке
EFFECTOR	2019	Активация рабочего органа (EFFECTOR)
EXTCYCLE	1053	Расширенный цикл
FEDRAT	1009	Подача
FINI	14000	Завершающая запись
FROM	5003	Исходная точка
GOHOME	17	Возврат в исходное положение
GOTO.abs	5005	Линейный переход

Команда	Код	Описание
HEAD	1002	Номер шпиндельной головки
INSERT	1046	Вставка
INTERPOLATION	1047	Режим интерполяции
LOADTL	1055	Загрузка инструмента
MULTIARC	9014	Многоосевое движение по окружности
MULTIGOTO	9010	Многокоординатное перемещение
OPSKIP	1012	Условный пропуск
OPSTOP	2003	Дополнительный останов
ORIGIN	1027	Исходные координаты
PALETA	1001	Смена палеты
PARTNO	1045	Номер детали
PhysicGOTO	9013	Перемещение физических осей станка
PLANE	99	Рабочая плоскость
POWER	2018	Регулировка мощности (POWER)
PPFUN	1079	Функция постпроцессора
PPRINT	1044	Печать постпроцессора
RAPID	2005	Ускоренная подача
ROTABL	1026	Поворот стола
SAFPOS	1094	Точка смены инструмента

Команда	Код	Описание
SELWORKPIECE	2011	Выбор активной державки заготовки
SINGLETHREAD	1037	Команда однопоточности
SPINDL	1031	Шпиндель
STOP	2002	Останов
STRUCTURE	2015	Команда структуры
SYNCAXES	2016	Синхронизация движения осей
TAKEOVER	2010	Перехват заготовки
TLCONTACT	2017	Команда контакта инструмента
WAIT	2014	Ожидание точки синхронизации

Дополнительные слова:

N	Слово	Код	Описание
1	BOTH	83	Оба
2	BRKCHP	288	Дробление стружки
3	CCLW	59	Против часовой стрелки
4	CLW	60	По часовой стрелке
5	CUTS	511	Проходы
6	DEEP	153	Глубокое сверление
7	DEPTH	510	Глубина
8	DRILL	163	Сверление

N	Слово	Код	Описание
9	DWELL	279	Выдержка
10	FACE	81	Стандартный цикл G82
11	FINCUT	512	Финишный проход
12	INCR	66	Приращение
13	LENGTH	9	Длина
14	MMPM	315	Миллиметров в минуту
15	MMPR	316	Миллиметров за оборот
16	MULTRD	119	Многозаходная резьба
17	OFF	72	Выключить
18	ON	71	Включить
19	ORIENT	246	Ориентация
20	TAP	168	Нарезание резьбы метчиком
21	XYPLAN	33	Плоскость XY
22	YZPLAN	37	Плоскость YZ
23	ZXPLAN	41	Плоскость ZX
24	R	23	Радиус
25	RGT	24	Позиционирование инструмента справа
26	LEFT	8	Позиционирование инструмента слева

Глобальные обработчики (не команды CLData)

Обработчики, вызываемые вне поочерёдной диспетчеризации команд, - перечислены здесь, чтобы все обработчики постпроцессора можно было найти в одном месте:

Обработчик	sppx	.NET	Описание
Фильтр вывода	`sub Filter`	`OnFilterString`	Выполняется для каждой строки управляющей программы, независимо от того, чем она была сформирована.

Сопутствующие ссылки

- CLData

AXESBRAKE - Управление тормозами осей станка

Отображение команды

Как команда отображается в списке команд CLData:

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

Команда AXESBRAKE используется для включения и выключения тормозов осей, если на станке есть тормоза осей. Чаще всего эта команда используется для включения/выключения тормоза шпинделя (ось C) на токарно-фрезерных станках при переключении с точения на фрезерование и обратно. При токарной обработке шпиндель вращается на высокой скорости, а при фрезеровании шпиндель фиксируется тормозом в определённом угловом положении. Использование тормозов позволяет увеличить нагрузку на шпиндель и снизить износ механизмов точного позиционирования шпинделя. Также тормоза используются на фрезерных станках на поворотных осях для выполнения обработки с делением (индексацией).

Доступ из sppx

Обработчик: `program AxesBrake`

Параметр	Массив CLD	Описание
N	CLD[1]	Число осей, затрагиваемых этой командой.
n1	CLD[2]	Индекс оси с именем Axis1Pos в списке координат.
State1	CLD[3]	Новое состояние оси Axis1Pos: ON(71) - тормоз включён, OFF(72) - тормоз выключен.
n2	CLD[4]	Индекс оси Axis2Pos в списке координат.
State2	CLD[5]	Новое состояние оси Axis2Pos: ON(71) - тормоз

Параметр	Массив CLD	Описание
		включён, OFF(72) - тормоз выключен.
...	...	...
nN	CLD[2*N]	Индекс оси AxisNPos в списке координат.
StateN	CLD[2*N+1]	Новое состояние оси AxisNPos: ON(71) - тормоз включён, OFF(72) - тормоз выключен.

Параметры, доступные через оператор cmd

TCLDAxesBrake: ComplexType	Команда переключения состояний тормозов осей.
Axes: Array, Key="AxisID"	cmd.Ptr["Axes"] - Массив структур типа AxisBrake. Таким образом, одна команда может менять состояние тормоза сразу нескольких осей.
AxisBrake: ComplexType	cmd.Ptr["Axes"].Item[Index] или cmd.Ptr["Axes(AxisName)"] - Отдельный элемент массива Axes. Содержит состояние тормоза одной оси станка. Доступ к элементам массива возможен как по индексу, так и по ключевому полю. Здесь AxisName — значение ключевого поля, которое должно совпадать со значением поля AxisID.
AxisID: String	cmd.Str["Axes(AxisName).AxisID"] - Идентификатор оси станка, для которой задано новое состояние тормоза. Определяется схемой станка.
BrakeState: Integer	cmd.Int["Axes(AxisName).BrakeState"] - новое состояние тормоза оси станка: 0 - выключен, 1 - включён.

Синтаксис команды допускает изменение состояний тормозов нескольких осей одновременно. Список координат, имена которых фигурируют в этой команде, определяется схемой станка CAM-системы.

Ниже приведены два простых примера обработчиков-программ для этой команды.

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

Ещё один пример с использованием оператора cmd.

```
program AxesBrake
  if cmd.Ptr["Axes(AxisAPos)"]<>0 then begin ! Ось A присутствует в этой команде
    if cmd.Int["Axes(AxisAPos).BrakeState"]=1 then Output "M680" ! Включить тормоз
оси A
    else Output "M690" ! Выключить тормоз оси A
  end
  if cmd.Ptr["Axes(AxisCPos)"]<>0 then begin ! Ось C присутствует в этой команде
    if cmd.Int["Axes(AxisCPos).BrakeState"]=1 then Output "M68" ! Включить тормоз
оси C
    else Output "M69" ! Выключить тормоз оси C
  end
end
```

Доступ из .NET

Обработчик:

```
public override void OnAxesBrake(ICLDAxesBrakeCommand cmd, CLDArray cld)
```

Наследование: `ICLDAxesBrakeCommand : ICLDCommand`.

Читайте параметры из строго типизированного объекта `cmd` (рекомендуется); доступны также «сырой» массив `cld[i]` (те же индексы, что и в sppx) и именованные обёртки `cmd["..."]`. В таблице перечислены члены команды, включая унаследованные от базовых интерфейсов; общие члены `ICLDCommand` (навигация, `CLD`, `CLDFile`, `TechOperation`, ...) описаны в разделе Модель доступа к CLData.

Член	Тип	Описание
`cmd.AxesCount`	`int`	Число осей в списке "Axis".
`cmd.Axis`	`ICLDBrakeAxesIndexer`	Список состояний тормозов осей Axis[i] или Axis["AxisCPos"].
`cmd.Axes`	`IEnumerable`	Объект-перечислитель осей тормозов, позволяющий использовать удобный синтаксис foreach.
`cmd.HasAxis(...)`	`bool`	Возвращает "True", если список "Axis" содержит ось с указанным текстовым ID. Иначе возвращает "False".

Пример - `OnAxesBrake` (из постпроцессора Fanuc (30i)_Mill_DN):

```
public override void OnAxesBrake(ICLDAxesBrakeCommand cmd, CLDArray cld)
{
    foreach(CLDAxisBrake ax in cmd.Axes) {
        if (ax.IsA) {
            if (ax.StateIsOn)
                nc.MABrake.v = 593;
            else
                nc.MABrake.v = 592;
        } else if (ax.IsB) {
```

```
            if (ax.StateIsOn)
                nc.MBBrake.v = 595;
            else
                nc.MBBrake.v = 594;
        } else if (ax.IsC) {
            if (ax.StateIsOn)
                nc.MCBrake.v = 597;
            else
                nc.MCBrake.v = 596;
        }
    }
    if (nc.MABrake.Changed || nc.MBBrake.Changed || nc.MCBrake.Changed)
        nc.Block.Out();
}
```

Сопутствующие ссылки

- Модель доступа к CLData

CIRCLE - Движение по окружности

Отображение команды

Как команда отображается в списке команд CLData:

```
CIRCLE XC xc, YC yc, ZC zc, R r, XE xe, YE ye, ZE ze
```

Команда CIRCLE осуществляет перемещение по дуге окружности. Параметры CIRCLE - это центр дуги, конечная точка дуги и радиус дуги. Знак радиуса определяет направление дуги. Если радиус положительный, дуга направлена против часовой стрелки (круговая интерполяция G03). Если радиус отрицательный, дуга направлена по часовой стрелке (круговая интерполяция G02). Плоскость дуги (G17/G18/G19) определяется текущей плоскостью обработки, заданной командой PLANE.

Доступ из sppx

Обработчик: `program Circle`

Массив `cld` читается по индексу или по имени - в sppx по типизированному короткому имени (например, `cld.X`), в .NET по тому же имени как строковому ключу (например, `cld["X"]`).

Параметр	Массив CLD	Имя CLD	Описание
xc yc zc	CLD[1] CLD[2] CLD[3]	CLD.Xc CLD.Yc CLD.Zc	Координаты X, Y и Z центра дуги.
r	CLD[4]	CLD.R	Радиус дуги. Если CLD.R > 0, дуга направлена против часовой стрелки (G03), если CLD.R < 0 - по часовой стрелке (G02).
xe ye ze	CLD[5] CLD[6] CLD[7]	CLD.Xe CLD.Ye CLD.Ze	Координаты X, Y и Z конечной точки дуги.
xs ys	CLD[8] CLD[9]	CLD.Xs CLD.Ys	Координаты X, Y и Z начальной точки дуги

Параметр	Массив CLD	Имя CLD	Описание
zs	CLD[10]	CLD.Zs	
XcInc YcInc ZcInc	CLD[11] CLD[12] CLD[13]	CLD.XcInc CLD.YcInc CLD.ZcInc	Координаты X, Y и Z центра дуги относительно начальной точки (I, J и K)
RAbs	CLD[14]	CLD.RAbs	Абсолютное значение радиуса дуги
RIso	CLD[15]	CLD.RIso	Радиус дуги в варианте ISO - положительный, если угол дуги меньше 180 градусов, и отрицательный в противном случае
Dir	CLD[16]	CLD.Dir	Направление дуги: 2 - по часовой стрелке, 3 - против часовой стрелки относительно плоскости ISO G17-G19
Plane	CLD[17]	CLD.Plane	Плоскость дуги, может принимать 6 значений: 17, 18, 19, -17, -18, -19. Соответствует плоскостям ISO G17-G19. Отрицательное значение означает, что нормаль плоскости дуги обратна относительно

Параметр	Массив CLD	Имя CLD	Описание
			стандартной плоскости ISO.
Length	CLD[18]	CLD.Length	Длина дуги в текущих линейных единицах
Ang	CLD[19]	CLD.Ang	Угол дуги в градусах (от 0 до 360 градусов)
SAng	CLD[20]	CLD.SAng	Начальный угол дуги относительно первой оси текущей плоскости
EAng	CLD[21]	CLD.EAng	Конечный угол дуги относительно первой оси текущей плоскости
HelixAng	CLD[22]	CLD.HelixAng	Угол наклона спирали в градусах. Ноль для плоской окружности, положительное значение для поднимающейся спирали, отрицательное - для опускающейся спирали.

Пример (из постпроцессора Fanuc (30i)_Mill):

```
program Circle
  RSign: Integer

  if cld[4] > 0 then INTERP_ = 3 else INTERP_ = 2   ! G3/G2
  X = cld[5]                                        ! X,Y,Z в абсолютных значениях
  Y = cld[6]
```

```
  Z = cld[7]
  if CirclesThroughRadius>0 then begin
    call CheckCircleAngle(RSign)
    Rc = RSign*abs(CLD.R); Rc@ = MaxReal
  end else begin
    XC_ = cld[1] - XT_                          ! I,J,K всегда в приращениях
    YC_ = cld[2] - YT_
    ZC_ = cld[3] - ZT_
    XC_@ = MaxReal; ZC_@ = MaxReal
    if OperationIsLathe<>1 then
      YC_@ = MaxReal
    case Plane of
      17: ZC_@ = ZC_;
      18: YC_@ = YC_;
      19: XC_@ = XC_;
    end
  end
  case Plane of
    17: begin
      X@ = MaxReal;
      Y@ = MaxReal
    end
    18: begin
      X@ = MaxReal;
      Z@ = MaxReal
    end
    19: begin
      Y@ = MaxReal
      Z@ = MaxReal
    end
  end
  if IsRotaryInterpolation = 2 then begin
    Y = -Y/CylR/Pi*180
  end
  OutBlock                      ! вывод кадра УП
  XT_ = cld[5]                  ! сохранить текущие координаты
  YT_ = cld[6]
  ZT_ = cld[7]
end
```

Доступ из .NET

Обработчики (конкретный обработчик плюс скобки группы перемещений, которые срабатывают для любой команды движения):

```
public override void OnCircle(ICLDCircleCommand cmd, CLDArray cld)
public override void OnBeforeMovement(ICLDMotionCommand cmd, CLDArray cld)   //
перед любым перемещением
public override void OnAfterMovement(ICLDMotionCommand cmd, CLDArray cld)    //
после любого перемещения
```

Наследование: `ICLDCircleCommand : ICLDMotionCommand : ICLDCommand`.

Читайте параметры из строго типизированного объекта `cmd` (рекомендуется); также доступны «сырой» массив `cld[i]` (те же индексы, что и в sppx) и именованные помощники `cmd[...]`. В таблице перечислены члены команды, включая унаследованные от базовых интерфейсов; общие члены `ICLDCommand` (навигация, `CLD`, `CLDFile`, `TechOperation`, ...) описаны в разделе Модель доступа к CLData.

Свойство	Тип	Описание
`cmd.EP`	`TInp3DPoint`	Конечная точка (X, Y, Z) перемещения. *(унаследовано от `ICLDMotionCommand`)*
`cmd.SP`	`TInp3DPoint`	Начальная точка дуги (X, Y, Z).
`cmd.Center`	`TInp3DPoint`	Центр дуги (X, Y, Z), абсолютное значение.
`cmd.IncCenter`	`TInp3DPoint`	Центр дуги (I, J, K), приращение относительно начальной точки.
`cmd.R`	double	Радиус дуги; положительный для CCW (G03), отрицательный для CW (G02).
`cmd.RAbs`	double	Абсолютное значение радиуса дуги.
`cmd.RIso`	double	Радиус в стандарте ISO: положительный, если угол дуги < 180 градусов, отрицательный, если > 180 градусов.

Свойство	Тип	Описание
`cmd.Dir`	int	Код направления ISO: `2` = по часовой стрелке (G2), `3` = против часовой стрелки (G3).
`cmd.Plane`	int	Плоскость дуги: `17, 18, 19, -17, -18, -19` (ISO G17-G19; отрицательное значение - обратная нормаль).
`cmd.Length`	double	Длина дуги в текущих линейных единицах.
`cmd.Ang`	double	Угол дуги в градусах (от 0 до 360).
`cmd.SAng` `cmd.EAng`	double	Начальный / конечный угол дуги относительно первой оси текущей плоскости.
`cmd.HelixAng`	double	Угол наклона спирали в градусах (0 - плоская окружность, >0 - подъём, <0 - спуск).

Пример (из постпроцессора Fanuc 30i mill):

```
public override void OnCircle(ICLDCircleCommand cmd, CLDArray cld)
{
    nc.GInterp.Show(cmd.Dir);                                   // G02 / G03
    nc.X.v = cmd.EP.X;  nc.Y.v = cmd.EP.Y;  nc.Z.v = cmd.EP.Z;
    switch (Abs(cmd.Plane))                                     // центр дуги как
приращение I, J, K
    {
        case 17:
            nc.I.v = cmd.IncCenter.X;  nc.J.v = cmd.IncCenter.Y;
            nc.Block.Show(nc.X, nc.Y, nc.I, nc.J);
            break;
        case 18:
            nc.I.v = cmd.IncCenter.X;  nc.K.v = cmd.IncCenter.Z;
            nc.Block.Show(nc.X, nc.Z, nc.I, nc.K);
            break;
        case 19:
```

```
            nc.J.v = cmd.IncCenter.Y;  nc.K.v = cmd.IncCenter.Z;
            nc.Block.Show(nc.Y, nc.Z, nc.J, nc.K);
            break;
    }
}
```

Сопутствующие ссылки

- Модель доступа к CLData

CLAMP - Зажим приспособлений

Команда CLAMP используется для управления зажимными устройствами детали (патронами, тисками, цанговыми зажимами и т. д.) в операциях типа "взять-положить" (pick-and-place) и передачи заготовки между шпинделями.

Команда: CLAMP ID(N), ON(71)|OFF(72) DIR(N)

Доступ из sppx

Обработчик: `program Clamp`

Параметр	Массив CLD	Имя CLD	Описание
ID	CLD[1]	CLD.ID	Уникальный идентификатор зажимного устройства, которое нужно переключить. Зависит от настроек оборудования внутри проекта CAM.
IsOn	CLD[2]	CLD.IsOn	Новое состояние зажимного устройства: 71 - зажим включён, заготовка зафиксирована в зажимном устройстве. 72 - зажим выключен, заготовка освобождена.
Dir	CLD[3]	CLD.Dir	Направление зажима (-1; 0; +1) - направление перемещения зажимной губки (знак изменения

Параметр	Массив CLD	Имя CLD	Описание
			координаты по оси губки).

Простой пример программ-обработчиков для этой команды.

```
program Clamp
  if CLD[2] = 72 then begin
    call OutPause(1)
    if CLD[1] = 2 then begin
      Mchuck = 111; Mchuck@ = Maxreal
      Formblock
      Outstr$ = Outstr$ + " (OTKRYT-PROTIVOSHPINDEL)"
      Output Outstr$
      Formblock
      Outstr$ = Outstr$ + " M114 (AIR ON)"
      Output Outstr$
      call OutPause(1)
      Formblock
      Outstr$ = Outstr$ + " M115 (AIR OFF)"
      Output Outstr$
    end else begin
      Mchuck = 11; Mchuck@ = Maxreal
      Formblock
      Outstr$ = Outstr$ + " (OTKRYT-GLAVNYi SHPINDEL)"
      Output Outstr$
      Formblock
      Outstr$ = Outstr$ + " M14 (AIR ON)"
      Output Outstr$
      call OutPause(1)
      Formblock
      Outstr$ = Outstr$ + " M15 (AIR OFF)"
      Output Outstr$
    end
  end else begin
    if CLD[1] = 2 then begin
      Mchuck = 110; Mchuck@ = Maxreal
      Formblock
      Outstr$ = Outstr$ + "(ZAKRIT-PROTIVOSHPINDEL)"
    end else begin
      Mchuck = 10; Mchuck@ = Maxreal
      Formblock
      Outstr$ = Outstr$ + "(ZAKRIT-GLAVNIY SHPINDEL)"
    end
    Output Outstr$
```

```
    call OutPause(1)
  end
end
```

Доступ из .NET

Обработчик:

```
public override void OnClamp(ICLDClampCommand cmd, CLDArray cld)
```

Наследование: `ICLDClampCommand : ICLDCommand`.

Читайте параметры из строго типизированного объекта `cmd` (рекомендуется); также доступны «сырой» массив `cld[i]` (те же индексы, что и в sppx) и именованные помощники `cmd["..."]`. В таблице перечислены члены команды, включая унаследованные от базовых интерфейсов; общие члены `ICLDCommand` (навигация, `CLD`, `CLDFile`, `TechOperation`, ...) описаны в разделе Модель доступа к CLData.

Пример - `OnClamp` (из постпроцессора Goodway SW):

```
public override void OnClamp(ICLDClampCommand cmd, CLDArray cld)
{
    base.OnClamp(cmd, cld);
    bool MainSp = cmd.ClampID==1;
    outClamp(nc, MainSp, cmd.IsOn, 0.2);
}
```

Сопутствующие ссылки

- Модель доступа к CLData

COMMENT - Комментарии

Отображение команды

Как команда отображается в списке команд CLData:

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

COMMENT выводит строку комментария в управляющую программу. Комментарий помещается в переменную CLDATA$. Комментарии могут транслитерироваться.

Доступ из sppx

Обработчик: `program Comment`

Пример (из постпроцессора Fanuc (30i)_Mill):

```
program Comment
  if CLDATA$<>"" then begin
    if pos("@", CLDATA$) = 1  then begin
      CLDATA$ = ""!Copy(CLDATA$, 2, Len(CLDATA$))
    end
    if CLDATA$<>"" then
      Output "( "+CLDATA$+" )"
  end
end
```

Доступ из .NET

Обработчик:

```
public override void OnComment(ICLDCommentCommand cmd, CLDArray cld)
```

Наследование: `ICLDCommentCommand : ICLDCommand`.

Читайте параметры из строго типизированного объекта `cmd` (рекомендуется); доступны также «сырой» массив `cld[i]` (те же индексы, что и в sppx) и именованные обёртки `cmd["..."]`. В таблице перечислены члены команды, включая унаследованные от базовых интерфейсов; общие члены `ICLDCommand` (навигация, `CLD`, `CLDFile`, `TechOperation`, ...) описаны в разделе Модель доступа к CLData.

Член	Тип	Описание
`cmd.Text`	`string`	Текстовая строка комментария.
`cmd.UpperCaseText`	`string`	Текстовая строка комментария в верхнем регистре.
`cmd.IsOperationName`	`bool`	Возвращает "true", если эта команда — первая команда COMMENT в файле операции, содержащая имя этой операции. Если это комментарий другого типа, флаг содержит "false".
`cmd.IsToolName`	`bool`	Возвращает "true", если эта команда — специальная команда COMMENT, содержащая имя инструмента. Если это комментарий другого типа, флаг содержит "false".

Пример - `OnComment` (из постпроцессора Fanuc (30i)_Mill_DN):

```
public override void OnComment(ICLDCommentCommand cmd, CLDArray cld)
{
    if (!(cmd.IsOperationName || cmd.IsToolName)) {
        nc.OutWithN("( " + Transliterate(cmd.CLDataS) + " )");
    }
}
```

Сопутствующие ссылки

- Модель доступа к CLData

COOLNT - Охлаждение

Отображение команды

Как команда отображается в списке команд CLData:

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

Команда COOLNT включает или выключает системы охлаждения станка. Первый параметр задаёт тип команды: включение или выключение. Второй параметр - номер трубки, на которую воздействует команда COOLNT.

Доступ из sppx

Обработчик: `program Coolnt`

Параметр	Массив CLD	Имя CLD	Описание
ON(71) OFF(72)	CLD[1]	CLD.OnOff	Тип команды: ON(71) - включить СОЖ, OFF(72) - выключить СОЖ.
n	CLD[2]	CLD.N	Номер трубки СОЖ: 1 - обильная подача, 2 - распыление (туман), 3 - подача через инструмент.

Для создания ISO-постпроцессора с использованием шаблонов используйте значения ISO.M, соответствующие параметрам:

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

Пример (из постпроцессора Fanuc (30i)_Mill):

```
program CoolNt
  if cld[1] = 71 then begin
```

```
    if cld[2] = 1 then begin
      Mc = 8
    end else if cld[2] = 2 then begin
      Mc = 8
    end else if cld[2] = 3 then begin
      Mc = 8
    end else begin
      Mc = 8
    end
  end else begin
    Mc = 9     ! Включение/выключение СОЖ
    OutBlock
  end
end
```

Доступ из .NET

Обработчик:

```
public override void OnCoolant(ICLDCoolantCommand cmd, CLDArray cld)
```

Наследование: `ICLDCoolantCommand : ICLDCommand`.

Читайте параметры из строго типизированного объекта `cmd` (рекомендуется); также доступны «сырой» массив `cld[i]` (те же индексы, что и в sppx) и именованные помощники `cmd["..."]`. В таблице перечислены члены команды, включая унаследованные от базовых интерфейсов; общие члены `ICLDCommand` (навигация, `CLD`, `CLDFile`, `TechOperation`, ...) описаны в разделе Модель доступа к CLData.

Член	Тип	Описание
`cmd.IsOn`	`bool`	Возвращает "True", если это команда включения и нужно включить все трубки, отмеченные в списке TubeIsOn[].
`cmd.IsOff`	`bool`	Возвращает "True", если это команда выключения и нужно выключить все трубки, отмеченные в списке TubeIsOn[].

Член	Тип	Описание
`cmd.TubeCount`	`int`	Возвращает количество трубок, присутствующих в списке TubeIsOn[].
`cmd.TubeIsOn`	`ICLDCoolantTubeIndexer`	Позволяет определить, нужно ли переключать состояние для трубки с заданным индексом. Индекс должен быть в диапазоне 0..31 (TubeCount-1).
`cmd.FirstEnabledTube`	`int`	Возвращает индекс первой трубки, которую нужно переключить, т. е. TubeIsOn[FirstEnabledTube]=true.

Пример - `OnCoolant` (из постпроцессора Fanuc (30i)_Mill_DN):

```
public override void OnCoolant(ICLDCoolantCommand cmd, CLDArray cld)
{
    if (cmd.IsOn) {
        nc.MCoolant.v = 8;
    } else {
        nc.MCoolant.v = 9;
        nc.Block.Out();
    }
}
```

Сопутствующие ссылки

- Модель доступа к CLData

CUTCOM - Компенсация инструмента

Отображение команды

Как команда отображается в списке команд CLData:

```
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)
```

Команда CUTCOM используется для переключения режима компенсации на длину или радиус инструмента.

Доступ из sppx

Обработчик: `program CutCom`

Параметр	Массив CLD	Имя CLD	Описание
ON(71) OFF(72)	CLD[1]	CLD.OnOff	Тип команды: 71 - включение компенсации, 72 - выключение компенсации.
LENGTH(9) R(23)	CLD[2]	CLD.Length	Вид компенсации: 9 - компенсация на длину инструмента, 23 - компенсация на радиус инструмента.
LR	CLD[3]	CLD.LR	Значение компенсации на длину или радиус инструмента.
x y z	CLD[4] CLD[5] CLD[6]	CLD.X CLD.Y CLD.Z	Компенсация по осям.
n m k	CLD[7] CLD[8] CLD[9]	CLD.N CLD.M CLD.K	Значения компенсации в плоскостях.

Параметр	Массив CLD	Имя CLD	Описание
RIGHT(24) LEFT(8)	CLD[10]	CLD.RGT	Направление компенсации на радиус инструмента: 24 - компенсация направо, 8 - компенсация налево.

Для создания ISO-постпроцессора с помощью шаблонов используйте значения ISO.G, которые соответствуют параметрам CUTCOM:

Параметр 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

Пример (из постпроцессора Fanuc (30i)_Mill):

```
program CutCom
  if cld[2]=9 then begin                          ! LENGTH
    X = MaxReal; X@ = X
    Y = MaxReal; Y@ = Y
    Z = MaxReal; Z@ = Z
    if cld[1]=71 then begin ! Включение
      CoordSys = WCSNumber
      OutBlock
      if (NeedKorDl=0) and (OperationIsLathe=0) and
         ((WasLoadTool>0) or (H<>cld[3]))
      then begin
        NeedKorDl = 43
        H = cld[3]; H@ = H
      end
      OutBlock
    end else begin          ! Выключение
      KORDL = 49; !KORDL@ = MaxReal
      H = 0; H@ = H
```

```
      OutBlock

      call SwitchOffLocalCS
    end
  end else if cld[2]=23 then begin              ! RADIUS
    if cld[1]=72 then
      KorEcv = 40
    else begin
      if GFeed<>GFeed@ then OutBlock
      if cld[10]=24 then KorEcv=42 else KorEcv=41
      D = cld[3]
      D@ = MaxReal
    end
  end else begin
  end
end
```

Доступ из .NET

Обработчики:

```
public override void OnCutCom(ICLDCutComCommand cmd, CLDArray cld)
public override void OnLengthCompensation(ICLDCutComCommand cmd, CLDArray cld)   //
компенсация LENGTH(9)
public override void OnRadiusCompensation(ICLDCutComCommand cmd, CLDArray cld)   //
компенсация RADIUS(23)
```

Наследование: `ICLDCutComCommand : ICLDCommand`.

Читайте параметры из строго типизированного объекта `cmd` (рекомендуется); также доступны «сырой» массив `cld[i]` (те же индексы, что и в sppx) и именованные помощники `cmd["..."]`. В таблице перечислены члены команды, включая унаследованные от базовых интерфейсов; общие члены `ICLDCommand` (навигация, `CLD`, `CLDFile`, `TechOperation`, ...) описаны в разделе Модель доступа к CLData.

Член	Тип	Описание
`cmd.IsOn`	`bool`	Возвращает «True», если это команда включения компенсации, иначе возвращает «False».

Член	Тип	Описание
`cmd.IsOff`	`bool`	Возвращает «True», если это команда выключения компенсации, иначе возвращает «False».
`cmd.IsLength`	`bool`	Возвращает «True», если это команда переключения компенсации LENGTH, иначе возвращает «False».
`cmd.IsRadius`	`bool`	Возвращает «True», если это команда переключения компенсации RADIUS, иначе возвращает «False».
`cmd.IsRightDirection`	`bool`	Определяет направление компенсации. «True» означает правое направление, «False» - левое.
`cmd.IsLeftDirection`	`bool`	Определяет направление компенсации. «True» означает левое направление, «False» - правое.
`cmd.CorrectorNumber`	`int`	Номер корректора инструмента в таблице корректоров станка.

Пример - `OnLengthCompensation` (из постпроцессора Fanuc (30i)_Mill_DN):

```
public override void OnLengthCompensation(ICLDCutComCommand cmd, CLDArray cld)
{
    if (cmd.IsOn) {
        nc.GLCompens.Reset(43);
        nc.HLCompens.Reset(cmd.CorrectorNumber);
    } else {
        nc.GLCompens.Show(49);
        nc.HLCompens.Hide(0);
        nc.Block.Out();
```

```
    }
}
```

Пример - `OnRadiusCompensation` (из постпроцессора Fanuc (30i)_Mill_DN):

```
public override void OnRadiusCompensation(ICLDCutComCommand cmd, CLDArray cld)
{
    if (cmd.IsOn) {
        if (cmd.IsLeftDirection)
            nc.GRCompens.Show(41);
        else
            nc.GRCompens.Show(42);
        nc.DRCompens.Show(cmd.CorrectorNumber);
    } else {
        nc.GRCompens.Show(40);
    }
}
```

Сопутствующие ссылки

- Модель доступа к CLData

CYCLE - Циклы обработки отверстий

> Оставлена для совместимости. CYCLE сохранена только для совместимости со старыми постпроцессорами. Текущие версии CAM-системы вместо неё генерируют команду расширенного цикла EXTCYCLE: она предоставляет больше возможностей и позволяет CAM-системе полностью моделировать все вложенные перемещения цикла, чего CYCLE не умеет. Вывод CYCLE всё ещё можно включить в настройках CAM-системы, но это не рекомендуется.

Отображение команды

Как команда отображается в списке команд CLData:

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

Команда CYCLE используется для передачи стандартных циклов сверления G81–G89. С помощью команды CYCLE определите тип цикла и обработайте цикл соответствующим образом. Тип цикла задаётся первым параметром; типы остальных параметров и их количество зависят от типа цикла.

Доступ из sppx

Обработчик: `program Cycle`

Массив `cld` читается по индексу или по имени — в sppx по типизированному короткому имени (например, `cld.Type`), в .NET по тому же имени в виде строкового ключа (например, `cld["Type"]`).

Параметр	Массив CLD	Имя CLD	Описание
Type	CLD[1]	CLD.Type	Тип цикла: 71(ON) – включить цикл, 72(OFF) – выключить цикл, 163(DRILL) - цикл сверления G81, 81(FACE) - цикл сверления G82, 153(DEEP) - цикл глубокого

Параметр	Массив CLD	Имя CLD	Описание
			сверления, 288(BRKCHP) - сверление с ломкой стружки, 168(TAP) - цикл нарезания резьбы G84, 209(BORE5) - цикл сверления G85, 210(BORE6) - цикл сверления G86, 211(BORE7) - цикл сверления G87, 212(BORE8) - цикл сверления G88, 213(BORE9) - цикл сверления G89.

Обычно CAM-система генерирует технологию сверления, подобную следующей:

```
...
GOTO.abs X, Y, Z ! Позиционирование инструмента у первого отверстия.
CYCLE DRILL(163) ! Выполнить сверление вызовом цикла соответствующего типа.
GOTO.abs X, Y, Z ! Переместить инструмент к следующему отверстию.
CYCLE DRILL(163) ! Выполнить сверление вызовом цикла соответствующего типа.
...
GOTO.abs X, Y, Z ! Переместить инструмент к последнему отверстию.
CYCLE DRILL(163) ! Выполнить сверление вызовом цикла соответствующего типа.
CYCLE OFF(72) ! Выключить цикл.
...
```

Первая команда цикла сверления должна формировать кадр УП, содержащий определение цикла (например, G81) и параметры цикла. Например:

```
G81 X_Y_Z_R_F_
```

Обработка последующих команд должна формировать кадр, содержащий только изменившиеся регистры. Например:

```
X_Y_Z_
X_Y_Z_
```

```
...
X_Y_Z_
```

Команда `CYCLE OFF(72)` должна формировать код отмены цикла. Например:

```
G80
```

При разработке ISO-постпроцессоров с использованием шаблонов можно применять оператор ISO.G. Значение ISO.G устанавливается автоматически согласно следующей таблице:

Значение параметра CLD	Значение ISO.G
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

Пример (из постпроцессора Fanuc (30i)_Mill):

```
program Cycle
  if cld[1]=71 then CycleOn = 1 else ! включение цикла
  if cld[1]=72 then CycleOn = 0 else ! выключение цикла
  begin
    CycleOn = 1                      ! включение цикла
    KodCycle = cld[1]
    Za = cld[2]                      ! рабочая глубина
    if cld[4] > 0 then begin         ! формирование значения FEED_
```

```
      if cld[3]=316 then Fr=cld[4]*S else Fr=cld[4]
    end
    Zf = cld[5]                      ! безопасный уровень
    ZP_ = cld[6]                      ! уровень обратного хода
    Dwell = cld[10]*1000              ! время выстоя
    Zl = cld[6]                      ! глубина одного шага сверления
    Zi = cld[8]                     ! переходное значение

  end

  if CycleOn=1 then begin
    !X@=maxreal; Y@ = maxreal;
    !Feed_@=Maxreal;
    Zcycle=Zf-Za
    ZClear=Zf
    p=0;  P@=0
    If KodCycle=153 or KodCycle=288 or KodCycle=211 then Q=Zl else Q=0
    If KodCycle=81 or KodCycle=213 then P=Dwell else P=0
    if KodCycle=163 then Cycle = 81   ! DRILL
    if KodCycle=81  then Cycle = 82   ! FACE
    if KodCycle=168 then begin
      Fr = S * cld[12]
      ! Klient: Pered cziklom narezaniya
      ! rezby dolzhna stoyat strochka M84S2000 (zhyostkoj)
      if (Cycle<>84) and (Cycle<>74) then begin
        if cld[14]=1 then  ! esli zhyostkij tip derzhatelya
          output "M84 S" + Str(S);
      end
      if Msp=4 then ! SHpindel nazad krutitsya, levaya relba
        Cycle = 74
      else          ! SHpindel vperyod krutitsya, obychnaya relba
        Cycle = 84;
    end;
    if KodCycle=209 then Cycle = 85   ! BORE5
    if KodCycle=210 then begin ! BORE6
      if cld[14]>0 then begin
        Cycle = 76;
        QStep = sqr(cld[6]^2+cld[7]^2);
!        DrillPause = cld[15]*1000;
!        if DrillPause=0 then
!          DrillPause@ = 0
      end else begin
        Cycle = 86;
      end
    end
    if KodCycle=211 then begin ! BORE7
```

```
      if cld[14]>0 then begin
        call Swap(ZCycle, ZClear)
        Cycle = 87;
        QStep = sqr(cld[6]^2+cld[7]^2);
!        DrillPause = cld[15]*1000;
!        if DrillPause=0 then
!          DrillPause@ = 0
      end else begin
        Cycle = 88;
      end
    end
    if KodCycle=212 then Cycle = 88   ! BORE8
    if KodCycle=213 then Cycle = 89   ! BORE9
    if KodCycle=153 then Cycle = 83   ! DEEP
    if KodCycle=288 then Cycle = 73   ! BRKCHP
    FEED_ = Fr
    OutBlock
  end else
    begin
      Cycle = 80;
      OutBlock;
    end
end
```

Доступ из .NET

```
public override void OnCycle(ICLDCycleCommand cmd, CLDArray cld)
```

Наследование: `ICLDCycleCommand : ICLDCommand`. Читайте из строго типизированного `cmd`, из «сырого» массива `cld[i]` (те же индексы, что и в sppx) или через именованные помощники `cmd["..."]`; общие члены `ICLDCommand` описаны в разделе Модель доступа к CLData.

Пример:

```
public override void OnCycle(ICLDCycleCommand cmd, CLDArray cld)
{
    // cld[1] — тип цикла: 71 = ON, 72 = OFF, иначе конкретный цикл (DRILL 163, TAP
168, ...)
    if ((int)cld[1] == 72) {
        nc.G.Show(80);                 // отменить стандартный цикл
        nc.Block.Out();
    } else if ((int)cld[1] != 71) {
        nc.G.Show((int)cld["ISO.G"]);  // ISO.G — сопоставленный G-код (G81..G89)
        nc.Z.Show(cld["H"]);           // уровень дна отверстия
```

```
        nc.R.Show(cld["Top"]);          // уровень отвода
        nc.F.Show(cld["F"]);            // подача
        nc.Block.Out();
    }
}
```

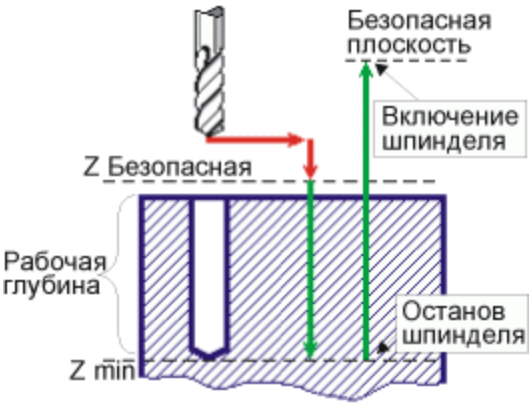
Сопутствующие ссылки

- Цикл сверления G81 `CYCLE_DRILL(163)`
- Цикл сверления G82 `CYCLE_FACE(81)`
- Цикл сверления G84 `CYCLE_TAP(168)`
- Цикл сверления G85 `CYCLE_BORE5(209)`
- Цикл сверления G86 `CYCLE_BORE6(210)`
- Цикл сверления G87 `CYCLE_BORE7(211)`
- Цикл сверления G88 `CYCLE_BORE8(212)`
- Цикл сверления G89 `CYCLE_BORE9(213)`
- Глубокое сверление с удалением стружки `CYCLE_DEEP(153)`
- Глубокое сверление с ломкой стружки `CYCLE_BRKCHP(288)`
- Модель доступа к CLData

CYCLE BORE5(209) - Цикл сверления G85

> Оставлена для совместимости. CYCLE сохранена только для совместимости со старыми постпроцессорами. Текущие версии CAM-системы вместо неё генерируют команду расширенного цикла EXTCYCLE: она предоставляет больше возможностей и позволяет CAM-системе полностью моделировать все вложенные перемещения цикла, чего CYCLE не умеет. Вывод CYCLE всё ещё можно включить в настройках CAM-системы, но это не рекомендуется.

Цикл растачивания G85 выполняет ускоренный подход на безопасный уровень, растачивание на рабочей подаче с остановом шпинделя на уровне Z min и отвод на рабочей подаче на безопасную плоскость.



Цикл включает в себя:

- Ускоренный подвод на уровень Z безопасный.
- Рабочий ход до уровня Z min.
- Останов шпинделя.
- Отвод на рабочей подаче на уровень безопасная плоскость.
- Восстановление направления и частоты вращения шпинделя.

Команда

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

или

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

Параметры

Параметр	Массив CLD	Имя CLD	Описание
BORE5(209)	CLD[1]	CLD.Type	Тип цикла 209 (BORE5).
a	CLD[2]	CLD.A	Глубина отверстия в текущих единицах измерения (мм или дюймы).
MMPM(315) MMPR(316)	CLD[3]	CLD.MMPM	Единицы измерения подачи: 315 (MMPM) – мм в минуту (дюймы в минуту), 316 (MMPR) – мм на оборот (дюймы на оборот).
nm	CLD[4]	CLD.NM	Значение рабочей подачи.
f	CLD[5]	CLD.F	Безопасный уровень.
	CLD[6] CLD[7]		Зарезервировано.
p	CLD[8]	CLD.P	Уровень безопасной плоскости.
	CLD[9] CLD[10]		Зарезервировано.
t	CLD[11]	CLD.Top	Уровень верха отверстия.

Доступ из .NET

Доставляется через обработчик `OnCycle` (диспетчеризация по типу цикла в `cmd/cld[1]`):

```
public override void OnCycle(ICLDCycleCommand cmd, CLDArray cld)
```

Общая модель .NET описана в разделе Циклы обработки отверстий (CYCLE).

Сопутствующие ссылки

- Циклы обработки отверстий (CYCLE)
- Модель доступа к CLData

CYCLE BORE6(210) - Цикл сверления G86

> Оставлена для совместимости. CYCLE сохранена только для совместимости со старыми постпроцессорами. Текущие версии CAM-системы вместо неё генерируют команду расширенного цикла EXTCYCLE: она предоставляет больше возможностей и позволяет CAM-системе полностью моделировать все вложенные перемещения цикла, чего CYCLE не умеет. Вывод CYCLE всё ещё можно включить в настройках CAM-системы, но это не рекомендуется.

Стандартный цикл растачивания G86 выполняет ускоренный подход на безопасный уровень, растачивание на рабочей подаче до уровня глубины отверстия, останов шпинделя на уровне глубины отверстия и ускоренный отвод на безопасную плоскость.



Цикл G86 включает в себя:

- Ускоренный подвод на уровень Z безопасный.
- Рабочий ход до уровня Z min.
- Останов шпинделя. Если включён ориентированный останов шпинделя (параметр CLD [14]), шпиндель устанавливается в заданное угловое положение, после чего инструмент смещается на заданные малые величины по координатам X, Y и Z.
- Ускоренный отвод на уровень безопасная плоскость.
- Восстановление направления и частоты вращения шпинделя.

Команда

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

или

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

Параметры

Параметр	Массив CLD	Имя CLD	Описание
BORE6(210)	CLD[1]	CLD.Type	Тип цикла 210 (BORE6)
a	CLD[2]	CLD.A	Глубина отверстия в текущих единицах измерения (мм или дюймы).
MMPM(315) MMPR(316)	CLD[3]	CLD.MMPM	Единицы измерения подачи: 315 (MMPM) – мм в минуту (дюймы в минуту), 316 (MMPR) – мм на оборот (дюймы на оборот).
nm	CLD[4]	CLD.NM	Значение рабочей подачи.
f	CLD[5]	CLD.F	Безопасный уровень.
	CLD[6]		Величина смещения инструмента по координате X после ориентированного останова шпинделя на дне отверстия.
	CLD[7]		Величина смещения инструмента по координате Y после ориентированного останова шпинделя на дне отверстия

Параметр	Массив CLD	Имя CLD	Описание
p	CLD[8]	CLD.P	Уровень безопасной плоскости.
	CLD[9] CLD[10]		Зарезервировано.
t	CLD[11]	CLD.Top	Уровень верха отверстия.
	CLD[12]		Величина смещения инструмента по координате Z после ориентированного останова шпинделя на дне отверстия
	CLD[13]		Угловое положение шпинделя при останове на дне отверстия (градусы)
	CLD[14]		Состояние ориентированного останова шпинделя на дне отверстия: 0 - выключен, 1 - включён.

Доступ из .NET

Доставляется через обработчик `OnCycle` (диспетчеризация по типу цикла в `cmd`/`cld[1]`):

```
public override void OnCycle(ICLDCycleCommand cmd, CLDArray cld)
```

Общая модель .NET описана в разделе Циклы обработки отверстий (CYCLE).

Сопутствующие ссылки

- Циклы обработки отверстий (CYCLE)
- Модель доступа к CLData

CYCLE BORE7(211) - Цикл сверления G87

> Оставлена для совместимости. CYCLE сохранена только для совместимости со старыми постпроцессорами. Текущие версии CAM-системы вместо неё генерируют команду расширенного цикла EXTCYCLE: она предоставляет больше возможностей и позволяет CAM-системе полностью моделировать все вложенные перемещения цикла, чего CYCLE не умеет. Вывод CYCLE всё ещё можно включить в настройках CAM-системы, но это не рекомендуется.

Стандартный цикл растачивания G87 выполняет ускоренный подход на безопасный уровень, растачивание на рабочей подаче с остановом шпинделя на уровне глубины отверстия и отводом вручную на безопасную плоскость.



Цикл G87 включает в себя:

- Ускоренный подвод на уровень Z безопасный.
- Рабочий ход до уровня Z min.
- Останов шпинделя. Если включён ориентированный останов шпинделя (параметр CLD [14]), шпиндель устанавливается в заданное угловое положение, после чего инструмент смещается на заданные малые величины по координатам X, Y и Z.
- Отвод вручную на уровень безопасная плоскость.
- Восстановление направления и частоты вращения шпинделя.

Команда

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

или

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

Параметры

Параметр	Массив CLD	Имя CLD	Описание
BORE7(211)	CLD[1]	CLD.Type	Тип цикла 211 (BORE7).
a	CLD[2]	CLD.A	Глубина отверстия в текущих единицах измерения (мм или дюймы).
MMPM(315) MMPR(316)	CLD[3]	CLD.MMPM	Единицы измерения подачи: 315 (MMPM) – мм в минуту (дюймы в минуту), 316 (MMPR) – мм на оборот (дюймы на оборот).
nm	CLD[4]	CLD.NM	Значение рабочей подачи.
f	CLD[5]	CLD.F	Безопасный уровень.
	CLD[6]		Величина смещения инструмента по координате X после ориентированного останова шпинделя на дне отверстия.
	CLD[7]		Величина смещения инструмента по координате Y после ориентированного останова шпинделя на дне отверстия.

Параметр	Массив CLD	Имя CLD	Описание
p	CLD[8]	CLD.P	Уровень безопасной плоскости.
	CLD[9] CLD[10]		Зарезервировано.
t	CLD[11]	CLD.Top	Уровень верха отверстия.
	CLD[12]		Величина смещения инструмента по координате Z после ориентированного останова шпинделя на дне отверстия.
	CLD[13]		Угловое положение шпинделя при останове на дне отверстия (градусы)
	CLD[14]		Состояние ориентированного останова шпинделя на дне отверстия: 0 - выключен, 1 - включён.

Доступ из .NET

Доставляется через обработчик `OnCycle` (диспетчеризация по типу цикла в `cmd`/`cld[1]`):

```
public override void OnCycle(ICLDCycleCommand cmd, CLDArray cld)
```

Общая модель .NET описана в разделе Циклы обработки отверстий (CYCLE).

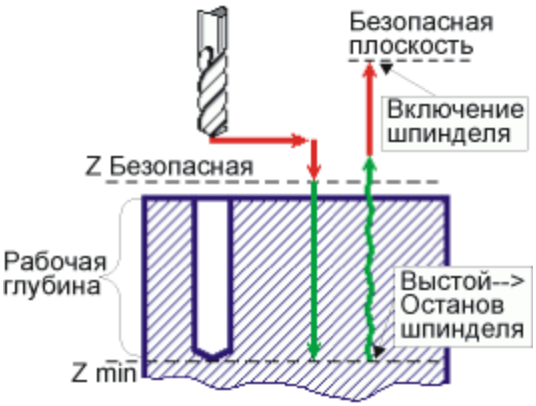
Сопутствующие ссылки

- Циклы обработки отверстий (CYCLE)
- Модель доступа к CLData

CYCLE BORE8(212) - Цикл сверления G88

> Оставлена для совместимости. CYCLE сохранена только для совместимости со старыми постпроцессорами. Текущие версии CAM-системы вместо неё генерируют команду расширенного цикла EXTCYCLE: она предоставляет больше возможностей и позволяет CAM-системе полностью моделировать все вложенные перемещения цикла, чего CYCLE не умеет. Вывод CYCLE всё ещё можно включить в настройках CAM-системы, но это не рекомендуется.

Стандартный цикл растачивания G88 выполняет ускоренный подход на безопасный уровень, растачивание на рабочей подаче с выстоем на уровне глубины отверстия, останов шпинделя на уровне глубины отверстия и отвод вручную на безопасную плоскость.



Цикл G88 включает в себя:

- Ускоренный подвод на уровень Z безопасный.
- Рабочий ход до уровня Z min.
- Выстой.
- Останов шпинделя.
- Отвод вручную на уровень безопасная плоскость.
- Восстановление направления и частоты вращения шпинделя.

Команда

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

или

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

Параметры

Параметр	Массив CLD	Имя CLD	Описание
BORE8(212)	CLD[1]	CLD.Type	Тип цикла 212 (BORE8).
a	CLD[2]	CLD.A	Глубина отверстия в текущих единицах измерения (мм или дюймы).
MMPM(315) MMPR(316)	CLD[3]	CLD.MMPM	Единицы измерения подачи: 315 (MMPM) – мм в минуту (дюймы в минуту), 316 (MMPR) – мм на оборот (дюймы на оборот).
nm	CLD[4]	CLD.NM	Значение рабочей подачи.
f	CLD[5]	CLD.F	Безопасный уровень.
	CLD[6] CLD[7]		Зарезервировано.
p	CLD[8]	CLD.P	Уровень безопасной плоскости.
DWELL	CLD[9]	CLD.DWELL	Ключевое слово времени выстоя.
h	CLD[10]	CLD.H	Время выстоя в секундах.
t	CLD[11]	CLD.Top	Уровень верха отверстия.

Доступ из .NET

Доставляется через обработчик `OnCycle` (диспетчеризация по типу цикла в `cmd/cld[1]`):

```
public override void OnCycle(ICLDCycleCommand cmd, CLDArray cld)
```

Общая модель .NET описана в разделе Циклы обработки отверстий (CYCLE).

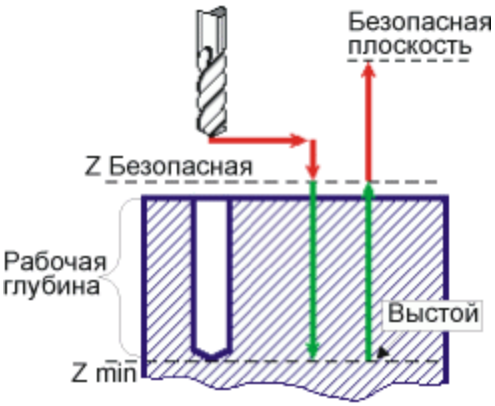
Сопутствующие ссылки

- Циклы обработки отверстий (CYCLE)
- Модель доступа к CLData

CYCLE BORE9(213) - Цикл сверления G89

> Оставлена для совместимости. CYCLE сохранена только для совместимости со старыми постпроцессорами. Текущие версии CAM-системы вместо неё генерируют команду расширенного цикла EXTCYCLE: она предоставляет больше возможностей и позволяет CAM-системе полностью моделировать все вложенные перемещения цикла, чего CYCLE не умеет. Вывод CYCLE всё ещё можно включить в настройках CAM-системы, но это не рекомендуется.

Стандартный цикл растачивания G89 выполняет ускоренный подход на безопасный уровень, растачивание на рабочей подаче с выстоем на уровне глубины отверстия и отвод на рабочей подаче на безопасную плоскость.



Цикл G89 включает в себя:

- Ускоренный подвод на уровень Z безопасный.
- Рабочий ход до уровня Z min.
- Выстой.
- Отвод на рабочей подаче на уровень безопасная плоскость.

Команда

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

или

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

Параметры

Параметр	Массив CLD	Имя CLD	Описание
BORE9(213)	CLD[1]	CLD.Type	Тип цикла 213 (BORE9).
a	CLD[2]	CLD.A	Глубина отверстия в текущих единицах измерения (мм или дюймы).
MMPM(315) MMPR(316)	CLD[3]	CLD.MMPM	Единицы измерения подачи: 315 (MMPM) – мм в минуту (дюймы в минуту), 316 (MMPR) – мм на оборот (дюймы на оборот).
nm	CLD[4]	CLD.NM	Значение рабочей подачи.
f	CLD[5]	CLD.F	Безопасный уровень.
	CLD[6] CLD[7]		Зарезервировано.
p	CLD[8]	CLD.P	Уровень безопасной плоскости.
DWELL	CLD[9]	CLD.DWELL	Ключевое слово времени выстоя.
h	CLD[10]	CLD.H	Время выстоя в секундах.
t	CLD[11]	CLD.Top	Уровень верха отверстия.

Доступ из .NET

Доставляется через обработчик `OnCycle` (диспетчеризация по типу цикла в `cmd/cld[1]`):

```
public override void OnCycle(ICLDCycleCommand cmd, CLDArray cld)
```

Общая модель .NET описана в разделе Циклы обработки отверстий (CYCLE).

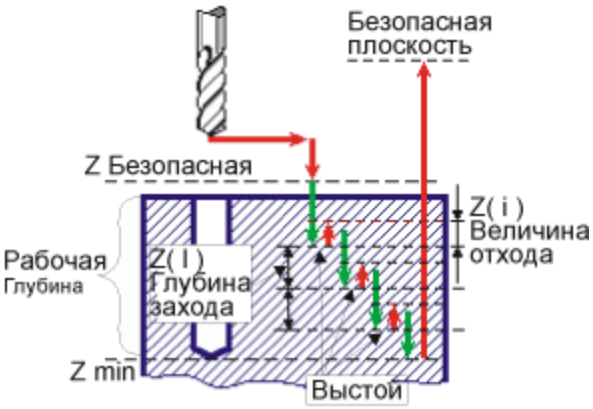
Сопутствующие ссылки

- Циклы обработки отверстий (CYCLE)
- Модель доступа к CLData

CYCLE BRKCHP(288) - Глубокое сверление с ломкой стружки

> Оставлена для совместимости. CYCLE сохранена только для совместимости со старыми постпроцессорами. Текущие версии CAM-системы вместо неё генерируют команду расширенного цикла EXTCYCLE: она предоставляет больше возможностей и позволяет CAM-системе полностью моделировать все вложенные перемещения цикла, чего CYCLE не умеет. Вывод CYCLE всё ещё можно включить в настройках CAM-системы, но это не рекомендуется.

Цикл глубокого сверления с удалением стружки выполняет ускоренный подход на безопасный уровень и повторяющееся сверление согласно заданным значениям глубины захода Zl и отхода Zi.



Цикл включает в себя:

- Ускоренный подвод на уровень Z безопасный.
- Рабочий ход до Zl.
- Ускоренный отвод на величину отхода Zi и Выстой.
- Рабочий ход до уровня Zl + Zi.
- Ускоренный отвод на величину отхода Zi и Выстой.
- Повторять до достижения уровня глубины отверстия.
- Ускоренный отвод на уровень безопасная плоскость.

Команда

```
CYCLE BRKCHP(288), A a, MMPM(315), N nm, F f,
L l, I i, P p, DWELL(279) h, T t
```

или

```
CYCLE BRKCHP(288), A a, MMPR(316), M nm, F f,
L l, I i, P p, DWELL(279) h, T t
```

Параметры

Параметр	Массив CLD	Имя CLD	Описание
BRKCHP(288)	CLD[1]	CLD.Type	Тип цикла 288 (BRKCHP).
a	CLD[2]	CLD.A	Глубина отверстия в текущих единицах измерения (мм или дюймы).
MMPM(315) MMPR(316)	CLD[3]	CLD.MMPM	Единицы измерения подачи: 315 (MMPM) – мм в минуту (дюймы в минуту), 316 (MMPR) – мм на оборот (дюймы на оборот).
nm	CLD[4]	CLD.NM	Значение рабочей подачи.
f	CLD[5]	CLD.F	Безопасный уровень.
L	CLD[6]	CLD.L	Величина захода Zl
i	CLD[7]	CLD.I	Величина отхода Zi
p	CLD[8]	CLD.P	Уровень безопасной плоскости.
DWELL	CLD[9]	CLD.DWELL	Ключевое слово времени выстоя.
h	CLD[10]	CLD.H	Выстой инструмента в секундах.

Параметр	Массив CLD	Имя CLD	Описание
t	CLD[11]	CLD.Top	Уровень верха отверстия.

Доступ из .NET

Доставляется через обработчик `OnCycle` (диспетчеризация по типу цикла в `cmd`/`cld[1]`):

```
public override void OnCycle(ICLDCycleCommand cmd, CLDArray cld)
```

Общая модель .NET описана в разделе Циклы обработки отверстий (CYCLE).

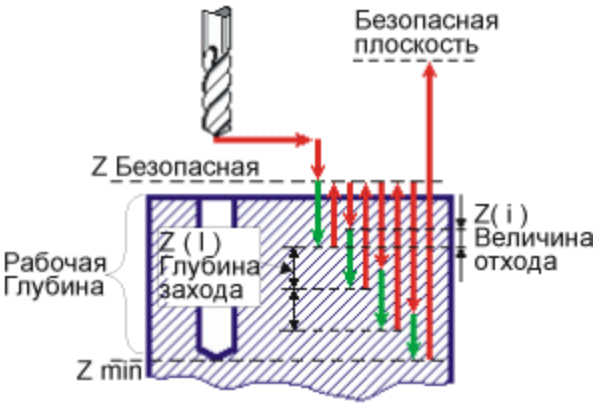
Сопутствующие ссылки

- Циклы обработки отверстий (CYCLE)
- Модель доступа к CLData

CYCLE DEEP(153) - Глубокое сверление с удалением стружки

> Оставлена для совместимости. CYCLE сохранена только для совместимости со старыми постпроцессорами. Текущие версии CAM-системы вместо неё генерируют команду расширенного цикла EXTCYCLE: она предоставляет больше возможностей и позволяет CAM-системе полностью моделировать все вложенные перемещения цикла, чего CYCLE не умеет. Вывод CYCLE всё ещё можно включить в настройках CAM-системы, но это не рекомендуется.

Цикл глубокого сверления с удалением стружки выполняет ускоренный подход на безопасный уровень и повторяющееся сверление согласно заданным значениям глубины захода Zl и отхода Zi.



Цикл включает в себя:

- Ускоренный подвод на уровень Z безопасный.
- Рабочий ход до Zl.
- Ускоренный отвод на уровень Z безопасный.
- Ускоренный подвод на уровень Zl - Zi.
- Рабочий ход до уровня Zl + Zi.
- Ускоренный отвод на уровень Z безопасный.
- Ускоренный подвод на уровень 2*Zl - Zi.
- Рабочий ход до уровня Zl + Zi.
- Ускоренный отвод на уровень Z безопасный.
- Повторять до достижения уровня глубины отверстия.
- Ускоренный отвод на уровень безопасная плоскость.

Команда

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

или

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

Параметры

Параметр	Массив CLD	Имя CLD	Описание
DEEP(153)	CLD[1]	CLD.Type	Тип цикла 153 (DEEP).
a	CLD[2]	CLD.A	Глубина отверстия в текущих единицах измерения (мм или дюймы).
MMPM(315) MMPR(316)	CLD[3]	CLD.MMPM	Единицы измерения подачи: 315 (MMPM) – мм в минуту (дюймы в минуту), 316 (MMPR) – мм на оборот (дюймы на оборот).
nm	CLD[4]	CLD.NM	Значение рабочей подачи.
f	CLD[5]	CLD.F	Безопасный уровень.
L	CLD[6]	CLD.L	Величина захода Zl
i	CLD[7]	CLD.I	Величина отхода Zi
p	CLD[8]	CLD.P	Уровень безопасной плоскости.
	CLD[9] CLD[10]		Зарезервировано.

Параметр	Массив CLD	Имя CLD	Описание
t	CLD[11]	CLD.Top	Уровень верха отверстия.

Доступ из .NET

Доставляется через обработчик `OnCycle` (диспетчеризация по типу цикла в `cmd`/`cld[1]`):

```
public override void OnCycle(ICLDCycleCommand cmd, CLDArray cld)
```

Общая модель .NET описана в разделе Циклы обработки отверстий (CYCLE).

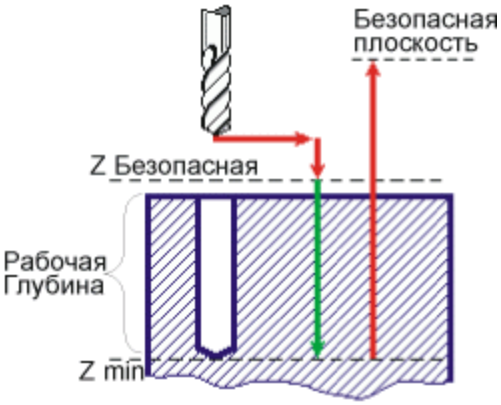
Сопутствующие ссылки

- Циклы обработки отверстий (CYCLE)
- Модель доступа к CLData

CYCLE DRILL(163) - Цикл сверления G81

> Оставлена для совместимости. CYCLE сохранена только для совместимости со старыми постпроцессорами. Текущие версии CAM-системы вместо неё генерируют команду расширенного цикла EXTCYCLE: она предоставляет больше возможностей и позволяет CAM-системе полностью моделировать все вложенные перемещения цикла, чего CYCLE не умеет. Вывод CYCLE всё ещё можно включить в настройках CAM-системы, но это не рекомендуется.

Стандартный цикл простого сверления G81 выполняет сверление отверстия с ускоренным подходом на безопасный уровень и последующим отводом на безопасную плоскость.



Цикл включает в себя:

- Ускоренный подвод на уровень Z безопасный.
- Рабочий ход до уровня Z min.
- Ускоренный отвод на уровень безопасная плоскость.

Команда

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

или

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

Параметры

Параметр	Массив CLD	Имя CLD	Описание
DRILL(163)	CLD[1]	CLD.Type	Тип цикла 163 (DRILL).
a	CLD[2]	CLD.A	Глубина отверстия в текущих единицах измерения (мм или дюймы).
MMPM(315) MMPR(316)	CLD[3]	CLD.MMPM	Единицы измерения подачи: 315 (MMPM) – мм в минуту (дюймы в минуту), 316 (MMPR) – мм на оборот (дюймы на оборот).
nm	CLD[4]	CLD.NM	Подача.
f	CLD[5] CLD[6], CLD[7]	CLD.F Зарезервировано	Безопасный уровень.
p	CLD[8] CLD[9], CLD[10]	CLD.P Зарезервировано	Уровень безопасной плоскости.
t	CLD[11]	CLD.Top	Уровень верха отверстия.

Доступ из .NET

Доставляется через обработчик `OnCycle` (диспетчеризация по типу цикла в `cmd`/`cld[1]`):

```
public override void OnCycle(ICLDCycleCommand cmd, CLDArray cld)
```

Общая модель .NET описана в разделе Циклы обработки отверстий (CYCLE).

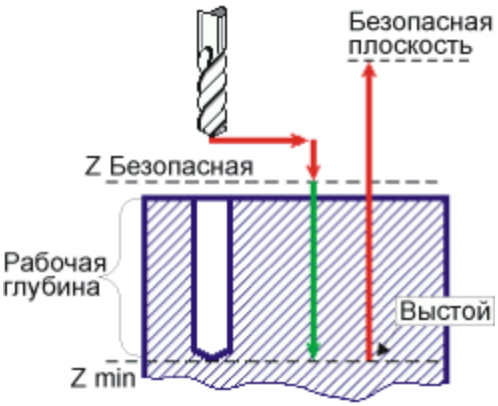
Сопутствующие ссылки

- Циклы обработки отверстий (CYCLE)
- Модель доступа к CLData

CYCLE FACE(81) - Цикл сверления G82

> Оставлена для совместимости. CYCLE сохранена только для совместимости со старыми постпроцессорами. Текущие версии CAM-системы вместо неё генерируют команду расширенного цикла EXTCYCLE: она предоставляет больше возможностей и позволяет CAM-системе полностью моделировать все вложенные перемещения цикла, чего CYCLE не умеет. Вывод CYCLE всё ещё можно включить в настройках CAM-системы, но это не рекомендуется.

Стандартный цикл сверления с выстоем G82 выполняет сверление с ускоренным подходом на безопасный уровень, выстоем на уровне Z min на рабочей подаче и отводом на безопасную плоскость на ускоренной подаче. Длительность выстоя задаётся параметрами команды.



Цикл сверления G82 включает в себя:

- Ускоренный подвод на уровень Z безопасный.
- Рабочий ход до уровня Z min.
- Выстой на уровне Z min.
- Ускоренный отвод на уровень безопасная плоскость.

Команда

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

или

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

Параметры

Параметр	Массив CLD	Имя CLD	Описание
FACE(81)	CLD[1]	CLD.Type	Тип цикла 81 (FACE).
a	CLD[2]	CLD.A	Глубина отверстия в текущих единицах измерения (мм или дюймы).
MMPM(315) MMPR(316)	CLD[3]	CLD.MMPM	Единицы измерения подачи: 315 (MMPM) – мм в минуту (дюймы в минуту), 316 (MMPR) – мм на оборот (дюймы на оборот).
nm	CLD[4]	CLD.NM	Значение подачи.
f	CLD[5]	CLD.F	Безопасный уровень.
	CLD[6] CLD[7]		Зарезервировано.
p	CLD[8]	CLD.P	Уровень безопасной плоскости.
DWELL	CLD[9]	CLD.DWell	Ключевое слово времени выстоя.
h	CLD[10]	CLD.H	Время выстоя в секундах.
t	CLD[11]	CLD.Top	Уровень верха отверстия.

Доступ из .NET

Доставляется через обработчик `OnCycle` (диспетчеризация по типу цикла в `cmd/cld[1]`):

```
public override void OnCycle(ICLDCycleCommand cmd, CLDArray cld)
```

Общая модель .NET описана в разделе Циклы обработки отверстий (CYCLE).

Сопутствующие ссылки

- Циклы обработки отверстий (CYCLE)
- Модель доступа к CLData

CYCLE TAP(168) - Цикл сверления G84

> Оставлена для совместимости. CYCLE сохранена только для совместимости со старыми постпроцессорами. Текущие версии CAM-системы вместо неё генерируют команду расширенного цикла EXTCYCLE: она предоставляет больше возможностей и позволяет CAM-системе полностью моделировать все вложенные перемещения цикла, чего CYCLE не умеет. Вывод CYCLE всё ещё можно включить в настройках CAM-системы, но это не рекомендуется.

Стандартный цикл нарезания резьбы метчиком G84 выполняет ускоренный подход на безопасный уровень и нарезание резьбы с возвратом на рабочей подаче при реверсивном вращении шпинделя.



Цикл G84 состоит из:

- Ускоренный подвод на уровень Z безопасный.
- Рабочий ход до уровня Z min.
- Остановка вращения шпинделя.
- Реверс шпинделя и возврат на рабочем ходу до уровня Z безопасный.
- Остановка вращения шпинделя.
- Восстановление нормального вращения шпинделя и ускоренный отвод на уровень безопасная плоскость.

Команда

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

или

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

Параметры

Параметр	Массив CLD	Имя CLD	Описание
TAP(168)	CLD[1]	CLD.Type	Тип цикла 168 (TAP).
a	CLD[2]	CLD.A	Глубина отверстия в текущих единицах измерения (мм или дюймы).
MMPM(315) MMPR(316)	CLD[3]	CLD.MMPM	Единицы измерения подачи: 315 (MMPM) – мм в минуту (дюймы в минуту), 316 (MMPR) – мм на оборот (дюймы на оборот).
nm	CLD[4]	CLD.NM	Значение подачи
f	CLD[5]	CLD.F	Уровень Z безопасный
L	CLD[6]	CLD.L	Величина захода Zl
i	CLD[7]	CLD.I	Величина отхода Zi
p	CLD[8]	CLD.P	Уровень безопасной плоскости
DWELL(279)	CLD[9]	CLD.DWell	Ключевое слово времени выстоя.
h	CLD[10]	CLD.H	Время выстоя в секундах.
t	CLD[11]	CLD.Top	Уровень верха отверстия

Параметр	Массив CLD	Имя CLD	Описание
s	CLD[12]	CLD.Step	Шаг резьбы
ps	CLD[13]	CLD.Pos	Начальный угол шпинделя
sc	CLD[14]	CLD.Socket	Тип патрона: 0 – плавающий, 1 – жёсткий
fr	CLD[15]	CLD.FR	Подача возврата

Доступ из .NET

Доставляется через обработчик `OnCycle` (диспетчеризация по типу цикла в `cmd`/`cld[1]`):

```
public override void OnCycle(ICLDCycleCommand cmd, CLDArray cld)
```

Общая модель .NET описана в разделе Циклы обработки отверстий (CYCLE).

Сопутствующие ссылки

- Циклы обработки отверстий (CYCLE)
- Модель доступа к CLData

DELAY - Задержка

Отображение команды

Как команда отображается в списке команд CLData:

```
DELAY A a
```

DELAY — команда используется для приостановки выполнения программы на заданное время. DELAY обычно транслируется в G04.

Доступ из sppx

Обработчик: `program Delay`

Параметр	Массив CLD	Имя CLD	Описание
a	CLD[1]	CLD.A	Время задержки в секундах

Пример (из постпроцессора Fanuc (30i)_Mill):

```
program Delay
  OutBlock
  GFunc = 4; GFunc@ = MaxReal
  Pause = CLD[1]; Pause@ = MaxReal
  OutBlock
end
```

Доступ из .NET

Обработчик:

```
public override void OnDelay(ICLDDelayCommand cmd, CLDArray cld)
```

Наследование: `ICLDDelayCommand : ICLDCommand`.

Читайте параметры из строго типизированного объекта `cmd` (рекомендуется); доступны также «сырой» массив `cld[i]` (те же индексы, что и в sppx) и именованные обёртки `cmd["..."]`. В таблице перечислены члены команды, включая унаследованные от базовых интерфейсов; общие члены `ICLDCommand` (навигация, `CLD`, `CLDFile`, `TechOperation`, ...) описаны в разделе Модель доступа к CLData.

Член	Тип	Описание
`cmd.TimeSpan`	`double`	Время задержки в секундах.

Пример - `OnDelay` (из постпроцессора Fanuc (30i)_Mill_DN):

```
public override void OnDelay(ICLDDelayCommand cmd, CLDArray cld)
{
    nc.GDelay.Show(4);
    nc.XDelay.Show(cmd.TimeSpan);
    nc.Block.Out();
}
```

Сопутствующие ссылки

- Модель доступа к CLData

EDMMOVE - Электроэрозионное перемещение

Отображение команды

Как команда отображается в списке команд CLData:

```
EDMMOVE MODE: RAPID(0) | 2AXIS(1) | TAPER(2) | MULTIPROF(3)| 4AXIS(4),
SPAN1: CUT(0) | ARC(1), SPAN2: CUT(0) | ARC(1),
EP1: (X x; Y y; Z z), EP2: (X x; Y y; Z z),
PC1: (X x; Y y), PC2: (X x; Y y), R1 r1, R2 r2, A a,
ROLLMODE: OFF(0) | SHARP(1) | CONICAL(2) | CYLINDRICAL(3) |
FIXED(4),
ROLLR1 rollr1, ROLLR2 rollr2
```

Команда перемещения при электроэрозионной обработке формируется операциями проволочно-вырезной электроэрозионной обработки CAM-системы. Команда задаёт координаты перемещения проволоки и устанавливает режимы интерполяции перемещения.

Доступ из sppx

Обработчик: `program EDMMove`

Параметр	Массив CLD	Имя CLD	Описание
Режим перемещения	CLD[1]	CLD.Mode	RAPID(0) - ускоренное перемещение, 2AXIS(1) - 2-х координатное перемещение, TAPER(2) - 2-х координатное перемещение с конусностью, MULTIPROF(3) - многоконтурное перемещение (дуги и линии), 4AXIS(4) - 4-осевое перемещение (только линии)
Тип элемента нижнего контура	CLD[2]	CLD.Span1	CUT(0) - линия, ARC(1) - дуга

Параметр	Массив CLD	Имя CLD	Описание
Тип элемента верхнего контура	CLD[3]	CLD.Span2	CUT(0) - линия, ARC(1) - дуга
Конечная точка элемента нижнего контура, EP1	CLD[4] CLD[5] CLD[6]	CLD.Ep1X CLD.Ep1Y CLD.Ep1Z	Координаты X, Y и Z конечной точки элемента нижнего контура
Конечная точка элемента верхнего контура, EP2	CLD[7] CLD[8] CLD[9]	CLD.Ep2X CLD.Ep2Y CLD.Ep2Z	Координаты X, Y и Z конечной точки элемента верхнего контура
Центр дуги нижнего контура, PC1	CLD[10] CLD[11]	CLD.Pc1X CLD.Pc1Y	Координаты X и Y центра дуги нижнего контура, если этот элемент является дугой.
Центр дуги верхнего контура, PC2	CLD[12] CLD[13]	CLD.Pc2X CLD.Pc2Y	Координаты X и Y центра дуги верхнего контура, если этот элемент является дугой.
Радиус дуги нижнего контура, R1	CLD[14]	CLD.R1	Если элемент нижнего контура является дугой, параметр задаёт знаковый радиус дуги: R > 0 - дуга против часовой стрелки, R < 0 - дуга по часовой стрелке.
Радиус дуги верхнего контура, R2	CLD[15]	CLD.R2	Если элемент верхнего контура является дугой, параметр задаёт знаковый радиус

Параметр	Массив CLD	Имя CLD	Описание
			дуги: R > 0 - дуга против часовой стрелки, R < 0 - дуга по часовой стрелке.
Угол конусности, A	CLD[16]	CLD.A	В режиме перемещения TAPER(2) параметр задаёт знаковый угол конусности: A > 0 - правая конусность, A < 0 - левая конусность.
Режим обката острых углов, ROLLMODE	CLD[17]	CLD.RollMode	Параметр задаёт режим обката острых углов для 2-координатной и конусной обработки (функция CornerR): OFF(0) - обкат углов выключен (R1 = 0, R2 = 0); SHARP(1) - в этом режиме обкатываются только углы нижнего контура (R1 = R1, R2 = 0); CONICAL(2) - обкатываются углы и нижнего, и верхнего контура, радиусы обката вычисляются из угла конусности (R1 = R1, R2 = R1 + смещение конусности); CYLINDRICAL(3) - обкатываются углы нижнего и верхнего

Параметр	Массив CLD	Имя CLD	Описание
			контура, радиусы обката равны (R1 = R1, R2 = R1); FIXED(4) - обкатываются углы нижнего и верхнего контура, радиусы обката независимы (R1 = R1, R2 = R2)
Радиус обката элемента нижнего контура, ROLLR1	CLD[18]	CLD.RollR1	Параметр задаёт радиус обката острого угла элемента нижнего контура
Радиус обката элемента верхнего контура, ROLLR2	CLD[19]	CLD.RollR2	Параметр задаёт радиус обката острого угла элемента верхнего контура

Пример (из постпроцессора AgieCharmilles (ROBOFIL)_WireEDM):

```
program EDMMove

  ! Задать LCS
  if LCSSet<>1 then begin
    LCSSet = 1
    GCS = 92;        GCS@ = MaxReal
    X1  = XOrigin;   X1@  = MaxReal
    Y1  = YOrigin;   Y1@  = MaxReal
    ITop    = TopLevel;  ITop@    = MaxReal
    JBottom = BaseLevel; JBottom@ = MaxReal
    OutBlock
  end

  RollModeChanged = 0
  if (cld.RollMode<>0) and (RollOn<>48) then begin
    GRoll  = 48
    RollOn = 48
    RollModeChanged = 1
    GRoll@ = MaxReal
```

```
  end else
  if (cld.RollMode=0) and (RollOn<>49) then begin
    GRoll  = 49
    RollOn = 49
    GRoll@ = MaxReal
  end

  ! Команды заправки и обрыва проволоки
  if (cld.Mode<>0) and (WireInserted=0) and (MainProg=1) then begin
    call InsertWire
  end
  if (cld.Mode=0) and (WireInserted=1) and (MainProg=1) then begin
    if (cld.Ep1X<>Fp1X) or (cld.Ep1Y<>Fp1Y) then begin
      call BreakWire
    end
  end

  ! Вывод номера технологии прохода
  if (cld.Mode<>0) and (WireInserted=1) then begin
    If OldSCode<>SCode then begin
      OldSCode  = SCode
      STechCode = SCode
      STechCode@ = MaxReal
      OUTBLOCK
    end
  end

  ! Включение конического режима обработки
  if (Cld.Mode=2) and (TaperMode<>60) then begin
    TaperMode  = 60
    TaperMode@ = MaxReal
    OUTBLOCK
  end

  ! Включение коррекции
  if (CompensNeedOut>0) and (GCompens<>40) then begin
    GCompens@ = MaxReal
    CompensNeedOut = 0
  end

  ! Включение режима движения
  if cld.Mode=3 then G2Contour = 51 else ! 2Contour
  if cld.Mode=4 then GUV = 74            ! 4Axis_UV

!  if (G2Contour<>G2Contour@) or (GUV<>GUV@) then begin
!    OutBlock
```

```
!    GInterp1  = 0;
!    GInterp1@ = GInterp1;
!  end

  ! Выключение коррекции
  if (CompensNeedOut>0) and (GCompens=40) then begin
    GCompens@ = MaxReal
    CompensNeedOut = 0
  end

  ! Вывод координат
  X1 = cld.Ep1X
  Y1 = cld.Ep1Y
  OUTG = 0
  case cld.Mode of
    0: begin ! Ускоренное перемещение
      if (Abs(X1-X1@)>Toler) or (Abs(Y1-Y1@)>Toler) then
        GInterp1 = 0 ! Ускоренное перемещение
    end
    1, 2, 3, 4: begin ! 2-координатное, конусность, 4D
      if (Abs(X1-X1@)>Toler) or (Abs(Y1-Y1@)>Toler) then begin
        if cld.Span1=1 then begin ! Дуга
          if cld.R1>0 then GInterp1 = 3
                      else GInterp1 = 2
          I1 = cld.Pc1X - Fp1X;  I1@ = MaxReal
          J1 = cld.Pc1Y - Fp1Y;  J1@ = MaxReal
        end else begin ! Линия
            GInterp1 = 1
        end
        GInterp1@ = MaxReal
        OUTG = GInterp1
      end
      if cld.Mode=2 then begin      ! Конусность
        if cld.A>0 then GTaper = 52
                   else GTaper = 51
        A = abs(cld.A);
      end else
      if cld.Mode=3 then begin ! 4D
        U = cld.Ep2X; U@ = MaxReal
        V = cld.Ep2Y; V@ = MaxReal
        if cld.Span2=1 then begin ! Дуга
          if cld.R2>0 then begin
            if (OutG=0) or (OutG<>3) then begin
              GInterp2 = 3
            end
          end else begin
```

```
          if (OutG=0) or (OutG<>2)  then begin
            GInterp2 = 2
          end
        end
        K = cld.Pc2X - Fp2X;  K@ = MaxReal
        L = cld.Pc2Y - Fp2Y;  L@ = MaxReal
      end else begin ! Линия
        if (Abs(U-U@)>Toler) or (Abs(V-V@)>Toler) then begin
          if (OutG=0) or (OutG<>1) then begin
            GInterp2 = 1
            GInterp2@ = MaxReal
          end
        end
      end
    end
    if ((cld.Mode=1) or (cld.Mode=2)) and
       (cld.RollMode>0)
    then begin  ! Обкат углов (CornerR)
      RollR1 = cld.RollR1;
      if RollModeChanged=1 then begin
        RollR1@ = MaxReal
        RollModeChanged =0
      end
    end
  end
end

! Выключение режима движения
if (cld.Mode<>2) and (GTaper<>50) then begin
  GTaper = 50
  !TaperMode = MaxReal;
end
if (cld.Mode<>4) and (GUV<>75) then begin
  U = 0; V = 0
end
OutBlock
if (cld.Mode<>4) and (GUV<>75) then begin
  GUV = 75
  U = MaxReal; U@ = U
  V = MaxReal; V@ = V
end
OutBlock

! Запоминание текущих координат
Fp1X = cld.Ep1X
Fp1Y = cld.Ep1Y
```

```
  if cld.Mode<3 then begin ! 2D
    Fp2X = Fp1X
    Fp2Y = Fp1Y
  end else begin           ! 4D
    Fp2X = cld.Ep2X
    Fp2Y = cld.Ep2Y
  end
end
```

Доступ из .NET

Обработчики:

```
public override void OnBeforeMovement(ICLDMotionCommand cmd, CLDArray cld)   //
групповой обработчик - перед любой командой перемещения
public override void OnEDMMove(ICLDEDMMoveCommand cmd, CLDArray cld)
public override void OnAfterMovement(ICLDMotionCommand cmd, CLDArray cld)   //
групповой обработчик - после любой команды перемещения
```

Наследование: `ICLDEDMMoveCommand : ICLDMotionCommand : ICLDCommand`.

Читайте параметры из строго типизированного объекта `cmd` (рекомендуется); также доступны «сырой» массив `cld[i]` (те же индексы, что и в sppx) и именованные обёртки `cmd["..."]`. В таблице перечислены члены команды, включая унаследованные от базовых интерфейсов; общие члены `ICLDCommand` (навигация, `CLD`, `CLDFile`, `TechOperation`, ...) описаны в Модели доступа к CLData.

Член	Тип	Описание
`cmd.MotionMode`	`CLDEDMMotionMode`	Режим перемещения: Rapid, Plain2D, Taper2D, MultiProf4D или CutsOnly4D.
`cmd.IsRapidMove`	`bool`	Возвращает "True", если режим перемещения - Rapid, иначе возвращает "False".
`cmd.IsPlain2DMove`	`bool`	Возвращает "True", если режим перемещения - Plain2D, иначе возвращает "False".

Член	Тип	Описание
`cmd.IsTaper2DMove`	`bool`	Возвращает "True", если режим перемещения - Taper2D, иначе возвращает "False".
`cmd.IsMultiProf4DMove`	`bool`	Возвращает "True", если режим перемещения - MultiProf4D, иначе возвращает "False".
`cmd.IsCutsOnly4DMove`	`bool`	Возвращает "True", если режим перемещения - CutsOnly4D, иначе возвращает "False".
`cmd.Lower`	`ICLDEDMMoveSpan`	Параметры перемещения нижней направляющей (тип, координаты и радиус).
`cmd.Upper`	`ICLDEDMMoveSpan`	Параметры перемещения верхней направляющей (тип, координаты и радиус).
`cmd.TaperAngle`	`double`	В режиме перемещения Taper2D параметр задаёт знаковый угол конусности: "A > 0" - правая конусность, "A < 0" - левая конусность.
`cmd.RollMode`	`EDMRollMode`	Режим обката острых углов для 2-координатной и конусной обработки (функция CornerR: off, sharp, conical, cylindrical или fixed).
`cmd.EP`	`TInp3DPoint`	Конечная точка (X, Y, Z) перемещения

Пример - `OnEDMMove` (из постпроцессора AgieCharmilles_WireEDM_DN):

```
public override void OnEDMMove(ICLDEDMMoveCommand cmd, CLDArray cld)
{
    // Команды заправки и обрыва проволоки
    if (!cmd.IsRapidMove && !wireInserted)
    {
        InsertWire();
    }
    if (cmd.IsRapidMove && wireInserted)
    {
        if (cmd.Lower.EP != previousLowerPoint)
            BreakWire();
    }

    // Выбор условий резания
    if (conditionsNeedOut)
    {
        nc.C.Show();
        nc.Block.Out();
        conditionsNeedOut = false;
    }

    // Включение коррекции
    if (compensNeedOut && nc.GCompens != 40)
    {
        nc.GCompens.Show();
        nc.H.Show();
        nc.Block.Out();
        compensNeedOut = false;
    }

    // Включение режима движения
    if (cmd.IsMultiProf4DMove)
    {
        nc.G2Contour.v = 61;
    }
    else if (cmd.IsCutsOnly4DMove)
    {
        nc.GUV.v = 74;
    }
    if (nc.G2Contour.ValuesDiffer || nc.GUV.ValuesDiffer)
    {
        nc.Block.Out();
        nc.GInterp1.RestoreDefaultValue(false);
        nc.GInterp2.RestoreDefaultValue(false);
    }
```

```
// Выключение коррекции
if (compensNeedOut && nc.GCompens == 40)
{
    nc.GCompens.Show();
    compensNeedOut = false;
}

// Вывод координат
nc.X1.v = cmd.Lower.EP.X;
nc.Y1.v = cmd.Lower.EP.Y;
nc.Z1.v = cmd.Lower.EP.Z;
switch (cmd.MotionMode)
{
    case CLDEDMMotionMode.Rapid:
        if (nc.X1.ValuesDiffer || nc.Y1.ValuesDiffer || nc.Z1.ValuesDiffer)
        {
            nc.GInterp1.v = 0;
        }
        break;
    case CLDEDMMotionMode.Plain2D:
    case CLDEDMMotionMode.Taper2D:
    case CLDEDMMotionMode.MultiProf4D:
        if (cmd.Lower.IsArc)
        {
            nc.GInterp1.v = cmd.Lower.ArcR > 0d ? 3 : 2;
            nc.I1.Show(cmd.Lower.Center.X - previousLowerPoint.X);
            nc.J1.Show(cmd.Lower.Center.Y - previousLowerPoint.Y);
        }
        else // Линия
        {
            nc.GInterp1.v = 1;
        }
        if (cmd.IsTaper2DMove)
        {
            nc.GTaper.v = cmd.TaperAngle > 0 ? 52 : 51;
            nc.A.Show(Abs(cmd.TaperAngle));
        }
        else if (cmd.IsMultiProf4DMove)
        {
            nc.Colon.Show();
            nc.X2.v = cmd.Upper.EP.X;
            nc.Y2.v = cmd.Upper.EP.Y;
            nc.Z2.v = cmd.Upper.EP.Z;
            if (cmd.Upper.IsArc)
            {
                nc.GInterp2.v = cmd.Upper.ArcR > 0d ? 3 : 2;
```

```
                    nc.I2.Show(cmd.Upper.Center.X - previousUpperPoint.X);
                    nc.J2.Show(cmd.Upper.Center.Y - previousUpperPoint.Y);
                }
                else // Линия
                {
                    nc.GInterp2.v = 1;
                }
                // Обкат углов (CornerR)
                if ((cmd.IsPlain2DMove || cmd.IsTaper2DMove) && cmd.RollMode
!= EDMRollMode.Off)
                {
                    nc.RollR1.Show(cmd.Lower.RollR);
                    nc.RollR2.Show(cmd.Upper.RollR);
                }
            }
            break;
        case CLDEDMMotionMode.CutsOnly4D:
            nc.GInterp1.v = 1;
            var d = (T3DPoint)cmd.Upper.EP - cmd.Lower.EP;
            nc.U.v = d.X;
            nc.V.v = d.Y;
            nc.W.v = d.Z;
            break;
    }

    // Выключение режима движения
    if (!cmd.IsTaper2DMove && nc.GTaper != 50)
    {
        nc.GTaper.v = 50;
    }
    if (!cmd.IsMultiProf4DMove && nc.G2Contour != 60)
    {
        nc.G2Contour.v = 60;
    }
    if (!cmd.IsCutsOnly4DMove && nc.GUV != 75)
    {
        nc.U.v = 0;
        nc.V.v = 0;
    }
    nc.Block.Out();
    if (!cmd.IsCutsOnly4DMove && nc.GUV != 75)
    {
        nc.GUV.v = 75;
        nc.U.Hide();
        nc.V.Hide();
    }
```

```
    nc.Block.Out();

    // Запоминание текущих координат
    previousLowerPoint = cmd.Lower.EP;
    if (!cmd.IsMultiProf4DMove && !cmd.IsCutsOnly4DMove) // 2D
    {
        previousUpperPoint = previousLowerPoint;
    }
    else // 4D
    {
        previousUpperPoint = cmd.Upper.EP;
    }
}
```

Сопутствующие ссылки

- Модель доступа к CLData

EFFECTOR - Активация рабочего органа

Отображение команды

Как команда отображается в списке команд CLData:

```
EFFECTOR N n, Value ON(71) | OFF(72)
```

Команда CLData EFFECTOR активирует или деактивирует рабочий орган станка или робота (лазер, форсунку, сварочную горелку и т. д.).

Доступ из sppx

Обработчик: `program Effector`

Параметр	Массив CLD	Имя CLD	Описание
N	CLD[1]	CLD.N	Уникальный идентификатор рабочего органа, который включается или выключается
Value	CLD[2]	CLD.Value	Тип команды: 71 - включение, 72 - выключение.

Пример:

```
program Effector
  ! cld[1] - идентификатор рабочего органа, cld[2] - 71 (ON) | 72 (OFF)
  if cld[2] = 71 then
    Output "DOUT M#(" + cld[1] + ")=ON"
  else
    Output "DOUT M#(" + cld[1] + ")=OFF"
end
```

Доступ из .NET

Обработчик:

```
public override void OnEffector(ICLDEffectorCommand cmd, CLDArray cld)
```

Наследование: `ICLDEffectorCommand : ICLDCommand`.

Читайте параметры из строго типизированного объекта `cmd` (рекомендуется); доступны также «сырой» массив `cld[i]` (те же индексы, что и в sppx) и именованные обёртки `cmd["..."]`. В таблице перечислены члены команды, включая унаследованные от базовых интерфейсов; общие члены `ICLDCommand` (навигация, `CLD`, `CLDFile`, `TechOperation`, ...) описаны в разделе Модель доступа к CLData.

Член	Тип	Описание
`cmd.IsOn`	`bool`	Возвращает "true", если эта команда EFFECTOR - команда включения, иначе "false".
`cmd.IsOff`	`bool`	Возвращает "true", если эта команда EFFECTOR - команда выключения, иначе "false".
`cmd.EffectorID`	`int`	Числовой идентификатор активируемого-деактивируемого рабочего органа.

Пример - `OnEffector` (из постпроцессора CRP_robot_DN):

```
public override void OnEffector(ICLDEffectorCommand cmd, CLDArray cld)
{
//    if (cmd.IsOn && !effectorIsOn) {
//        prg.WriteLine("DOUT M#(111)=ON");
//    } else if (cmd.IsOff  && effectorIsOn) {
//        prg.WriteLine("DOUT M#(111)=OFF");
//    }
    effectorIsOn = cmd.IsOn;
}
```

Сопутствующие ссылки

- Модель доступа к CLData

EXTCYCLE - Расширенный цикл

Отображение команды

Как команда отображается в списке команд CLData:

```
EXTCYCLE ON(71)|CALL(52)|OFF(72)
DRILL(163)|FACE(81)|TAP(168)|DEEP(153)|W5DDrill(481)|
         W5DBore5(485)|LATHEROUGH(401)|LATHETHREAD(403)|... {, a} {, b} {, c} ...
```

Команда расширенного цикла EXTCYCLE используется для обработки стандартных циклов. Ниже приведён список стандартных циклов, поддерживаемых командой EXTCYCLE:

- Токарный чистовой цикл G70, G73 - `LATHEFINISH(400)`
- Токарный черновой цикл G71, G72 - `LATHEROUGH(401)`
- Токарный цикл прорезки канавок G74, G75 - `LATHEGROOVE(402)`
- Токарный резьбонарезной цикл G76 - `LATHETHREAD(403)`
- Токарный однопроходный резьбонарезной цикл G92 - `LATHETHREADG92(404)`
- Токарный сверлильный цикл G81 - `DRILL(163)`
- Токарный сверлильный цикл G82 - `FACE(81)`
- Токарный резьбонарезной цикл метчиком G84 - `TAP(168)`
- Токарный сверлильный цикл с ломкой стружки - `BRKCHP(288)`
- Токарный сверлильный цикл с удалением стружки - `DEEP(153)`
- Сверлильный цикл G81 - `W5DDrill(481)`
- Сверлильный цикл G82 - `W5DFace(482)`
- Сверлильный цикл с удалением стружки G83 - `W5DChipRemoving(483)`
- Сверлильный цикл с ломкой стружки G73 - `W5DChipBreaking(473)`
- Цикл нарезания резьбы метчиком G84 - `W5DTap(484)`
- Сверлильный цикл G85 - `W5DBore5(485)`
- Сверлильный цикл G86 - `W5DBore6(486)`
- Сверлильный цикл G87 - `W5DBore7(487)`
- Сверлильный цикл G88 - `W5DBore8(488)`
- Сверлильный цикл G89 - `W5DBore9(489)`
- Цикл фрезерования резьбы - `W5DThreadMill(490)`
- Цикл выборки отверстия - `W5DHolePocketing(491)`
- Измерительный цикл - `WProbing(500)`

Доступ из sppx

Обработчик: `program ExtCycle`

Массив `cld` читается по индексу или по имени: в sppx — по типизированному короткому имени (например, `cld.X`), в .NET — по тому же имени в виде строкового ключа (например, `cld["X"]`).

Параметр	Массив CLD	Имя CLD	Описание
ON(71) CALL(52) OFF(72)	CLD[1]	CLD.SubCmd	Модификатор команды: ON(71) – включить стандартный цикл, CALL(52) – вызвать стандартный цикл, OFF(72) – выключить стандартный цикл.
LATHEFINISH(400) LATHEROUGH(401) LATHEGROOVE(402) LATHETHREAD(403) DRILL(163) FACE(81) TAP(168) BRKCHP(288) DEEP(153) W5DDrill(481) W5DFace(482) W5DChipRemoving(483) W5DChipBreaking(473) W5DTap(484) W5DBore5(485) W5DBore6(486) W5DBore7(487) W5DBore8(488) W5DBore9(489) W5DThreadMill(490) W5DHolePocketing(491) WProbing(500)	CLD[2]	CLD.SubType	Идентификатор типа цикла.

Параметр	Массив CLD	Имя CLD	Описание
a, b, c, d …	CLD[3] – CLD[258]	CLD.CLParams(1) – CLD.CLParams(256)	Список параметров цикла.

Первый параметр задаёт запуск, вызов и отмену стандартного цикла.

CAM-система обычно формирует последовательность технологических команд с командой EXTCYCLE, подобную следующей:

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

Список параметров зависит от типа цикла.

Пример (из постпроцессора Fanuc (30i)_Mill):

```
program ExtCycle
  if cld[1] = 71 then CycleOn = 1                    ! цикл включён
  else if cld[1] = 72 then begin                     ! цикл выключен
    if (CLD.SubType >= 473) and (CLD.SubType <= 489) then Cycle = 80
    OutBlock
  end else if cld[1] = 52 then begin                 ! вызов цикла
    if cld[10] > 0 then
      if cld[9] = 0 then Fr = cld[10]*S else Fr = cld[10]   ! единицы /
значение подачи
  end
```

Доступ из .NET

Помимо базового `OnExtCycle`, команда диспетчеризуется (по семейству циклов) в один из специализированных обработчиков ниже:

```
public override void OnExtCycle(ICLDExtCycleCommand cmd, CLDArray cld)
// базовый
public override void OnHoleExtCycle(ICLDExtCycleCommand cmd, CLDArray cld)    //
циклы отверстий
// а также, по семейству циклов: OnTurnExtCycle, OnProbeExtCycle,
OnWeldingExtCycle, OnPickAndPlaceExtCycle
```

Наследование: `ICLDExtCycleCommand : ICLDCommand`.

Читайте параметры из строго типизированного объекта `cmd` (рекомендуется); также доступны «сырой» массив `cld[i]` (те же индексы, что и в sppx) и именованные помощники `cmd[...]`. В таблице перечислены члены команды, в том числе унаследованные от базовых интерфейсов; общие члены `ICLDCommand` (навигация, `CLD`, `CLDFile`, `TechOperation`, ...) описаны в разделе Модель доступа к CLData. «Сырой» массив **`cld[i]`** доступен с теми же индексами, что и в sppx, а параметры также достижимы по числовому коду через `cmd.Prm`.

Член	Тип	Описание

`cmd.IsOn`
`cmd.IsOff`
`cmd.IsCall`
	bool	True, если это команда включения / выключения / вызова (CALL) цикла.

`cmd.CycleType`	int	Числовой код типа цикла (простое сверление, глубокое сверление, нарезание резьбы метчиком, наружное точение, ...). См. константы в классе `CLDCycle`.
`cmd.Prm`	`ICodeParametersOfCLDCommand`	Список параметров; значение получается по его числовому коду.

`cmd.Prm` предоставляет `ParamCount`, `Code(i)`, `ParamType(i)`, `IndexOfCode(code)` и индексаторы значений `Prm.Flt[code]` / `Prm.Int[code]` / `Prm.Bol[code]` / `Prm.Str[code]`.

Пример (из постпроцессора Sinumerik 840D mill — обратите внимание на доступ через `cld[i]`):

```
public override void OnHoleExtCycle(ICLDExtCycleCommand cmd, CLDArray cld)
{
    if (cmd.IsOn)
    {
```

```
        nc.GFeed.v = cld[9] == 1 ? 94 : 95;   // единицы рабочей подачи (тот же
индекс cld, что и в sppx)
        nc.Feed.v = cld[10];                  // значение рабочей подачи
        nc.Block.Out();
    }
    else if (cmd.IsOff)
        nc.G.Show(80);                        // отмена цикла
    // else if (cmd.IsCall) диспетчеризация по cmd.CycleType ...
}
```

Сопутствующие ссылки

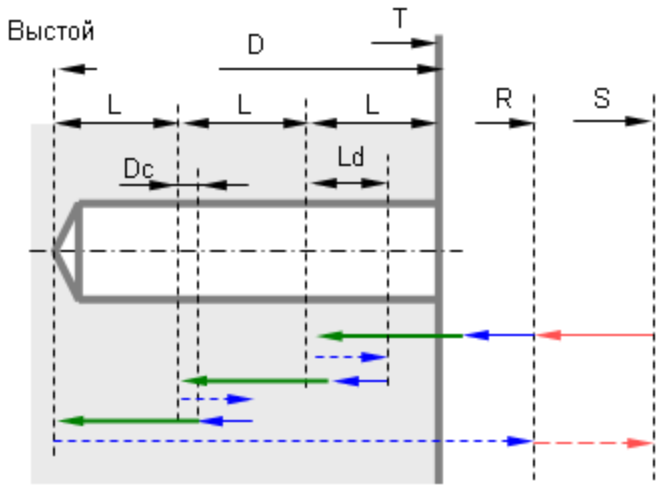
- Модель доступа к CLData

EXTCYCLE BRKCHP - Токарный сверлильный цикл с ломкой стружки

Цикл глубокого сверления с отводами инструмента для ломки стружки `BRKCHP(288)` применяется для обработки осевых отверстий неподвижным инструментом в заготовке, вращаемой шпинделем. Цикл состоит из:

1. Ускоренный подвод к уровню осевой безопасной плоскости S.
2. Перемещение к уровню возврата R.
3. Перемещение к уровню предыдущего сверления (для первого сверления равен уровню верха отверстия) с замедлением (Dc).
4. Перемещение на рабочей подаче на расстояние длины L.
5. Задержка на заданное время.
6. Отвод инструмента на заданное расстояние отвода Ld.
7. Повторение шагов с 3 по 6 до достижения полной глубины отверстия.
8. Возврат на безопасный уровень R.
9. Ускоренный возврат к уровню безопасной плоскости S.

Шаг сверления уменьшается на каждой итерации на заданную величину, если величина уменьшения не равна нулю.



Параметры

Массив CLD	Имя CLD	Описание
CLD[1]	CLD.SubCmd	Тип команды: ON(71) – включение стандартного цикла, CALL(52) – вызов стандартного цикла, OFF(72)

Массив CLD	Имя CLD	Описание
		– выключение стандартного цикла.
CLD[2]	CLD.SubType	Идентификатор типа стандартного цикла: BRKCHP(288).
CLD[3]	CLD.CLParams(1)	Верхний уровень отверстия (T).
CLD[4]	CLD.CLParams(2)	Глубина отверстия от верхнего уровня (D).
CLD[5]	CLD.CLParams(3)	Уровень возврата (R).
CLD[6]	CLD.CLParams(4)	Безопасный уровень (S).
CLD[7]	CLD.CLParams(5)	Задержка на нижнем уровне в секундах (Dwell).
CLD[8]	CLD.CLParams(6)	Тип 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 при сверлении (L)
CLD[12]	CLD.CLParams(10)	Замедление (Dc)
CLD[13]	CLD.CLParams(11)	Отвод для ломки (Ld)
CLD[14]	CLD.CLParams(12)	Уменьшение шага L на следующей итерации

Доступ из .NET

У этого цикла нет выделенного обработчика семейства — он доставляется через базовый `OnExtCycle` (диспетчеризация по `cmd.CycleType` / `cld[2]`). См. Расширенный цикл (EXTCYCLE).

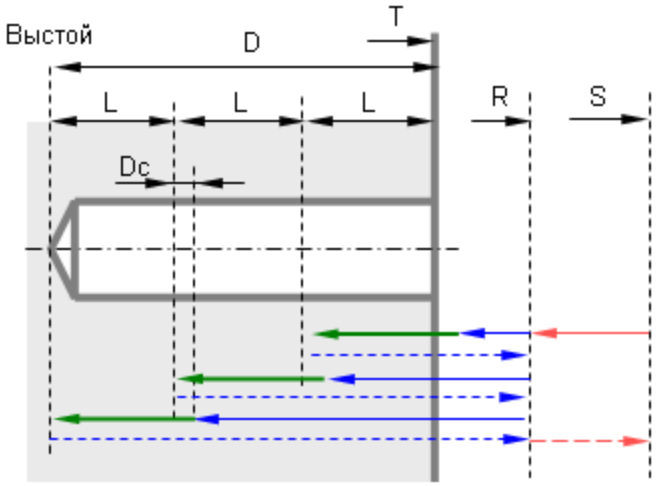
Сопутствующие ссылки

- Расширенный цикл (EXTCYCLE)
- Модель доступа к CLData

EXTCYCLE DEEP - Токарный сверлильный цикл с удалением стружки

Цикл глубокого сверления с отводом инструмента для удаления стружки `DEEP(153)` применяется для обработки осевых отверстий неподвижным инструментом. Цикл включает:

1. Ускоренное перемещение к уровню безопасной плоскости S.
2. Перемещение к уровню возврата R.
3. Перемещение на глубину предыдущей итерации (для первой итерации равна уровню верха отверстия) с замедлением Dc.
4. Перемещение на рабочей подаче на расстояние длины L.
5. Задержка на заданное время.
6. Возврат на уровень возврата R.
7. Повторение шагов 3 - 6 до достижения полной глубины отверстия.
8. Ускоренный отвод к уровню безопасной плоскости S.



Параметры

Массив CLD	Имя CLD	Описание
CLD[1]	CLD.SubCmd	Тип команды: ON(71) – включение стандартного цикла, CALL(52) – вызов стандартного цикла, OFF(72) – выключение стандартного цикла.
CLD[2]	CLD.SubType	Идентификатор типа стандартного цикла: DEEP(153).

Массив CLD	Имя CLD	Описание
CLD[3]	CLD.CLParams(1)	Верхний уровень отверстия (T).
CLD[4]	CLD.CLParams(2)	Глубина отверстия от верхнего уровня (D).
CLD[5]	CLD.CLParams(3)	Уровень возврата (R).
CLD[6]	CLD.CLParams(4)	Безопасный уровень (S).
CLD[7]	CLD.CLParams(5)	Задержка на нижнем уровне в секундах (Dwell).
CLD[8]	CLD.CLParams(6)	Тип 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 при сверлении (L)
CLD[12]	CLD.CLParams(10)	Замедление (Dc)
CLD[13]	CLD.CLParams(11)	Зарезервировано
CLD[14]	CLD.CLParams(12)	Уменьшение шага L на следующей итерации

Доступ из .NET

У этого цикла нет выделенного обработчика семейства — он доставляется через базовый `OnExtCycle` (диспетчеризация по `cmd.CycleType` / `cld[2]`). См. Расширенный цикл (EXTCYCLE).

Сопутствующие ссылки

- Расширенный цикл (EXTCYCLE)
- Модель доступа к CLData

EXTCYCLE DRILL - Токарный сверлильный цикл (G81)

Токарный сверлильный стандартный цикл `DRILL(163)` G81 сверлит осевые отверстия неподвижным инструментом внутри вращаемой заготовки. Инструмент подводится к отверстию на уровне безопасной плоскости, на ускоренной подаче опускается к уровню возврата, обрабатывает отверстие на рабочей подаче и ускоренно отводится к уровню безопасной плоскости.

Параметры

Массив CLD	Имя CLD	Описание
CLD[1]	CLD.SubCmd	Тип команды: ON(71) – включение стандартного цикла, CALL(52) – вызов стандартного цикла, OFF(72) – выключение стандартного цикла.
CLD[2]	CLD.SubType	Идентификатор типа стандартного цикла: DRILL(163).
CLD[3]	CLD.CLParams(1)	Верхний уровень отверстия (T).
CLD[4]	CLD.CLParams(2)	Глубина отверстия, измеренная относительно верхнего уровня (D).
CLD[5]	CLD.CLParams(3)	Уровень возврата (R).
CLD[6]	CLD.CLParams(4)	Безопасный уровень (S).
CLD[7]	CLD.CLParams(5)	Зарезервировано.
CLD[8]	CLD.CLParams(6)	Единицы подачи: 0 – мм/об, 1 – мм/мин.
CLD[9]	CLD.CLParams(7)	Рабочая подача.

Массив CLD	Имя CLD	Описание
CLD[10]	CLD.CLParams(8)	Подача отвода.

Доступ из .NET

У этого цикла нет выделенного обработчика семейства — он доставляется через базовый `OnExtCycle` (диспетчеризация по `cmd.CycleType` / `cld[2]`). См. Расширенный цикл (EXTCYCLE).

Сопутствующие ссылки

- Расширенный цикл (EXTCYCLE)
- Модель доступа к CLData

EXTCYCLE FACE - Токарный сверлильный цикл (G82)

Токарный сверлильный стандартный цикл `FACE(81)` G82 сверлит осевые отверстия неподвижным инструментом внутри вращаемой заготовки. Инструмент подводится к отверстию на уровне безопасной плоскости, на ускоренной подаче опускается к уровню возврата, обрабатывает отверстие на рабочей подаче, выдерживает задержку на дне отверстия и ускоренно отводится к уровню безопасной плоскости.

Параметры

Массив CLD	Имя CLD	Описание
CLD[1]	CLD.SubCmd	Тип команды: ON(71) – включение стандартного цикла, CALL(52) – вызов стандартного цикла, OFF(72) – выключение стандартного цикла.
CLD[2]	CLD.SubType	Идентификатор типа стандартного цикла: FACE(81).
CLD[3]	CLD.CLParams(1)	Верхний уровень отверстия (T).
CLD[4]	CLD.CLParams(2)	Глубина отверстия, измеренная относительно верхнего уровня (D).
CLD[5]	CLD.CLParams(3)	Уровень возврата (R).
CLD[6]	CLD.CLParams(4)	Безопасный уровень (S).
CLD[7]	CLD.CLParams(5)	Задержка на уровне дна отверстия (секунды).
CLD[8]	CLD.CLParams(6)	Единицы подачи: 0 – мм/об, 1 – мм/мин.
CLD[9]	CLD.CLParams(7)	Рабочая подача.

Массив CLD	Имя CLD	Описание
CLD[10]	CLD.CLParams(8)	Подача отвода.

Доступ из .NET

У этого цикла нет выделенного обработчика семейства — он доставляется через базовый `OnExtCycle` (диспетчеризация по `cmd.CycleType` / `cld[2]`). См. Расширенный цикл (EXTCYCLE).

Сопутствующие ссылки

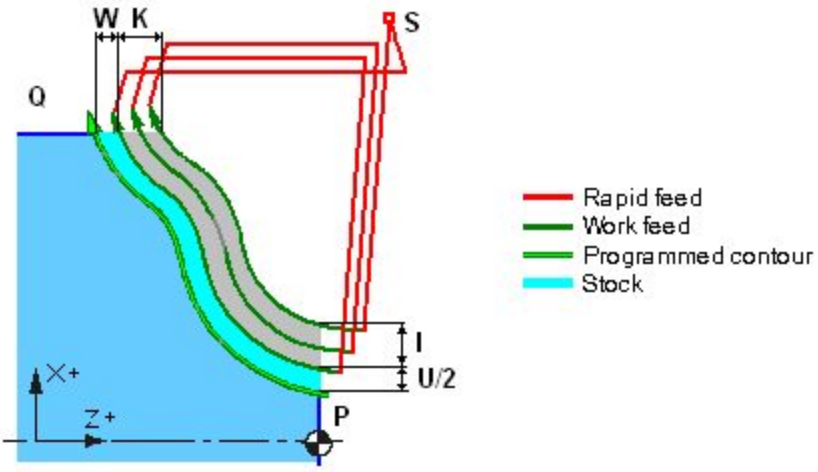
- Расширенный цикл (EXTCYCLE)
- Модель доступа к CLData

EXTCYCLE LATHEFINISH - Токарный чистовой цикл (G70, G73)

Стандартный цикл `LATHEFINISH(400)` (G73) применяется для черновой обработки предварительно сформированной (например, отлитой) заготовки. При обработке стандартным циклом предполагается, что материал снят или отсутствует на заданном расстоянии от запрограммированной траектории инструмента (блок PQ).

При обработке циклом значения F, S и T задаются до кадра G73 или непосредственно в кадре G73. Команды подачи (F), частоты вращения шпинделя (S) и смены инструмента (T) между строками P и Q игнорируются.

Смещение первого чернового прохода определяется как (U/2 + I) по оси X и (W + K) по оси Z. Каждый последующий черновой проход приближается к контуру последнего чернового прохода на (I/(D-1)) по оси X и (K/(D-1)) по оси Z. Последний черновой проход всегда оставляет припуск U/2 по оси X и W по оси Z.



Стандартный цикл LATHEFINISH также используется для формирования токарного чистового цикла G70 вдоль контура. Если припуски равны нулю и выполняется только один проход, то вызывается токарный чистовой цикл G70.

Токарные циклы предполагают, что задан запрограммированный контур детали PQ. CAM-система передаёт контур в виде подпрограммы УП. Номер подпрограммы указывается в параметре `CLD[3]`.

Параметры

Массив CLD	Имя CLD	Описание
CLD[1]	CLD.SubCmd	Тип команды: ON(71) — включение стандартного

Массив CLD	Имя CLD	Описание
		цикла, CALL(52) — вызов стандартного цикла, OFF(72) — выключение стандартного цикла.
CLD[2]	CLD.Subtype	Идентификатор типа стандартного цикла: LATHEFINISH(400)
CLD[3]	CLD.CLParams(1)	Номер подпрограммы УП, содержащей обрабатываемый контур PQ.
CLD[4]	CLD.CLParams(2)	Количество проходов (D). Если требуется один проход, то D=1 и выполняется G70. Если D>1, то выполняется цикл G73.
CLD[5]	CLD.CLParams(3)	Знаковая толщина слоя, направленная по оси Z (K). Равна нулю для G70.
CLD[6]	CLD.CLParams(4)	Знаковая толщина по оси X (I). Равна нулю для цикла G70.
CLD[7]	CLD.CLParams(5)	Знаковый припуск по оси Z (W). Равен нулю для цикла G70.
CLD[8]	CLD.CLParams(6)	Знаковый припуск по оси X (U/2). Равен нулю для цикла G70.
CLD[11]	CLD.CLParams(9)	Тип обработки: 0 — торцевая (Face), 1 — наружная (OD), 2 — внутренняя (ID). «Face» — очень редкий случай. Возникает только если

Массив CLD	Имя CLD	Описание
		контур не меняется вдоль осевой координаты с абсолютной точностью. Чаще всего, даже при торцевом точении, имеется ненулевой отвод вдоль оси, что превращает торцевое точение в наружное (OD) или внутреннее (ID).
CLD[12]	CLD.CLParams(10)	Наличие чистового прохода (G70): 0 — без чистового прохода, 1 — с чистовым проходом.
CLD[13]	CLD.CLParams(11)	Эквидистантный припуск на контур

Если «знаковый» параметр положительный, он откладывается в положительном направлении соответствующей оси, иначе — в отрицательном направлении.

Доступ из .NET

Помимо базового `OnExtCycle`, этот цикл доставляется через обработчик семейства токарных циклов `OnTurnExtCycle`:

```
public override void OnTurnExtCycle(ICLDExtCycleCommand cmd, CLDArray cld)
```

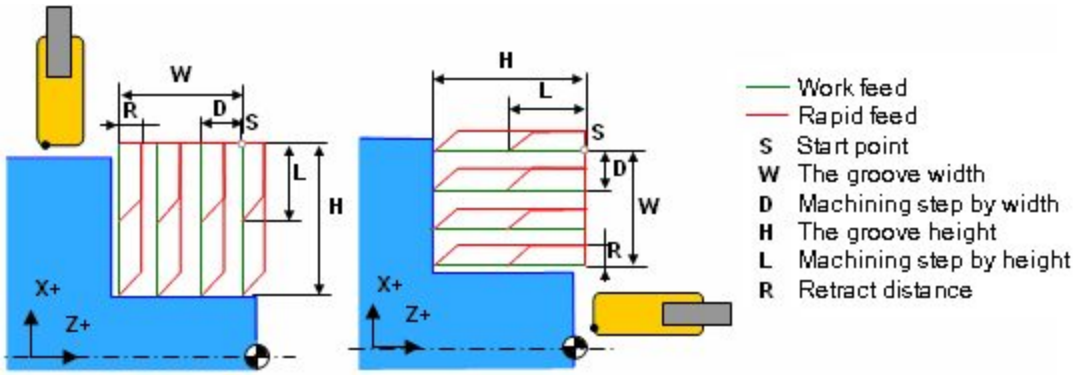
Общая модель .NET описана в разделе Расширенный цикл (EXTCYCLE).

Сопутствующие ссылки

- Расширенный цикл (EXTCYCLE)
- Модель доступа к CLData

EXTCYCLE LATHEGROOVE - Токарный цикл прорезки канавок (G74, G75)

Цикл `LATHEGROOVE(402)` реализует стандартный цикл прорезки цилиндрических канавок G74 и стандартный цикл прорезки торцевых канавок G75. Форма канавки прямоугольная и определяется её размерами: шириной и высотой. Резание выполняется последовательными рабочими проходами инструмента с шагом по осям Z и X.



Параметры

Массив CLD	Имя CLD	Описание
CLD[1]	CLD.SubCmd	Тип команды: ON(71) – включение стандартного цикла, CALL(52) – вызов стандартного цикла, OFF(72) – выключение стандартного цикла.
CLD[2]	CLD.SubType	Идентификатор типа стандартного цикла: LATHEGROOVE(402)
CLD[3]	CLD.CLParams(1)	Тип канавки: 0 – канавка точением (G75), 1 – торцевая канавка (G74)
CLD[4]	CLD.CLParams(2)	Знаковая ширина канавки, измеренная относительно начальной точки (W)
CLD[5]	CLD.CLParams(3)	Знаковая высота канавки, измеренная относительно

Массив CLD	Имя CLD	Описание
		начальной точки (H)
CLD[6]	CLD.CLParams(4)	Абсолютная величина шага обработки по ширине (D)
CLD[7]	CLD.CLParams(5)	Абсолютная величина шага обработки по высоте (глубина ломки стружки, L)
CLD[8]	CLD.CLParams(6)	Абсолютное расстояние бокового отвода перед возвратом инструмента (R)
CLD[9]	CLD.CLParams(7)	Абсолютная величина отвода для ломки стружки.
CLD[10]	CLD.CLParams(8)	Задержка на дне в секундах
CLD[11]	CLD.CLParams(9)	Режим ломки стружки: 0 - не использовать, 1 - при первом врезании, 2 - при каждом врезании.
CLD[12]	CLD.CLParams(10)	Черновой припуск по ширине
CLD[13]	CLD.CLParams(11)	Черновой припуск по высоте

Если «знаковый» параметр положительный, он откладывается в положительном направлении соответствующей оси, иначе — в отрицательном направлении.

Доступ из .NET

Помимо базового `OnExtCycle`, этот цикл доставляется через обработчик семейства токарных циклов `OnTurnExtCycle`:

```
public override void OnTurnExtCycle(ICLDExtCycleCommand cmd, CLDArray cld)
```

Общая модель .NET описана в разделе Расширенный цикл (EXTCYCLE).

Сопутствующие ссылки

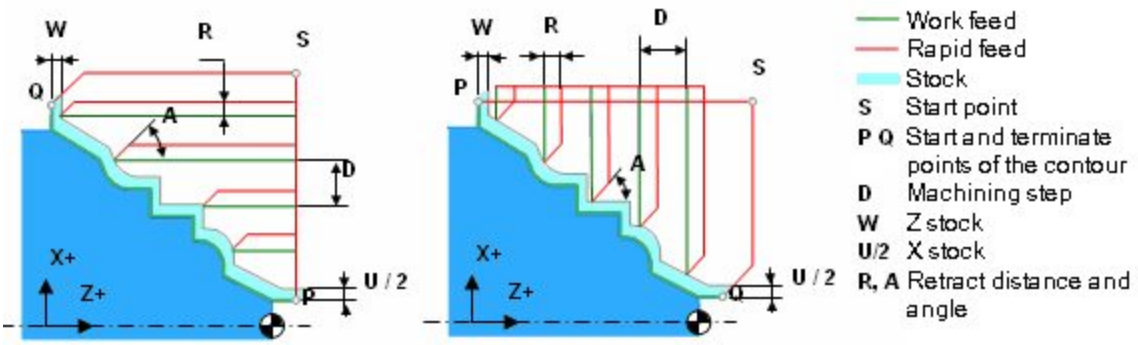
- Расширенный цикл (EXTCYCLE)
- Модель доступа к CLData

EXTCYCLE LATHEROUGH - Токарный черновой цикл (G71, G72)

Стандартный цикл `LATHEROUGH(401)` G71 выполняет черновую обработку заготовки по заданной форме детали. В токарном черновом цикле G71 используются команды F, S и T, указанные в строке G71 или уже заданные до строки G71. Цикл G71 обрабатывает два типа траекторий обработки. Если в запрограммированной траектории ось X не меняет направление, то это траектория первого типа (тип I). Второй тип траектории (тип II) допускает смену направления оси X. Направление оси Z не должно меняться для обоих типов.

Стандартный цикл G73 аналогичен G71, но применяется для обработки торца детали.

Последнее перемещение для обоих циклов — возврат в исходную позицию S. Как показано на рисунке, S — это положение инструмента в момент вызова цикла.



Токарные циклы предполагают, что задан запрограммированный контур детали PQ. CAM-система передаёт контур в виде подпрограммы УП. Номер подпрограммы указывается в параметре `CLD[3]`.

Если параметр CLD[12]=1, то после чернового цикла G71 или G72 может быть вызван чистовой проход вдоль контура — цикл G70.

Параметры

Массив CLD	Имя CLD	Описание
CLD[1]	CLD.SubCmd	Тип команды: ON(71) – включение стандартного цикла, CALL(52) – вызов стандартного цикла, OFF(72) – выключение стандартного цикла.

Массив CLD	Имя CLD	Описание
CLD[2]	CLD.SubType	Идентификатор типа стандартного цикла: LATHEROUGH(401)
CLD[3]	CLD.CLParams(1)	Номер подпрограммы, обрабатывающей 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)	Переключатель режима перекрытия: 0 – опция перекрытия выключена, 1 – опция перекрытия включена (траектория содержит дополнительные перемещения инструмента вдоль обрабатываемого контура для срезания гребешков)
CLD[7]	CLD.CLParams(5)	Знаковый припуск по Z (W)
CLD[8]	CLD.CLParams(6)	Знаковый припуск по X (U/2)
CLD[9]	CLD.CLParams(7)	Расстояние отвода (R)
CLD[10]	CLD.CLParams(8)	Угол отвода (A)
CLD[11]	CLD.CLParams(9)	Тип обработки: 1 – наружная (OD), 2 – внутренняя (ID)
CLD[12]	CLD.CLParams(10)	Наличие чистового прохода (G70): 0 - без чистового прохода, 1 - с чистовым проходом.

Массив CLD	Имя CLD	Описание
CLD[13]	CLD.CLParams(11)	Эквидистантный припуск на контур

Если «знаковый» параметр положительный, он откладывается в положительном направлении соответствующей оси, иначе — в отрицательном направлении.

Доступ из .NET

Помимо базового `OnExtCycle`, этот цикл доставляется через обработчик семейства токарных циклов `OnTurnExtCycle`:

```
public override void OnTurnExtCycle(ICLDExtCycleCommand cmd, CLDArray cld)
```

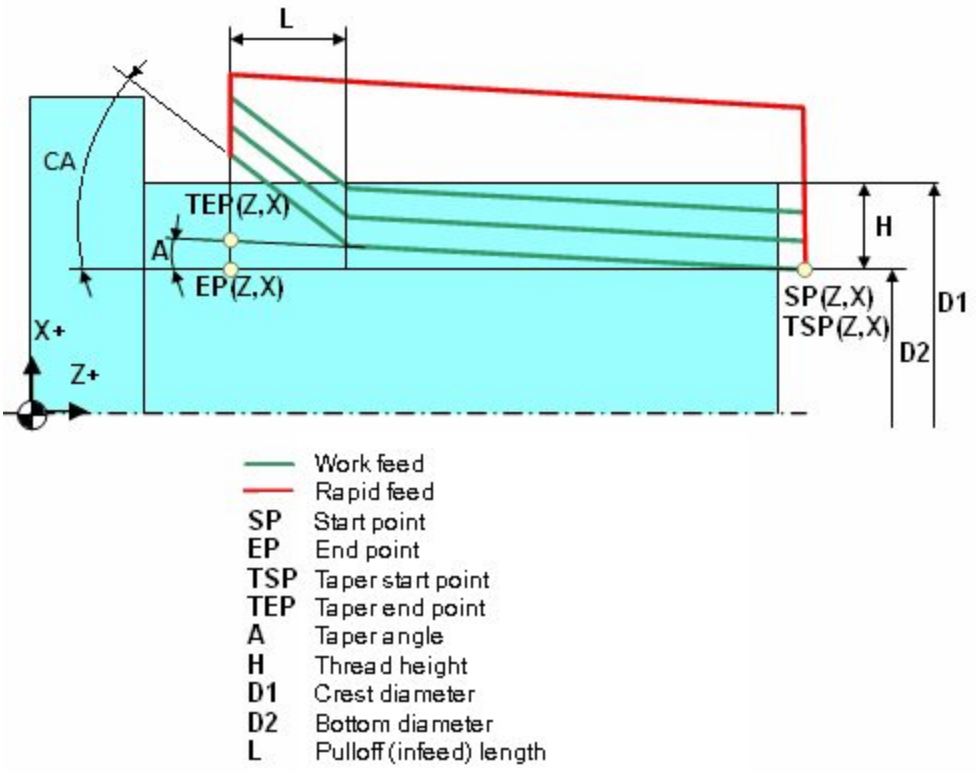
Общая модель .NET описана в разделе Расширенный цикл (EXTCYCLE).

Сопутствующие ссылки

- Расширенный цикл (EXTCYCLE)
- Модель доступа к CLData

EXTCYCLE LATHETHREAD - Токарный резьбонарезной цикл (G76)

Токарный многопроходный резьбонарезной стандартный цикл `LATHETHREAD(403)` G76 используется для формирования однострочной команды, задающей все параметры нарезания резьбы. Требуемая глубина резьбы достигается автоматическими многократными проходами. Параметры цикла включают координаты начальной и конечной точек резьбы, угол конуса (для конической резьбы), размеры фаски, углы профиля, глубину резьбы, число проходов, стратегию врезания и т. д.



Параметры

Массив CLD	Имя CLD	Описание
CLD[1]	CLD.SubCmd	Тип команды: ON(71) – включение стандартного цикла, CALL(52) – вызов стандартного цикла, OFF(72) – выключение стандартного цикла.

Массив CLD	Имя CLD	Описание
CLD[2]	CLD.SubType	Идентификатор типа стандартного цикла: LATHETHREAD(403).
CLD[3]	CLD.CLParams(1)	Направление резьбы: 0 – левая, 1 – правая.
CLD[4]	CLD.CLParams(2)	Ориентация резьбы: 0 – наружный диаметр, 1 – внутренний диаметр, 2 – торец.
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 конической резьбы (TSP.Z).
CLD[11]	CLD.CLParams(9)	Координата X начальной точки конической резьбы (TSP.X).
CLD[12]	CLD.CLParams(10)	Координата Z конечной точки конической резьбы (TEP.Z).
CLD[13]	CLD.CLParams(11)	Координата X конечной точки конической резьбы

Массив CLD	Имя CLD	Описание
		(TEP.X).
CLD[14]	CLD.CLParams(12)	Длина сбега при завершении резьбы (L).
CLD[15]	CLD.CLParams(13)	Тип резьбы: 0 – специальная, 1 – метрическая, 2 – цилиндрическая трубная, 3 – трапецеидальная, 4 – упорная, 5 – круглая, 6 – коническая трубная, 7 – внутренняя коническая, 8 – внутренняя цилиндрическая, 9 – коническая дюймовая.
CLD[16]	CLD.CLParams(14)	Диаметр вершины (D1).
CLD[17]	CLD.CLParams(15)	Диаметр впадины (D2).
CLD[18]	CLD.CLParams(16)	Высота резьбы (H).
CLD[19]	CLD.CLParams(17)	Угол профиля резьбы между двумя соседними гранями резьбы
CLD[20]	CLD.CLParams(18)	Угол профиля резьбы между вертикальной линией и

Массив CLD	Имя CLD	Описание
		гранью резьбы
CLD[21]	CLD.CLParams(19)	Число заходов резьбы.
CLD[22]	CLD.CLParams(20)	Режим задания шага резьбы: 0 – по расстоянию, 1 – по числу ниток на единицу длины.
CLD[23]	CLD.CLParams(21)	Величина шага резьбы.
CLD[24]	CLD.CLParams(22)	Последовательность резьбы: 0 – радиальная 1 – по боковой грани 2 – попеременно по боковым граням
CLD[25]	CLD.CLParams(23)	Метод расчёта глубины резания: 0 – сохранять постоянной площадь резания

Массив CLD	Имя CLD	Описание
		1 – сохранять постоянной глубину резания
CLD[26]	CLD.CLParams(24)	Режим задания глубины резания: 0 – по глубине первого прохода, 1 – по числу проходов.
CLD[27]	CLD.CLParams(25)	Глубина первого прохода (используется вместе с методом задания глубины резания «0»).
CLD[28]	CLD.CLParams(26)	Число проходов по глубине (используется вместе с режимом задания глубины резания «1»).
CLD[29]	CLD.CLParams(27)	Глубина чистового прохода.
CLD[30]	CLD.CLParams(28)	Число выхаживающих проходов.
CLD[31]	CLD.CLParams(29)	Минимальная глубина чернового прохода

Массив CLD	Имя CLD	Описание
CLD[32]	CLD.CLParams(30)	Угол сбега при завершении резьбы в градусах (CA).
CLD[33]	CLD.CLParams(31)	Начальный угол шпинделя в градусах (для многозаходных резьб).
CLD[34]	CLD.CLParams(32)	Длина захода резьбы вдоль оси вращения (как размер L на рисунке, но в начале резьбы).
CLD[35]	CLD.CLParams(33)	Угол врезания инструмента в градусах, который зависит также от CLD[24].
CLD[36]	CLD.CLParams(34)	Длина отвода резьбы вдоль оси вращения (как размер L на рисунке, но не зависит от угла отвода)

Доступ из .NET

Помимо базового `OnExtCycle`, этот цикл доставляется через обработчик семейства токарных циклов `OnTurnExtCycle`:

```
public override void OnTurnExtCycle(ICLDExtCycleCommand cmd, CLDArray cld)
```

Общая модель .NET описана в разделе Расширенный цикл (EXTCYCLE).

Сопутствующие ссылки

- Расширенный цикл (EXTCYCLE)
- Модель доступа к CLData

EXTCYCLE LATHETHREADG92(404) - Токарный однопроходный резьбонарезной цикл (G92)

Токарный однопроходный резьбонарезной цикл G92 `LATHETHREADG92(404)` (в разных системах ЧПУ может обозначаться G92, G78, G21 и т. д.) генерирует замкнутую последовательность перемещений для однопроходного нарезания резьбы. Иллюстрация выполнения показана ниже. Перед вызовом цикла инструмент находится в точке Start. Цикл вызывается одним кадром управляющей программы, в котором задаются конечная точка, шаг резьбы, конусность, размер фаски и т. д. В результате этого кадра инструмент из точки Start опустится в точку TSP, выполнит нарезание резьбы до точки TEP и вернётся в точку Start. Поскольку нарезание резьбы обычно выполняется в несколько проходов, управляющая программа часто состоит из нескольких вызовов цикла с различными значениями диаметра резьбы:

```
...
X60.0 Z20.0 M08
G01 Z10.0 F1.0 (Подвод к точке Start)
G92 X29.4 Z-52.0 F2.0 (Вызов цикла)
X28.9 (Модальный вызов цикла G92 с другим значением диаметра)
X28.5 (Модальный вызов цикла G92 с другим значением диаметра)
X28.1 (Модальный вызов цикла G92 с другим значением диаметра)
X27.8 (Модальный вызов цикла G92 с другим значением диаметра)
X27.56 (Модальный вызов цикла G92 с другим значением диаметра)
X27.36 (Модальный вызов цикла G92 с другим значением диаметра)
X27.26 (Модальный вызов цикла G92 с другим значением диаметра)
G00 X200.0Z150.0M09 (Возврат инструмента)
...
```

Параметры

Массив CLD	Имя CLD	Описание
CLD[1]	CLD.SubCmd	Тип команды: ON(71) – включение стандартного цикла, CALL(52) – вызов стандартного цикла, OFF(72) – выключение стандартного цикла.
CLD[2]	CLD.SubType	Идентификатор типа стандартного цикла: LATHETHREADG92(404).

Массив CLD	Имя CLD	Описание
CLD[4]	CLD.CLParams(2)	Ориентация резьбы: 0 – наружный диаметр, 1 – внутренний диаметр, 2 – торец.
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 конической резьбы (TSP.Z).
CLD[11]	CLD.CLParams(9)	Координата X начальной точки конической резьбы (TSP.X).
CLD[12]	CLD.CLParams(10)	Координата Z конечной точки конической резьбы (TEP.Z).
CLD[13]	CLD.CLParams(11)	Координата X конечной точки конической резьбы (TEP.X).
CLD[14]	CLD.CLParams(12)	Длина сбега при завершении резьбы (L).
CLD[21]	CLD.CLParams(19)	Число заходов резьбы.

Массив CLD	Имя CLD	Описание
CLD[22]	CLD.CLParams(20)	Режим задания шага резьбы: 0 – по расстоянию, 1 – по числу ниток на единицу длины.
CLD[23]	CLD.CLParams(21)	Величина шага резьбы.
CLD[32]	CLD.CLParams(30)	Угол сбега при завершении резьбы в градусах (CA).
CLD[33]	CLD.CLParams(31)	Начальный угол шпинделя в градусах (для многозаходных резьб).

Доступ из .NET

Помимо базового `OnExtCycle`, этот цикл доставляется через обработчик семейства токарных циклов `OnTurnExtCycle`:

```
public override void OnTurnExtCycle(ICLDExtCycleCommand cmd, CLDArray cld)
```

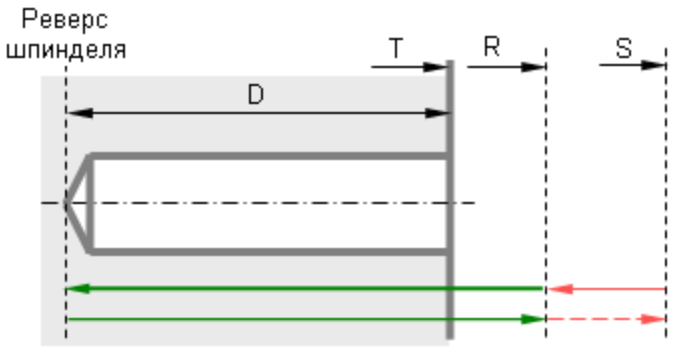
Общая модель .NET описана в разделе Расширенный цикл (EXTCYCLE).

Сопутствующие ссылки

- Расширенный цикл (EXTCYCLE)
- Модель доступа к CLData

EXTCYCLE TAP - Токарный резьбонарезной цикл метчиком (G84)

Токарный резьбонарезной цикл метчиком `TAP(168)` G84 применяется для нарезания резьбы неподвижным метчиком в осевом отверстии заготовки, вращаемой шпинделем. Инструмент подводится к оси отверстия на уровне безопасной плоскости, ускоренно перемещается к уровню возврата, затем на рабочей подаче, равной шагу резьбы, нарезает резьбу на заданную глубину. На дне отверстия направление вращения шпинделя реверсируется, и инструмент отводится на рабочей подаче к уровню возврата, где восстанавливаются направление и частота вращения шпинделя.



Параметры

Массив CLD	Имя CLD	Описание
CLD[1]	CLD.SubCmd	Тип команды: ON(71) – включение стандартного цикла, CALL(52) – вызов стандартного цикла, OFF(72) – выключение стандартного цикла.
CLD[2]	CLD.SubType	Идентификатор типа стандартного цикла: TAP(168).
CLD[3]	CLD.CLParams(1)	Верхний уровень отверстия.
CLD[4]	CLD.CLParams(2)	Глубина отверстия от верхнего уровня.
CLD[5]	CLD.CLParams(3)	Уровень возврата.

Массив CLD	Имя CLD	Описание
CLD[6]	CLD.CLParams(4)	Безопасный уровень.
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.

Доступ из .NET

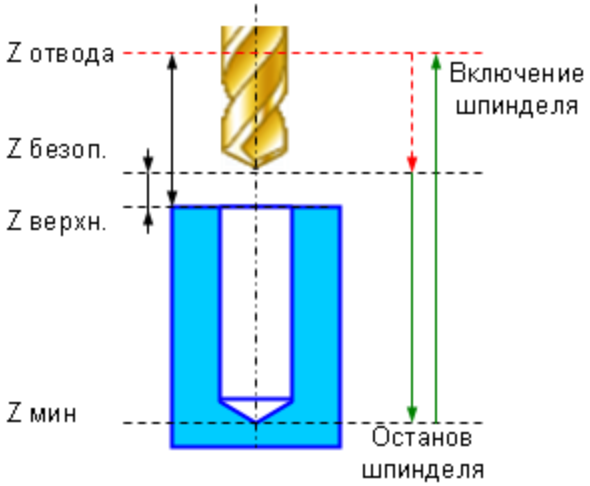
У этого цикла нет выделенного обработчика семейства — он доставляется через базовый `OnExtCycle` (диспетчеризация по `cmd.CycleType` / `cld[2]`). См. Расширенный цикл (EXTCYCLE).

Сопутствующие ссылки

- Расширенный цикл (EXTCYCLE)
- Модель доступа к CLData

EXTCYCLE W5DBore5 - Сверлильный цикл (G85)

Расточной цикл `W5DBore5(485)` (G85) выполняет подвод инструмента к центру отверстия, растачивание отверстия с остановом на минимальном уровне и отвод на рабочей подаче к уровню Return.



Стандартный расточной цикл G85 состоит из:

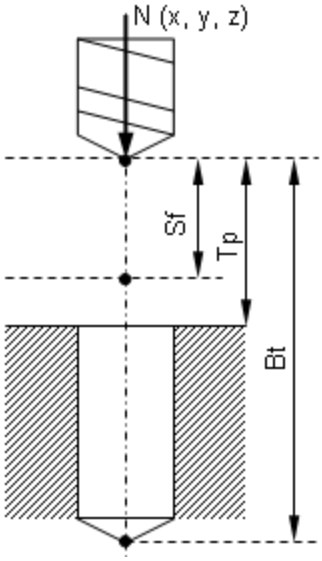
- Ускоренное перемещение к центру отверстия на уровень Z return.
- Ускоренное перемещение на уровень Z safe.
- Перемещение на рабочей подаче на уровень Z min.
- Останов шпинделя.
- Возврат на рабочей подаче на уровень Z return.
- Восстановление направления и частоты вращения шпинделя.

Параметры

Массив CLD	Имя CLD	Описание
CLD[1]	CLD.SubCmd	Тип команды: ON(71) – включение стандартного цикла, CALL(52) – вызов стандартного цикла, OFF(72) – выключение стандартного цикла.
CLD[2]	CLD.SubType	Идентификатор типа стандартного цикла: W5DBore5(485).

Массив CLD	Имя CLD	Описание
CLD[3]	CLD.CLParams(1)	Nx, координата X вектора нормали инструмента
CLD[4]	CLD.CLParams(2)	Ny, координата Y вектора нормали инструмента
CLD[5]	CLD.CLParams(3)	Nz, координата Z вектора нормали инструмента
CLD[6]	CLD.CLParams(4)	Sf, нормальное расстояние от текущего положения инструмента до уровня безопасной плоскости
CLD[7]	CLD.CLParams(5)	Tp, нормальное расстояние от текущего положения инструмента до уровня верха отверстия
CLD[8]	CLD.CLParams(6)	Bt, нормальное расстояние от текущего положения инструмента до уровня дна отверстия
CLD[9]	CLD.CLParams(7)	Единицы рабочей подачи: 0 – мм/об, 1 – мм/мин
CLD[10]	CLD.CLParams(8)	Значение рабочей подачи
CLD[11]	CLD.CLParams(9)	Единицы подачи подвода: 0 – мм/об, 1 – мм/мин
CLD[12]	CLD.CLParams(10)	Значение подачи подвода
CLD[13]	CLD.CLParams(11)	Единицы подачи возврата: 0 – мм/об, 1 – мм/мин
CLD[14]	CLD.CLParams(12)	Значение подачи возврата
CLD[15]	CLD.CLParams(13)	Задержка на уровне дна в секундах

Массив CLD	Имя CLD	Описание
CLD[50]	CLD.CLParams(48)	Какой шпиндель используется для обработки: 1 - приводной инструмент, 2 - шпиндель заготовки (токарный).



Доступ из .NET

Помимо базового `OnExtCycle`, этот цикл доставляется через обработчик семейства циклов отверстий `OnHoleExtCycle`:

```
public override void OnHoleExtCycle(ICLDExtCycleCommand cmd, CLDArray cld)
```

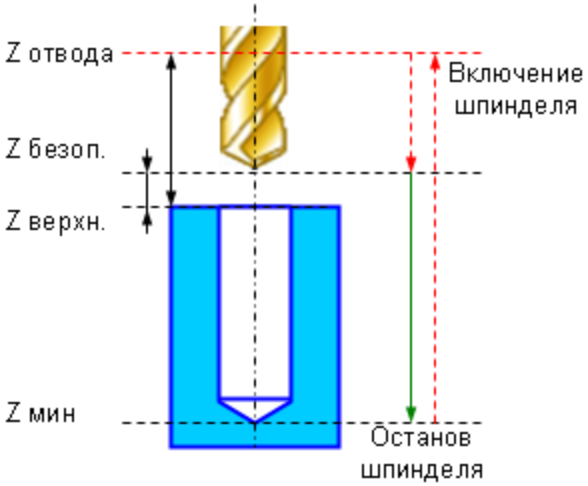
Общая модель .NET описана в разделе Расширенный цикл (EXTCYCLE).

Сопутствующие ссылки

- Расширенный цикл (EXTCYCLE)
- Модель доступа к CLData

EXTCYCLE W5DBore6 - Сверлильный цикл (G86)

Расточной цикл `W5DBore6(486)` (G86) выполняет подвод инструмента к центру отверстия, растачивание отверстия с остановом на минимальном уровне и ускоренный отвод к уровню Return.



Стандартный расточной цикл G86 состоит из:

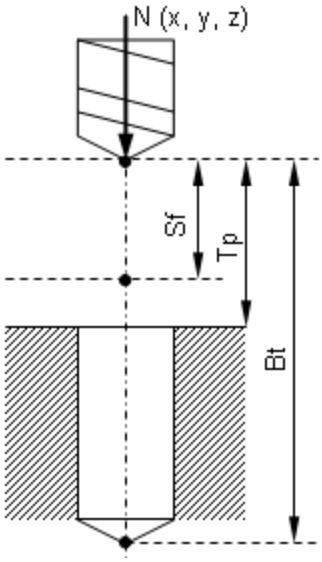
- Ускоренное перемещение к центру отверстия на уровень Z return.
- Ускоренное перемещение на уровень Z safe.
- Перемещение на рабочей подаче на уровень Z min.
- Останов шпинделя. Если включён ориентированный останов шпинделя (параметр CLD[17]), выполняется позиционирование шпинделя в заданное угловое положение, после чего инструмент смещается на заданные малые величины по координатам X, Y и Z.
- Ускоренный возврат на уровень Z return.
- Восстановление направления и частоты вращения шпинделя.

Параметры

Массив CLD	Имя CLD	Описание
CLD[1]	CLD.SubCmd	Тип команды: ON(71) – включение стандартного цикла, CALL(52) – вызов стандартного цикла, OFF(72) – выключение стандартного цикла.
CLD[2]	CLD.SubType	Идентификатор типа стандартного цикла: W5DBore6(486) (G86).

Массив CLD	Имя CLD	Описание
CLD[3]	CLD.CLParams(1)	Nx, координата X вектора нормали инструмента
CLD[4]	CLD.CLParams(2)	Ny, координата Y вектора нормали инструмента
CLD[5]	CLD.CLParams(3)	Nz, координата Z вектора нормали инструмента
CLD[6]	CLD.CLParams(4)	Sf, нормальное расстояние от текущего положения инструмента до уровня безопасной плоскости
CLD[7]	CLD.CLParams(5)	Tp, нормальное расстояние от текущего положения инструмента до уровня верха отверстия
CLD[8]	CLD.CLParams(6)	Bt, нормальное расстояние от текущего положения инструмента до уровня дна отверстия
CLD[9]	CLD.CLParams(7)	Единицы рабочей подачи: 0 – мм/об, 1 – мм/мин
CLD[10]	CLD.CLParams(8)	Значение рабочей подачи
CLD[11]	CLD.CLParams(9)	Единицы подачи подвода: 0 – мм/об, 1 – мм/мин
CLD[12]	CLD.CLParams(10)	Значение подачи подвода
CLD[13]	CLD.CLParams(11)	Единицы подачи возврата: 0 – мм/об, 1 – мм/мин
CLD[14]	CLD.CLParams(12)	Значение подачи возврата
CLD[15]	CLD.CLParams(13)	Задержка на уровне дна в секундах

Массив CLD	Имя CLD	Описание
CLD[16]	CLD.CLParams(14)	Задержка на верхнем уровне в секундах
CLD[17]	CLD.CLParams(15)	Состояние ориентированного останова шпинделя на дне отверстия: 0 - выключено, 1 - включено.
CLD[18]	CLD.CLParams(16)	Угловое положение шпинделя при останове на дне отверстия (градусы)
CLD[19]	CLD.CLParams(17)	Величина смещения инструмента по координате X после ориентированного останова шпинделя на дне отверстия.
CLD[20]	CLD.CLParams(18)	Величина смещения инструмента по координате Y после ориентированного останова шпинделя на дне отверстия.
CLD[21]	CLD.CLParams(19)	Величина смещения инструмента по координате Z после ориентированного останова шпинделя на дне отверстия.
CLD[50]	CLD.CLParams(48)	Какой шпиндель используется для обработки: 1 - приводной инструмент, 2 - шпиндель заготовки (токарный).



Доступ из .NET

Помимо базового `OnExtCycle`, этот цикл доставляется через обработчик семейства циклов отверстий `OnHoleExtCycle`:

```
public override void OnHoleExtCycle(ICLDExtCycleCommand cmd, CLDArray cld)
```

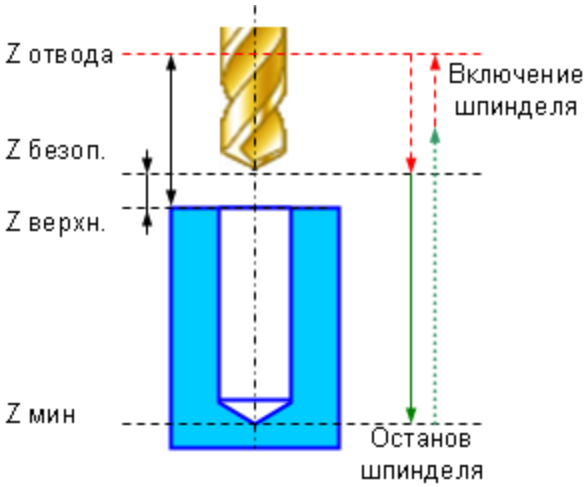
Общая модель .NET описана в разделе Расширенный цикл (EXTCYCLE).

Сопутствующие ссылки

- Расширенный цикл (EXTCYCLE)
- Модель доступа к CLData

EXTCYCLE W5DBore7 - Сверлильный цикл (G87)

Расточной цикл `W5DBore7(487)` (G87) выполняет подвод инструмента к центру отверстия, растачивание отверстия с остановом на минимальном уровне и ручной отвод к уровню Return.



Стандартный расточной цикл G87 состоит из:

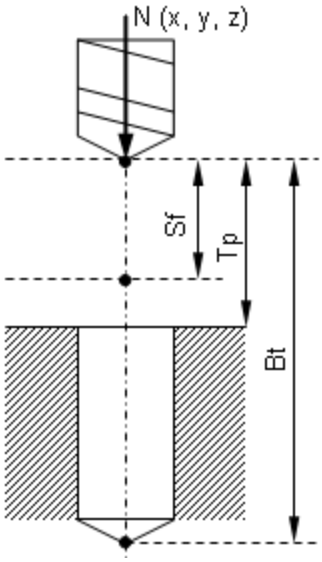
- Ускоренное перемещение к центру отверстия на уровень Z return.
- Ускоренное перемещение на уровень Z safe.
- Перемещение на рабочей подаче на уровень Z min.
- Останов шпинделя. Если включён ориентированный останов шпинделя (параметр CLD[17]), выполняется позиционирование шпинделя в заданное угловое положение, после чего инструмент смещается на заданные малые величины по координатам X, Y и Z.
- Ручной отвод на уровень Z return.
- Восстановление направления и частоты вращения шпинделя.

Параметры

Массив CLD	Имя CLD	Описание
CLD[1]	CLD.SubCmd	Тип команды: ON(71) – включение стандартного цикла, CALL(52) – вызов стандартного цикла, OFF(72) – выключение стандартного цикла.
CLD[2]	CLD.SubType	Идентификатор типа стандартного цикла: W5DBore7(487) (G87).

Массив CLD	Имя CLD	Описание
CLD[3]	CLD.CLParams(1)	Nx, координата X вектора нормали инструмента
CLD[4]	CLD.CLParams(2)	Ny, координата Y вектора нормали инструмента
CLD[5]	CLD.CLParams(3)	Nz, координата Z вектора нормали инструмента
CLD[6]	CLD.CLParams(4)	Sf, нормальное расстояние от текущего положения инструмента до уровня безопасной плоскости
CLD[7]	CLD.CLParams(5)	Tp, нормальное расстояние от текущего положения инструмента до уровня верха отверстия
CLD[8]	CLD.CLParams(6)	Bt, нормальное расстояние от текущего положения инструмента до уровня дна отверстия
CLD[9]	CLD.CLParams(7)	Единицы рабочей подачи: 0 – мм/об, 1 – мм/мин
CLD[10]	CLD.CLParams(8)	Значение рабочей подачи
CLD[11]	CLD.CLParams(9)	Единицы подачи подвода: 0 – мм/об, 1 – мм/мин
CLD[12]	CLD.CLParams(10)	Значение подачи подвода
CLD[13]	CLD.CLParams(11)	Единицы подачи возврата: 0 – мм/об, 1 – мм/мин
CLD[14]	CLD.CLParams(12)	Значение подачи возврата
CLD[17]	CLD.CLParams(15)	Состояние ориентированного останова

Массив CLD	Имя CLD	Описание
		шпинделя на дне отверстия: 0 - выключено, 1 - включено.
CLD[18]	CLD.CLParams(16)	Угловое положение шпинделя при останове на дне отверстия (градусы)
CLD[19]	CLD.CLParams(17)	Величина смещения инструмента по координате X после ориентированного останова шпинделя на дне отверстия.
CLD[20]	CLD.CLParams(18)	Величина смещения инструмента по координате Y после ориентированного останова шпинделя на дне отверстия.
CLD[21]	CLD.CLParams(19)	Величина смещения инструмента по координате Z после ориентированного останова шпинделя на дне отверстия.
CLD[50]	CLD.CLParams(48)	Какой шпиндель используется для обработки: 1 - приводной инструмент, 2 - шпиндель заготовки (токарный).



Доступ из .NET

Помимо базового `OnExtCycle`, этот цикл доставляется через обработчик семейства циклов отверстий `OnHoleExtCycle`:

```
public override void OnHoleExtCycle(ICLDExtCycleCommand cmd, CLDArray cld)
```

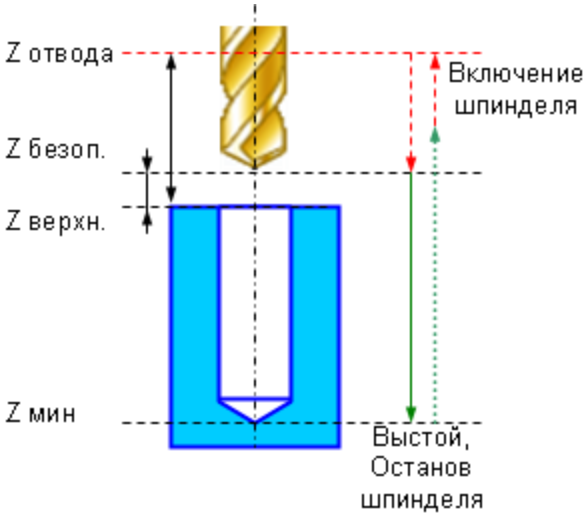
Общая модель .NET описана в разделе Расширенный цикл (EXTCYCLE).

Сопутствующие ссылки

- Расширенный цикл (EXTCYCLE)
- Модель доступа к CLData

EXTCYCLE W5DBore8 - Сверлильный цикл (G88)

Сверлильный цикл типа `W5DBore8(488)` (G88) сверлит отверстия с ускоренным подводом к безопасному уровню, задержкой на уровне дна отверстия, остановом шпинделя и ручным отводом к уровню безопасной плоскости.



Сверлильный цикл G88 состоит из следующих шагов:

- Ускоренное перемещение инструмента к центру отверстия на уровень Z return.
- Ускоренное опускание на уровень Z safe.
- Перемещение на рабочей подаче на уровень Z min.
- Задержка на уровне Z min.
- Останов шпинделя.
- Ручной отвод инструмента на уровень Z return.
- Восстановление направления и частоты вращения шпинделя.

Параметры

Массив CLD	Имя CLD	Описание
CLD[1]	CLD.SubCmd	Тип команды: ON(71) – включение стандартного цикла, CALL(52) – вызов стандартного цикла, OFF(72) – выключение стандартного цикла.
CLD[2]	CLD.SubType	Идентификатор типа стандартного цикла: W5DBore8(488) (G88).

Массив CLD	Имя CLD	Описание
CLD[3]	CLD.CLParams(1)	Nx, координата X вектора нормали инструмента
CLD[4]	CLD.CLParams(2)	Ny, координата Y вектора нормали инструмента
CLD[5]	CLD.CLParams(3)	Nz, координата Z вектора нормали инструмента
CLD[6]	CLD.CLParams(4)	Sf, нормальное расстояние от текущего положения инструмента до уровня безопасной плоскости
CLD[7]	CLD.CLParams(5)	Tp, нормальное расстояние от текущего положения инструмента до уровня верха отверстия
CLD[8]	CLD.CLParams(6)	Bt, нормальное расстояние от текущего положения инструмента до уровня дна отверстия
CLD[9]	CLD.CLParams(7)	Единицы рабочей подачи: 0 – мм/об, 1 – мм/мин
CLD[10]	CLD.CLParams(8)	Значение рабочей подачи
CLD[11]	CLD.CLParams(9)	Единицы подачи подвода: 0 – мм/об, 1 – мм/мин
CLD[12]	CLD.CLParams(10)	Значение подачи подвода
CLD[13]	CLD.CLParams(11)	Единицы подачи возврата: 0 – мм/об, 1 – мм/мин
CLD[14]	CLD.CLParams(12)	Значение подачи возврата
CLD[15]	CLD.CLParams(13)	Задержка на уровне дна в секундах

Массив CLD	Имя CLD	Описание
CLD[50]	CLD.CLParams(48)	Какой шпиндель используется для обработки: 1 - приводной инструмент, 2 - шпиндель заготовки (токарный).

Доступ из .NET

Помимо базового `OnExtCycle`, этот цикл доставляется через обработчик семейства циклов отверстий `OnHoleExtCycle`:

```
public override void OnHoleExtCycle(ICLDExtCycleCommand cmd, CLDArray cld)
```

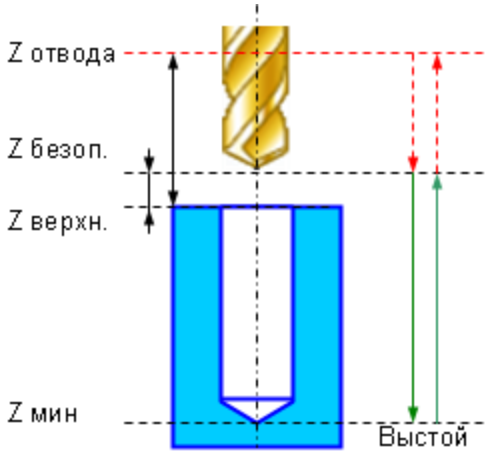
Общая модель .NET описана в разделе Расширенный цикл (EXTCYCLE).

Сопутствующие ссылки

- Расширенный цикл (EXTCYCLE)
- Модель доступа к CLData

EXTCYCLE W5DBore9 - Сверлильный цикл (G89)

Расточной цикл типа `W5DBore9(489)` (G89) растачивает отверстия с ускоренным подводом к безопасному уровню, задержкой на уровне дна отверстия, остановом шпинделя и ручным отводом к уровню безопасной плоскости.



Расточной цикл G89 состоит из следующих шагов:

- Ускоренное перемещение инструмента к центру отверстия на уровень Z return.
- Ускоренное опускание на уровень Z safe.
- Перемещение на рабочей подаче на уровень Z min.
- Задержка на уровне Z min.
- Возврат на рабочей подаче на уровень Z safe.
- Восстановление направления и частоты вращения шпинделя.

Параметры

Массив CLD	Имя CLD	Описание
CLD[1]	CLD.SubCmd	Тип команды: ON(71) – включение стандартного цикла, CALL(52) – вызов стандартного цикла, OFF(72) – выключение стандартного цикла.
CLD[2]	CLD.SubType	Идентификатор типа стандартного цикла: W5DBore9(489) (G89).

Массив CLD	Имя CLD	Описание
CLD[3]	CLD.CLParams(1)	Nx, координата X вектора нормали инструмента
CLD[4]	CLD.CLParams(2)	Ny, координата Y вектора нормали инструмента
CLD[5]	CLD.CLParams(3)	Nz, координата Z вектора нормали инструмента
CLD[6]	CLD.CLParams(4)	Sf, нормальное расстояние от текущего положения инструмента до уровня безопасной плоскости
CLD[7]	CLD.CLParams(5)	Tp, нормальное расстояние от текущего положения инструмента до уровня верха отверстия
CLD[8]	CLD.CLParams(6)	Bt, нормальное расстояние от текущего положения инструмента до уровня дна отверстия
CLD[9]	CLD.CLParams(7)	Единицы рабочей подачи: 0 – мм/об, 1 – мм/мин
CLD[10]	CLD.CLParams(8)	Значение рабочей подачи
CLD[11]	CLD.CLParams(9)	Единицы подачи подвода: 0 – мм/об, 1 – мм/мин
CLD[12]	CLD.CLParams(10)	Значение подачи подвода
CLD[13]	CLD.CLParams(11)	Единицы подачи возврата: 0 – мм/об, 1 – мм/мин
CLD[14]	CLD.CLParams(12)	Значение подачи возврата
CLD[15]	CLD.CLParams(13)	Задержка на уровне дна в секундах

Массив CLD	Имя CLD	Описание
CLD[50]	CLD.CLParams(48)	Какой шпиндель используется для обработки: 1 - приводной инструмент, 2 - шпиндель заготовки (токарный).

Доступ из .NET

Помимо базового `OnExtCycle`, этот цикл доставляется через обработчик семейства циклов отверстий `OnHoleExtCycle`:

```
public override void OnHoleExtCycle(ICLDExtCycleCommand cmd, CLDArray cld)
```

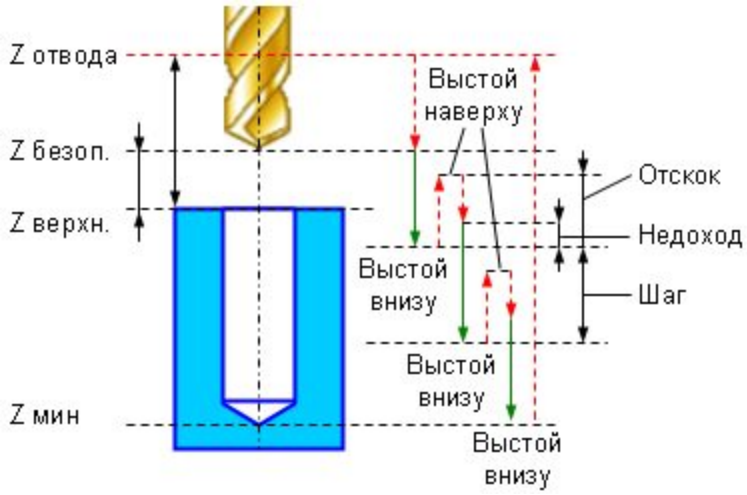
Общая модель .NET описана в разделе Расширенный цикл (EXTCYCLE).

Сопутствующие ссылки

- Расширенный цикл (EXTCYCLE)
- Модель доступа к CLData

EXTCYCLE W5DChipBreaking - Сверлильный цикл с ломкой стружки (G73)

Сверлильный цикл с ломкой стружки `W5DChipBreaking(473)` (G73) выполняет подвод инструмента к центру отверстия на уровень Z return. Циклическое сверление выполняется с отводом инструмента для ломки стружки.



Цикл состоит из:

- Ускоренный подвод к центру отверстия на уровень Z return.
- Ускоренное перемещение на уровень Z safe.
- Перемещение на рабочей подаче на глубину шага S.
- Задержка на время задержки на дне.
- Ускоренный отвод инструмента на `расстояние вывода (Ld)`.
- Задержка на время задержки на верху.
- Ускоренное перемещение на уровень предыдущей глубины с `замедлением (Dcl)`.
- Рабочая подача до `замедления (Dcl)` с `шагом (S)`.
- Задержка на время задержки на дне.
- Повторение предыдущих пяти итераций до достижения полной глубины отверстия.
- Ускоренный возврат инструмента на уровень Z retract.

Параметры

Массив CLD	Имя CLD	Описание
CLD[1]	CLD.SubCmd	Тип команды: ON(71) – включение стандартного цикла, CALL(52) – вызов стандартного цикла, OFF(72) – выключение стандартного цикла.
CLD[2]	CLD.SubType	Идентификатор типа стандартного цикла: W5DChipBreaking(473).
CLD[3]	CLD.CLParams(1)	Nx, координата X вектора нормали инструмента
CLD[4]	CLD.CLParams(2)	Ny, координата Y вектора нормали инструмента
CLD[5]	CLD.CLParams(3)	Nz, координата Z вектора нормали инструмента
CLD[6]	CLD.CLParams(4)	Sf, нормальное расстояние от текущего положения инструмента до уровня безопасной плоскости
CLD[7]	CLD.CLParams(5)	Tp, нормальное расстояние от текущего положения инструмента до уровня верха отверстия
CLD[8]	CLD.CLParams(6)	Bt, нормальное расстояние от текущего положения инструмента до уровня дна отверстия
CLD[9]	CLD.CLParams(7)	Единицы рабочей подачи: 0 – мм/об, 1 – мм/мин
CLD[10]	CLD.CLParams(8)	Значение рабочей подачи

Массив CLD	Имя CLD	Описание
CLD[11]	CLD.CLParams(9)	Единицы подачи подвода: 0 – мм/об, 1 – мм/мин
CLD[12]	CLD.CLParams(10)	Значение подачи подвода
CLD[13]	CLD.CLParams(11)	Единицы подачи возврата: 0 – мм/об, 1 – мм/мин
CLD[14]	CLD.CLParams(12)	Значение подачи возврата
CLD[15]	CLD.CLParams(13)	Задержка на уровне дна в секундах
CLD[16]	CLD.CLParams(14)	Задержка на верхнем уровне в секундах
CLD[17]	CLD.CLParams(15)	St, величина шага ломки стружки
CLD[18]	CLD.CLParams(16)	Dg, уменьшение шага ломки стружки на каждой итерации
CLD[19]	CLD.CLParams(17)	Dc, величина замедления перед сверлением для каждого шага
CLD[20]	CLD.CLParams(18)	Ld, величина вывода на каждом шаге для прерывания процесса резания
CLD[50]	CLD.CLParams(48)	Какой шпиндель используется для обработки: 1 - приводной инструмент, 2 - шпиндель заготовки (токарный).

Доступ из .NET

Помимо базового `OnExtCycle`, этот цикл доставляется через обработчик семейства циклов отверстий `OnHoleExtCycle`:

```
public override void OnHoleExtCycle(ICLDExtCycleCommand cmd, CLDArray cld)
```

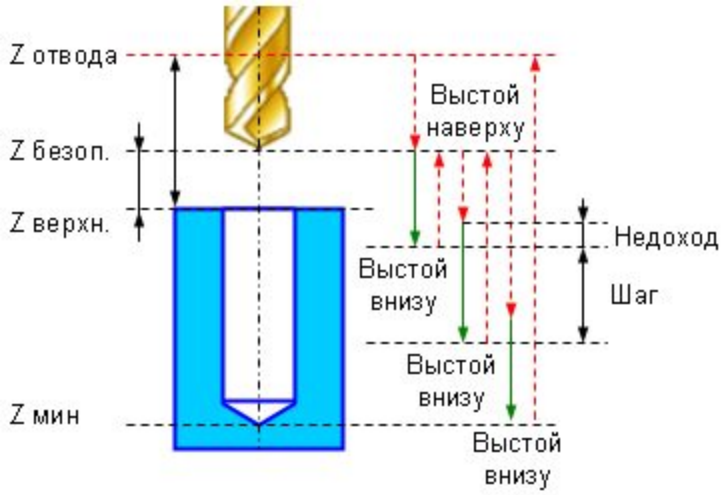
Общая модель .NET описана в разделе Расширенный цикл (EXTCYCLE).

Сопутствующие ссылки

- Расширенный цикл (EXTCYCLE)
- Модель доступа к CLData

EXTCYCLE W5DChipRemoving - Сверлильный цикл с удалением стружки (G83)

Сверлильный цикл с удалением стружки `W5DChipRemoving(483)` (G83) выполняет перемещение инструмента к центру отверстия на уровень Z return и последующее циклическое сверление с отводом инструмента на уровень Z safe.



Цикл состоит из:

- Ускоренный подвод к центру отверстия на уровень Z return.
- Ускоренное перемещение на уровень Z safe.
- Перемещение на рабочей подаче на глубину шага S.
- Задержка на время задержки на дне.
- Ускоренный возврат на уровень Z safe.
- Задержка на время задержки на верху.
- Ускоренное перемещение на уровень предыдущей глубины с `замедлением (Dcl)`.
- Рабочая подача до `замедления (Dcl)` с `шагом (S)`.
- Задержка на время задержки на дне.
- Повторение предыдущих пяти итераций до достижения полной глубины отверстия.
- Ускоренный возврат инструмента на уровень Z retract.

Параметры

Массив CLD	Имя CLD	Описание
CLD[1]	CLD.SubCmd	Тип команды: ON(71) – включение стандартного цикла, CALL(52) – вызов стандартного цикла, OFF(72)

Массив CLD	Имя CLD	Описание
		– выключение стандартного цикла.
CLD[2]	CLD.SubType	Идентификатор типа стандартного цикла: W5DChipRemoving(483).
CLD[3]	CLD.CLParams(1)	Nx, координата X вектора нормали инструмента
CLD[4]	CLD.CLParams(2)	Ny, координата Y вектора нормали инструмента
CLD[5]	CLD.CLParams(3)	Nz, координата Z вектора нормали инструмента
CLD[6]	CLD.CLParams(4)	Sf, нормальное расстояние от текущего положения инструмента до уровня безопасной плоскости
CLD[7]	CLD.CLParams(5)	Tp, нормальное расстояние от текущего положения инструмента до уровня верха отверстия
CLD[8]	CLD.CLParams(6)	Bt, нормальное расстояние от текущего положения инструмента до уровня дна отверстия
CLD[9]	CLD.CLParams(7)	Единицы рабочей подачи: 0 – мм/об, 1 – мм/мин
CLD[10]	CLD.CLParams(8)	Значение рабочей подачи
CLD[11]	CLD.CLParams(9)	Единицы подачи подвода: 0 – мм/об, 1 – мм/мин
CLD[12]	CLD.CLParams(10)	Значение подачи подвода

Массив CLD	Имя CLD	Описание
CLD[13]	CLD.CLParams(11)	Единицы подачи возврата: 0 – мм/об, 1 – мм/мин
CLD[14]	CLD.CLParams(12)	Значение подачи возврата
CLD[15]	CLD.CLParams(13)	Задержка на уровне дна в секундах
CLD[16]	CLD.CLParams(14)	Задержка на верхнем уровне в секундах
CLD[17]	CLD.CLParams(15)	St, величина шага удаления стружки
CLD[18]	CLD.CLParams(16)	Dg, уменьшение шага удаления стружки на каждой итерации
CLD[19]	CLD.CLParams(17)	Dc, величина замедления перед сверлением для каждого шага
CLD[50]	CLD.CLParams(48)	Какой шпиндель используется для обработки: 1 - приводной инструмент, 2 - шпиндель заготовки (токарный).

Доступ из .NET

Помимо базового `OnExtCycle`, этот цикл доставляется через обработчик семейства циклов отверстий `OnHoleExtCycle`:

```
public override void OnHoleExtCycle(ICLDExtCycleCommand cmd, CLDArray cld)
```

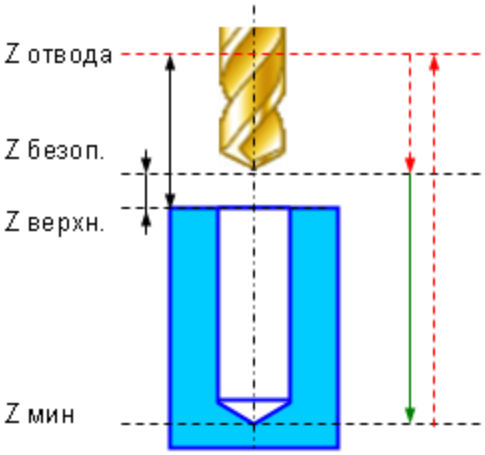
Общая модель .NET описана в разделе Расширенный цикл (EXTCYCLE).

Сопутствующие ссылки

- Расширенный цикл (EXTCYCLE)
- Модель доступа к CLData

EXTCYCLE W5DDrill - Сверлильный цикл (G81)

Сверлильный цикл типа `W5DDrill(481)` (G81) сверлит отверстия с ускоренным подводом к безопасному уровню и ускоренным отводом к уровню безопасной плоскости.



Сверлильный цикл G81 состоит из следующих шагов:

- Ускоренное перемещение инструмента к центру отверстия на уровень Z return.
- Ускоренное опускание на уровень Z safe.
- Перемещение на рабочей подаче на уровень Z min.
- Ускоренный возврат инструмента на уровень Z return.

Параметры

Массив CLD	Имя CLD	Описание
CLD[1]	CLD.SubCmd	Тип команды: ON(71) – включение стандартного цикла, CALL(52) – вызов стандартного цикла, OFF(72) – выключение стандартного цикла.
CLD[2]	CLD.SubType	Идентификатор типа стандартного цикла: W5DDrill(481).
CLD[3]	CLD.CLParams(1)	Nx, координата X вектора нормали инструмента
CLD[4]	CLD.CLParams(2)	Ny, координата Y вектора нормали инструмента

Массив CLD	Имя CLD	Описание
CLD[5]	CLD.CLParams(3)	Nz, координата Z вектора нормали инструмента
CLD[6]	CLD.CLParams(4)	Sf, нормальное расстояние от текущего положения инструмента до уровня безопасной плоскости
CLD[7]	CLD.CLParams(5)	Tp, нормальное расстояние от текущего положения инструмента до уровня верха отверстия
CLD[8]	CLD.CLParams(6)	Bt, нормальное расстояние от текущего положения инструмента до уровня дна отверстия
CLD[9]	CLD.CLParams(7)	Единицы рабочей подачи: 0 – мм/об, 1 – мм/мин
CLD[10]	CLD.CLParams(8)	Значение рабочей подачи
CLD[11]	CLD.CLParams(9)	Единицы подачи подвода: 0 – мм/об, 1 – мм/мин
CLD[12]	CLD.CLParams(10)	Значение подачи подвода
CLD[13]	CLD.CLParams(11)	Единицы подачи возврата: 0 – мм/об, 1 – мм/мин
CLD[14]	CLD.CLParams(12)	Значение подачи возврата
CLD[50]	CLD.CLParams(48)	Какой шпиндель используется для обработки: 1 - приводной инструмент, 2 - шпиндель заготовки (токарный).

Доступ из .NET

Помимо базового `OnExtCycle`, этот цикл доставляется через обработчик семейства циклов отверстий `OnHoleExtCycle`:

```
public override void OnHoleExtCycle(ICLDExtCycleCommand cmd, CLDArray cld)
```

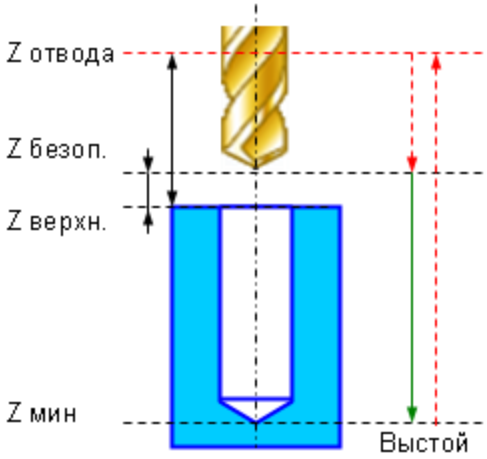
Общая модель .NET описана в разделе Расширенный цикл (EXTCYCLE).

Сопутствующие ссылки

- Расширенный цикл (EXTCYCLE)
- Модель доступа к CLData

EXTCYCLE W5DFace - Сверлильный цикл (G82)

Сверлильный цикл типа `W5DFace(482)` (G82) сверлит отверстия с ускоренным подводом к безопасному уровню, задержкой на уровне дна отверстия и ускоренным отводом к уровню безопасной плоскости.



Сверлильный цикл G82 состоит из следующих шагов:

- Ускоренное перемещение инструмента к центру отверстия на уровень Z return.
- Ускоренное опускание на уровень Z safe.
- Перемещение на рабочей подаче на уровень Z min.
- Задержка на уровне Z min.
- Ускоренный возврат инструмента на уровень Z return.

Параметры

Массив CLD	Имя CLD	Описание
CLD[1]	CLD.SubCmd	Тип команды: ON(71) – включение стандартного цикла, CALL(52) – вызов стандартного цикла, OFF(72) – выключение стандартного цикла.
CLD[2]	CLD.SubType	Идентификатор типа стандартного цикла: W5DFace(482).
CLD[3]	CLD.CLParams(1)	Nx, координата X вектора нормали инструмента

Массив CLD	Имя CLD	Описание
CLD[4]	CLD.CLParams(2)	Ny, координата Y вектора нормали инструмента
CLD[5]	CLD.CLParams(3)	Nz, координата Z вектора нормали инструмента
CLD[6]	CLD.CLParams(4)	Sf, нормальное расстояние от текущего положения инструмента до уровня безопасной плоскости
CLD[7]	CLD.CLParams(5)	Tp, нормальное расстояние от текущего положения инструмента до уровня верха отверстия
CLD[8]	CLD.CLParams(6)	Bt, нормальное расстояние от текущего положения инструмента до уровня дна отверстия
CLD[9]	CLD.CLParams(7)	Единицы рабочей подачи: 0 – мм/об, 1 – мм/мин
CLD[10]	CLD.CLParams(8)	Значение рабочей подачи
CLD[11]	CLD.CLParams(9)	Единицы подачи подвода: 0 – мм/об, 1 – мм/мин
CLD[12]	CLD.CLParams(10)	Значение подачи подвода
CLD[13]	CLD.CLParams(11)	Единицы подачи возврата: 0 – мм/об, 1 – мм/мин
CLD[14]	CLD.CLParams(12)	Значение подачи возврата
CLD[15]	CLD.CLParams(13)	Задержка на уровне дна в секундах
CLD[50]	CLD.CLParams(48)	Какой шпиндель используется для обработки: 1 - приводной инструмент, 2

Массив CLD	Имя CLD	Описание
		- шпиндель заготовки (токарный).

Доступ из .NET

Помимо базового `OnExtCycle`, этот цикл доставляется через обработчик семейства циклов отверстий `OnHoleExtCycle`:

```
public override void OnHoleExtCycle(ICLDExtCycleCommand cmd, CLDArray cld)
```

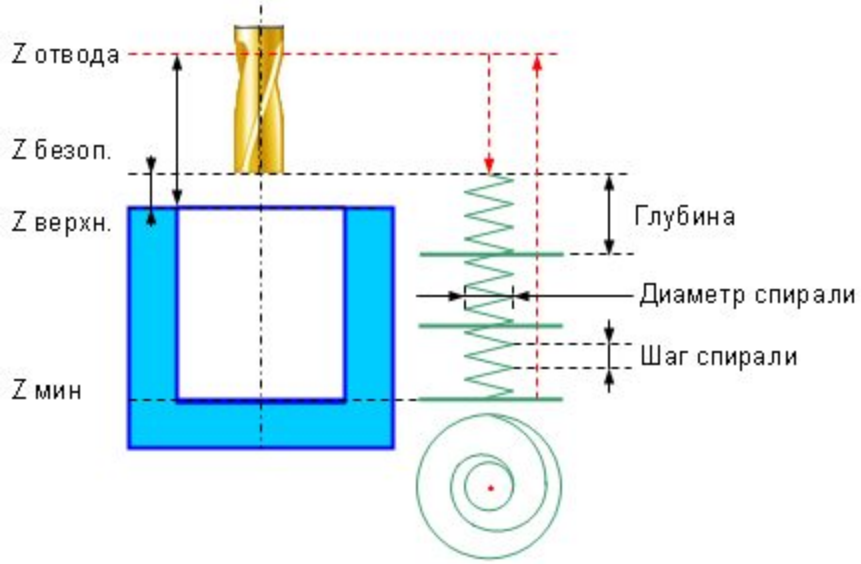
Общая модель .NET описана в разделе Расширенный цикл (EXTCYCLE).

Сопутствующие ссылки

- Расширенный цикл (EXTCYCLE)
- Модель доступа к CLData

EXTCYCLE W5DHolePocketing - Цикл выборки отверстия

Цикл `W5DHolePocketing(491)` применяется для обработки отверстий, диаметр которых больше диаметра инструмента. Выборка выполняется послойно. Инструмент врезается по спирали на каждый слой, затем расширяет отверстие до нужного диаметра, перемещаясь по спирали Архимеда, с чистовым проходом по окружности.

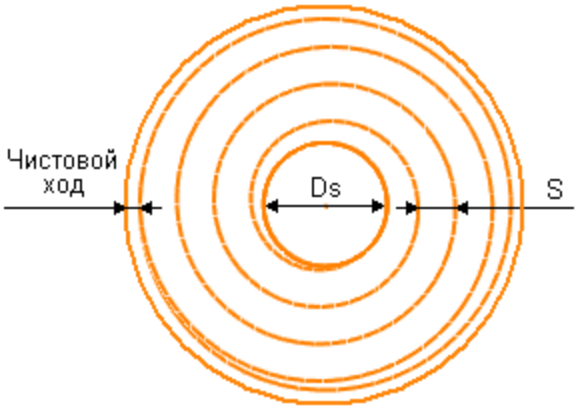


Цикл выборки отверстия включает следующее:

- Ускоренный подвод к центру отверстия на уровень Z retract.
- Ускоренное перемещение на уровень Z safe.
- Врезание по спирали на рабочей подаче на Z глубину обработки.
- Перемещение по спирали Архимеда с шагом S на этом уровне до тех пор, пока ось инструмента не достигнет окружности с диаметром, равным диаметру отверстия, уменьшенному на диаметр инструмента.
- Чистовой проход по указанной выше окружности без изменения уровня.
- Повторение предыдущих трёх шагов до обработки нужной глубины отверстия с перемещением к следующей точке врезания без изменения уровня.
- Возврат к центру отверстия.
- Ускоренное перемещение на уровень Z retract.

Направление закрутки спирали может быть одним из следующих:

- Right (правая). Спираль закручена вправо. Инструмент вращается по часовой стрелке при взгляде сверху.
- Left (левая). Спираль закручена влево. Инструмент вращается против часовой стрелки при взгляде сверху.
- Counter (встречное). Направление закрутки спирали определяется направлением вращения шпинделя и соответствует встречному фрезерованию. При встречной фрезерной выборке направления вращения инструмента и закрутки спирали противоположны.
- Follow (попутное). Направление закрутки спирали определяется направлением вращения шпинделя и соответствует попутному фрезерованию. При попутной фрезерной выборке направления вращения инструмента и закрутки спирали совпадают.



Если величина чистового прохода не равна нулю, то перед финальным проходом по окружности выполняется дополнительный проход по окружности с заданным припуском. Это позволяет обеспечить равномерный припуск финального прохода.

Параметры

Массив CLD	Имя CLD	Описание
CLD[1]	CLD.SubCmd	Тип команды: ON(71) – включение цикла, CALL(52) – вызов цикла, OFF(72) – выключение цикла.
CLD[2]	CLD.SubType	Идентификатор типа цикла: W5DHolePocketing(491).
CLD[3]	CLD.CLParams(1)	Nx, координата X вектора нормали инструмента
CLD[4]	CLD.CLParams(2)	Ny, координата Y вектора нормали инструмента

Массив CLD	Имя CLD	Описание
CLD[5]	CLD.CLParams(3)	Nz, координата Z вектора нормали инструмента
CLD[6]	CLD.CLParams(4)	Sf, нормальное расстояние от текущего положения инструмента до уровня безопасной плоскости
CLD[7]	CLD.CLParams(5)	Tp, нормальное расстояние от текущего положения инструмента до уровня верха отверстия
CLD[8]	CLD.CLParams(6)	Bt, нормальное расстояние от текущего положения инструмента до уровня дна отверстия
CLD[9]	CLD.CLParams(7)	Единицы рабочей подачи: 0 – мм/об, 1 – мм/мин
CLD[10]	CLD.CLParams(8)	Значение рабочей подачи
CLD[11]	CLD.CLParams(9)	Единицы подачи подвода: 0 – мм/об, 1 – мм/мин
CLD[12]	CLD.CLParams(10)	Значение подачи подвода
CLD[13]	CLD.CLParams(11)	Единицы подачи возврата: 0 – мм/об, 1 – мм/мин
CLD[14]	CLD.CLParams(12)	Значение подачи возврата
CLD[16]	CLD.CLParams(14)	Диаметр отверстия
CLD[17]	CLD.CLParams(15)	Шаг спирали
CLD[18]	CLD.CLParams(16)	Диаметр спирали
CLD[19]	CLD.CLParams(17)	Направление спирали: 0 – встречное, 1 – попутное, 2 – правое, 3 – левое. В случаях

Массив CLD	Имя CLD	Описание
		0 и 1 направление спирали зависит от направления вращения инструмента, в случаях 2 и 3 они независимы.
CLD[20]	CLD.CLParams(18)	Глубина резания вдоль оси инструмента
CLD[21]	CLD.CLParams(19)	Число осевых проходов
CLD[22]	CLD.CLParams(20)	Шаг спирали Архимеда
CLD[23]	CLD.CLParams(21)	Величина чистового прохода

Доступ из .NET

Помимо базового `OnExtCycle`, этот цикл доставляется через обработчик семейства циклов отверстий `OnHoleExtCycle`:

```
public override void OnHoleExtCycle(ICLDExtCycleCommand cmd, CLDArray cld)
```

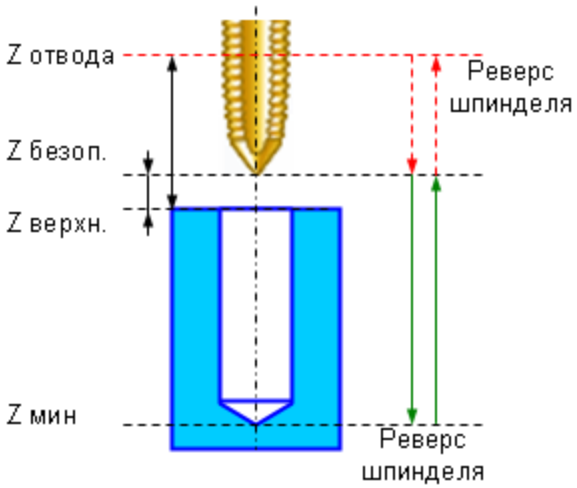
Общая модель .NET описана в разделе Расширенный цикл (EXTCYCLE).

Сопутствующие ссылки

- Расширенный цикл (EXTCYCLE)
- Модель доступа к CLData

EXTCYCLE W5DTap - Цикл нарезания резьбы метчиком (G84)

Цикл нарезания резьбы метчиком `W5DTap(484)` (G84) выполняет ускоренный подвод к уровню Z return, нарезание резьбы с последующим отводом на рабочей подаче с реверсом вращения шпинделя.



Цикл нарезания резьбы метчиком G84 включает:

- Ускоренный подвод к центру отверстия на уровень Z return.
- Ускоренное перемещение на уровень Z safe.
- Перемещение на рабочей подаче на уровень Z min.
- Реверс вращения шпинделя и перемещение на рабочей подаче на уровень Z safe.
- Ускоренный отвод на уровень Z return.
- Восстановление направления и частоты вращения шпинделя.

Если включена соответствующая настройка, то метчик отводится для ломки / удаления стружки во время выполнения цикла. Способ ломки стружки определяется параметром CLD[20].

Системы ЧПУ часто имеют разные стандартные циклы нарезания резьбы метчиком для жёсткого и плавающего типов патрона. Тип патрона задаётся параметром `CLD[19]` (CLD.CLParams(17)).

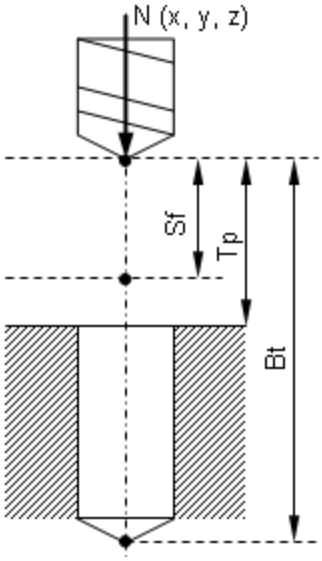
Параметры

Массив CLD	Имя CLD	Описание
CLD[1]	CLD.SubCmd	Тип команды: ON(71) – включение стандартного цикла, CALL(52) – вызов

Массив CLD	Имя CLD	Описание
		стандартного цикла, OFF(72) – выключение стандартного цикла.
CLD[2]	CLD.SubType	Идентификатор типа стандартного цикла: W5DTap(484).
CLD[3]	CLD.CLParams(1)	Nx, координата X вектора нормали инструмента
CLD[4]	CLD.CLParams(2)	Ny, координата Y вектора нормали инструмента
CLD[5]	CLD.CLParams(3)	Nz, координата Z вектора нормали инструмента
CLD[6]	CLD.CLParams(4)	Sf, нормальное расстояние от текущего положения инструмента до уровня безопасной плоскости
CLD[7]	CLD.CLParams(5)	Tp, нормальное расстояние от текущего положения инструмента до уровня верха отверстия
CLD[8]	CLD.CLParams(6)	Bt, нормальное расстояние от текущего положения инструмента до уровня дна отверстия
CLD[9]	CLD.CLParams(7)	Единицы рабочей подачи: 0 – мм/об, 1 – мм/мин
CLD[10]	CLD.CLParams(8)	Значение рабочей подачи
CLD[11]	CLD.CLParams(9)	Единицы подачи подвода: 0 – мм/об, 1 – мм/мин
CLD[12]	CLD.CLParams(10)	Значение подачи подвода

Массив CLD	Имя CLD	Описание
CLD[13]	CLD.CLParams(11)	Единицы подачи возврата: 0 – мм/об, 1 – мм/мин
CLD[14]	CLD.CLParams(12)	Значение подачи возврата
CLD[15]	CLD.CLParams(13)	Задержка на уровне дна в секундах
CLD[16]	CLD.CLParams(14)	Задержка на верхнем уровне в секундах
CLD[17]	CLD.CLParams(15)	Шаг резьбы
CLD[18]	CLD.CLParams(16)	Начальный угол шпинделя в градусах (для многозаходных резьб)
CLD[19]	CLD.CLParams(17)	Тип патрона метчика: 0 – плавающий (компенсирующий), 1 – жёсткий (некомпенсирующий)
CLD[20]	CLD.CLParams(18)	Способ ломки стружки: 0 – без ломки стружки, 1 – удаление стружки, 2 – ломка стружки
CLD[21]	CLD.CLParams(19)	St, величина шага ломки стружки
CLD[22]	CLD.CLParams(20)	Dg, уменьшение шага ломки стружки на каждой итерации
CLD[23]	CLD.CLParams(21)	Dc, величина замедления перед сверлением для каждого шага
CLD[24]	CLD.CLParams(22)	Ld, величина вывода на каждом шаге для

Массив CLD	Имя CLD	Описание
		прерывания процесса резания
CLD[50]	CLD.CLParams(48)	Какой шпиндель используется для обработки: 1 - приводной инструмент, 2 - шпиндель заготовки (токарный).



Доступ из .NET

Помимо базового `OnExtCycle`, этот цикл доставляется через обработчик семейства циклов отверстий `OnHoleExtCycle`:

```
public override void OnHoleExtCycle(ICLDExtCycleCommand cmd, CLDArray cld)
```

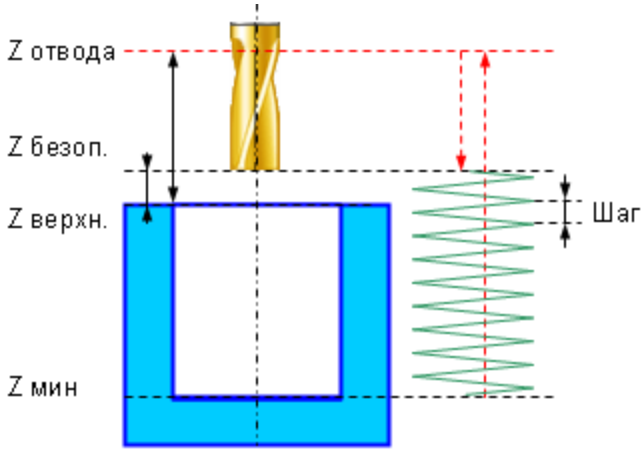
Общая модель .NET описана в разделе Расширенный цикл (EXTCYCLE).

Сопутствующие ссылки

- Расширенный цикл (EXTCYCLE)
- Модель доступа к CLData

EXTCYCLE W5DThreadMill - Цикл фрезерования резьбы

Цикл фрезерования резьбы `W5DThreadMill(490)` применяется для обработки наружной или внутренней резьбы либо для обработки отверстия по спирали. Спиральная обработка используется, когда диаметр отверстия больше диаметра инструмента. Инструмент вращается вокруг оси отверстия и одновременно перемещается вдоль оси. Диаметр спирали выбирается в соответствии с размерами отверстия и инструмента. Обработка может выполняться в несколько проходов для фрезерования отверстий нужного диаметра.



Спиральная обработка включает следующие шаги:

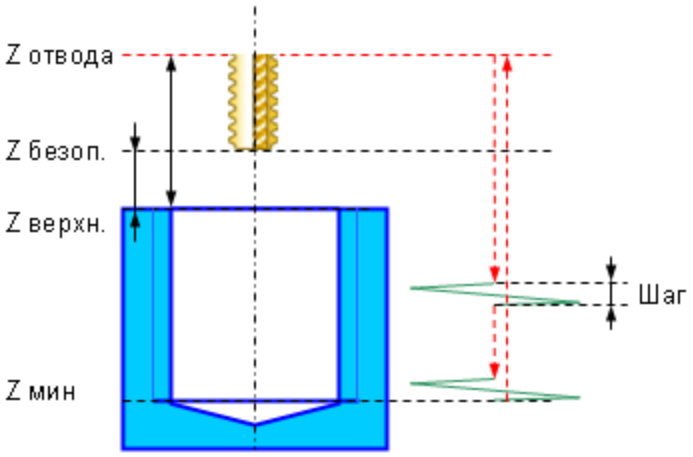
- Ускоренный подвод к центру отверстия на уровень Z retract.
- Ускоренное перемещение на уровень Z safe.
- Перемещение на рабочей подаче к началу спирали.
- Перемещение по спирали на рабочей подаче на уровень Z min.
- Необязательный круговой проход на уровне дна. Диаметр окружности равен диаметру спирали.
- Возврат к центру отверстия.
- Ускоренное перемещение на уровень Z safe.
- Если применяются дополнительные черновые и чистовые проходы, предыдущие пять шагов повторяются до достижения нужного диаметра отверстия.
- Ускоренное перемещение на уровень Z retract.

Фрезерование резьбы обладает следующими преимуществами перед традиционным нарезанием резьбы метчиком:

- глухие и сквозные, левые и правые резьбы обрабатываются одним и тем же инструментом;
- различные резьбы с одинаковым шагом обрабатываются одним и тем же инструментом;
- все параметры точности обеспечиваются одним и тем же инструментом;

- точная резьба нарезается на полную глубину глухого отверстия, так как фреза не имеет заходной фаски;
- различные материалы обрабатываются одним и тем же инструментом;
- высокая надёжность обработки благодаря хорошему стружкоудалению;
- высокая производительность фрезерования резьбы за счёт более высоких скорости резания и подачи;
- малый крутящий момент на шпинделе даже при обработке крупной резьбы.

Для фрезерования резьбы используются как однорезцовые инструменты, так и многорезцовые, позволяющие обрабатывать несколько витков резьбы за один проход. Обработка многорезцовым инструментом во многом схожа со спиральной обработкой.



При использовании многорезцового инструмента фрезерование резьбы включает следующие шаги:

- Ускоренный подвод к центру отверстия на уровень Z retract.
- Ускоренное перемещение на уровень Z safe.
- Ускоренное перемещение на расстояние длины режущей части инструмента, которое определяется числом и размером зубьев фрезы (шагом резьбы).
- Перемещение на рабочей подаче к началу спирали.
- Обработка вдоль одного витка спирали с шагом, равным шагу резьбы.
- Отвод к центру отверстия.
- Если одного витка спирали недостаточно для обработки резьбы на полную глубину отверстия, опускание на длину режущей части и перемещение по спирали повторяются до достижения нужной глубины резьбы.
- Ускоренный возврат на уровень Z retract.

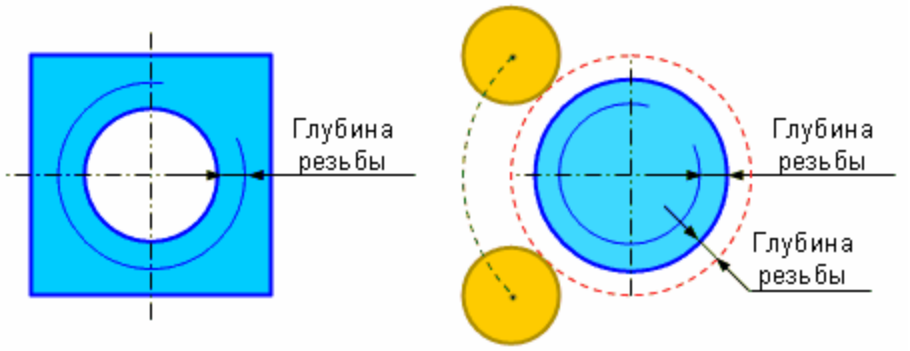
Если применяются дополнительные черновые и чистовые проходы, то указанные выше шаги повторяются до достижения заданной глубины резьбы.

Параметр Thread type (тип резьбы) задаёт, является ли резьба External (наружной) или Internal (внутренней).

Бывают случаи, когда технология требует, чтобы резьба обрабатывалась сверху вниз, и случаи, когда резьба нарезается снизу вверх. Этот цикл поддерживает обе технологии.

Вид резьбы, правая или левая, определяется параметром Thread spiral direction (направление спирали резьбы). Для спиральной обработки удобно задавать направление спирали в соответствии с направлением вращения шпинделя. При заданном направлении Follow (попутное) направления вращения инструмента и закрутки спирали совпадают, для направления Counter (встречное) они противоположны.

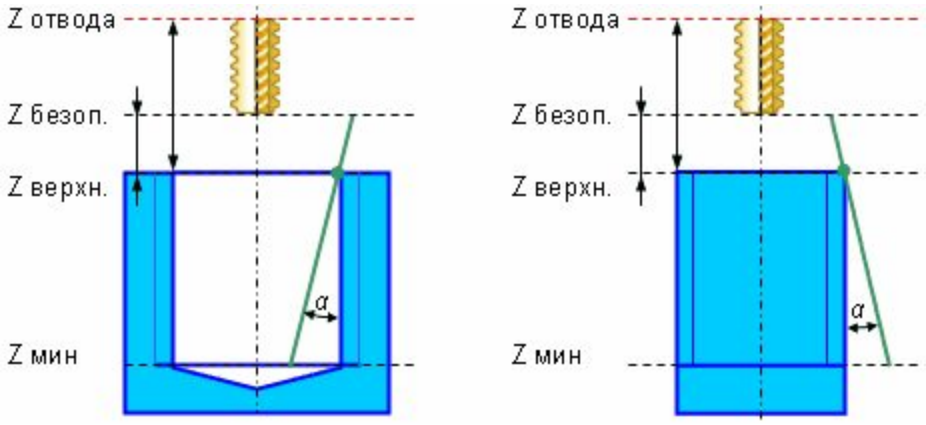
Параметр Threading depth (глубина резьбы) используется для задания разницы между наружным и внутренним диаметрами резьбы.



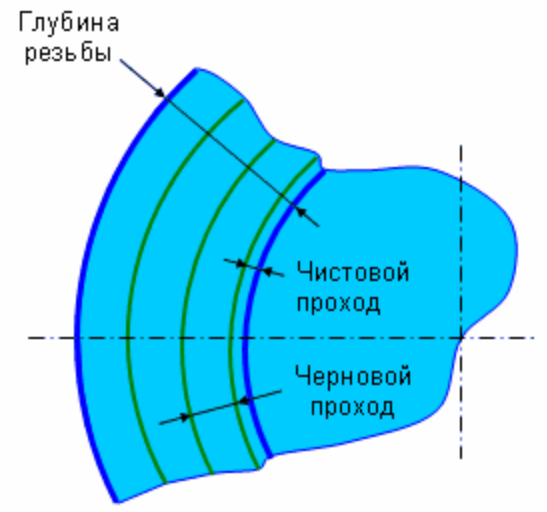
Параметр Bottom circle pass (проход по окружности на дне) задаёт, выполняется ли круговое перемещение при достижении дна резьбы.

Многозаходная резьба обрабатывается, если параметр Thread start count (число заходов резьбы) больше 1. Если число заходов равно 1, обрабатывается однозаходная резьба.

Цикл фрезерования резьбы поддерживает обработку конической резьбы. Параметр Taper angle (угол конуса) используется для задания угла конуса резьбы в градусах. Угол конуса измеряется от уровня верха отверстия (бобышки). Положительное направление угла для обработки конической резьбы отверстия — направление к центру отверстия. Положительное направление угла для обработки конической резьбы бобышки — направление от центра бобышки.



Фрезерование резьбы может выполняться в несколько проходов. Для обработки этих циклов используйте параметры Roughing passes count (число черновых проходов), Roughing pass step (шаг чернового прохода) и Finish pass step (шаг чистового прохода).



Параметр Toolpath type (тип траектории) задаёт тип траектории в соответствии с используемым типом инструмента. Он может быть одним из следующих:

- Continuous (непрерывный). Этот тип траектории используется для однорезцового инструмента, который формирует только один виток резьбы за каждый виток спирали. Геометрически траектория представляет собой непрерывную спираль.
- Transition along the axis (переход вдоль оси). Этот тип используется для многорезцового инструмента, который формирует несколько витков резьбы за один виток спирали траектории. Траектория состоит из последовательных витков спирали, соединённых ускоренными переходами длиной с режущую часть вдоль оси спирали.

Параметр Cutting edge length (длина режущей кромки) используется только для типа траектории Transition along the axis. Он задаёт длину переходов между соседними витками спирали.

Параметр Number of spiral turns (число витков спирали) используется для обработки резьбы с шагом меньше расстояния между зубьями инструмента. Этот параметр обычно не используется (равен 1).

Параметр Start angular position (начальное угловое положение) задаёт начальное угловое положение шпинделя в градусах, с которого начинается фрезерование резьбы.

Параметры

Массив CLD	Имя CLD	Описание
CLD[1]	CLD.SubCmd	Тип команды: ON(71) – включение цикла, CALL(52) – вызов цикла, OFF(72) – выключение цикла.
CLD[2]	CLD.SubType	Идентификатор типа цикла: W5DThreadMill(490).
CLD[3]	CLD.CLParams(1)	Nx, координата X вектора нормали инструмента
CLD[4]	CLD.CLParams(2)	Ny, координата Y вектора нормали инструмента
CLD[5]	CLD.CLParams(3)	Nz, координата Z вектора нормали инструмента
CLD[6]	CLD.CLParams(4)	Sf, нормальное расстояние от текущего положения инструмента до уровня безопасной плоскости
CLD[7]	CLD.CLParams(5)	Tp, нормальное расстояние от текущего положения инструмента до уровня верха отверстия
CLD[8]	CLD.CLParams(6)	Bt, нормальное расстояние от текущего положения инструмента до уровня дна отверстия
CLD[9]	CLD.CLParams(7)	Единицы рабочей подачи: 0 – мм/об, 1 – мм/мин
CLD[10]	CLD.CLParams(8)	Значение рабочей подачи
CLD[11]	CLD.CLParams(9)	Единицы подачи подвода: 0 – мм/об, 1 – мм/мин
CLD[12]	CLD.CLParams(10)	Значение подачи подвода

Массив CLD	Имя CLD	Описание
CLD[13]	CLD.CLParams(11)	Единицы подачи возврата: 0 – мм/об, 1 – мм/мин
CLD[14]	CLD.CLParams(12)	Значение подачи возврата
CLD[16]	CLD.CLParams(14)	D, наружный диаметр резьбы
CLD[17]	CLD.CLParams(15)	St, шаг резьбы
CLD[18]	CLD.CLParams(16)	Тип резьбы: 0 – внутренняя, 1 – наружная
CLD[19]	CLD.CLParams(17)	Направление спирали резьбы: 0 – встречное, 1 – попутное, 2 – правое, 3 – левое. В случаях 0 и 1 направление резьбы зависит от направления вращения инструмента, в случаях 2 и 3 они независимы.
CLD[20]	CLD.CLParams(18)	Последовательность обработки: 0 – сверху вниз, 1 – снизу вверх
CLD[21]	CLD.CLParams(19)	Число заходов резьбы
CLD[22]	CLD.CLParams(20)	Глубина резьбы (разница между наружным и внутренним радиусами)
CLD[23]	CLD.CLParams(21)	Шаг черновых проходов
CLD[24]	CLD.CLParams(22)	Число черновых проходов
CLD[25]	CLD.CLParams(23)	Шаг чистового прохода
CLD[26]	CLD.CLParams(24)	Необходимость прохода по окружности на дне: 0 – не выполнять последний проход по окружности, 1 –

Массив CLD	Имя CLD	Описание
		выполнять последний проход по окружности
CLD[27]	CLD.CLParams(25)	Угол конуса в градусах
CLD[28]	CLD.CLParams(26)	Метод формирования спирали: 0 – непрерывно на всю глубину отверстия, 1 – с переходом для многозубого инструмента
CLD[29]	CLD.CLParams(27)	Длина рабочей части резцов (определяется числом зубьев). Эта величина определяет длину перехода для метода формирования спирали «с переходом».
CLD[30]	CLD.CLParams(28)	Число оборотов спирали для метода формирования спирали «с переходом» (обычно равно 1, но может быть больше 1 при фрезеровании резьбы с шагом меньше шага резца).
CLD[31]	CLD.CLParams(29)	Начальное угловое положение шпинделя в градусах

Доступ из .NET

Помимо базового `OnExtCycle`, этот цикл доставляется через обработчик семейства циклов отверстий `OnHoleExtCycle`:

```
public override void OnHoleExtCycle(ICLDExtCycleCommand cmd, CLDArray cld)
```

Общая модель .NET описана в разделе Расширенный цикл (EXTCYCLE).

Сопутствующие ссылки

- Расширенный цикл (EXTCYCLE)
- Модель доступа к CLData

EXTCYCLE WProbing - Измерительный цикл

`WProbing` применяется для измерительных операций. Он измеряет деталь или инструмент на основе одного из выбранных циклов с подводом и возвратом к началу цикла. Каждый цикл имеет собственную траекторию, точки касания и параметры, на основе которых выполняются дальнейшие расчёты.

`WProbing` хранит параметры по коду, а не по индексу, как другие циклы команды `EXTCYCLE`.

Ниже приведены примеры синтаксиса работы:

Постпроцессор (sppx):

```
program ExtCycle
  if cld[1]=52 then begin ! Вызов цикла
    case cld[2] of
      500: begin ! Все измерительные циклы
        case cmdPrm.Int[-2] of ! Подтип измерительного цикла
          11: begin ! Мой цикл отверстия P9814
            Output "G65 P9814 E" + cmdPrm.Flt[101] + " Q" + cmdPrm.Flt[102]
          end
          12: begin ! Мой цикл бобышки P9812
            Output "G65 P9812 E" + cmdPrm.Flt[101] + " Q" + cmdPrm.Flt[102]
          end
        end
      end
    end ! Конец всех измерительных циклов
  end
end
```

Постпроцессор .NET:

```
public override void OnProbeExtCycle(ICLDExtCycleCommand cmd, CLDArray cld)
{
    if (!cmd.IsCall)
        return;
    switch (cmd.Prm.Int[-2]) // Тип цикла
    {
        case 11: //Мой цикл отверстия P9814
            nc.WriteLine($"G65 P9814 E{cmd.Prm.Flt[101]} Q{cmd.Prm.Flt[102]}");
            break;
        case 12: //Мой цикл бобышки P9812
            nc.WriteLine($"G65 P9812 E{cmd.Prm.Flt[101]} Q{cmd.Prm.Flt[102]}");
            break;
```

```
    }
}
```

Более подробный пример измерительных циклов можно посмотреть внутри дистрибутивного постпроцессора Fanuc (30i)_Mill_DN.dll.

Подтип измерительных циклов имеет значение «500».

Оператор cmd (параметры с уникальным кодом) описан здесь.

Обработчик OnProbeExtCycle

Для .NET-постпроцессора добавлен отдельный обработчик `OnProbeExtCycle`, который выполняется на команде CLData EXTCYCLE в случае измерительных циклов (у которых SubCode равен 500).

Объявление обработчика `OnProbeExtCycle`:

```
public virtual void OnProbeExtCycle(ICLDExtCycleCommand cmd, CLDArray cld)
```

Параметры

Тип	Имя	Описание
ICLDExtCycleCommand	cmd	Текущая команда CLData.
CLDArray	cld	Массив cld команды.

Дополнительную информацию об измерительных операциях и циклах можно найти в руководстве пользователя.

Сопутствующие ссылки

- Расширенный цикл (EXTCYCLE)
- Модель доступа к CLData

Дополнительные параметры

NCAction

Элемент `NCAction` имеет один собственный параметр:

CLDArray	Тип	Описание
CmdPrm.Int[-1]	Integer	Тип измерительного цикла: значение NCAction = 1
CmdPrm.Int[-2]	Integer	SubCode цикла, заданный в свойстве " SubCode for postprocessor " на вкладке `Job Assignment`

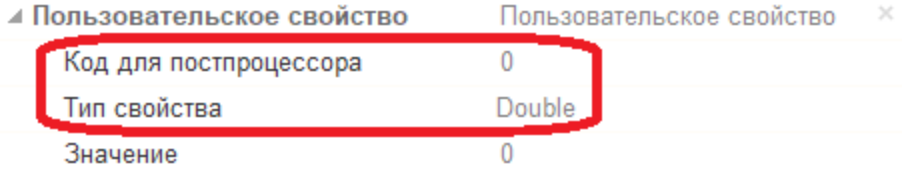
Пользовательские свойства

Пользовательские свойства имеют собственные коды параметров, которые хранятся в циклах и элементе `NCAction`:

CLDArray	Тип	Описание
CmdPrm.Int[-10]	Integer	Set WCS offset - режим. 0 - Global (все G54-G59); 1 - One WCS (одна из G54-G59); 2 - Parametrical offset (G10 L2 P1 XYZ); 3 - Local CS offset (G52 XYZ)
CmdPrm.Int[-11]	Integer	Set WCS offset - номер системы координат (Set WCS offset - режим = 1 или 2).
CmdPrm.Int[-12]	Integer	Set offset of tool number - номер инструмента
CmdPrm.Int[-13]	Integer	Set offset of tool number - Corrector 1
CmdPrm.Int[-14]	Integer	Set offset of tool number - Corrector 2

CLDArray	Тип	Описание
CmdPrm.Int[-15]	Integer	Check for broken tool - номер инструмента
CmdPrm.Int[-16]	Integer	Check for broken tool - Corrector 1
CmdPrm.Int[-17]	Integer	Check for broken tool - Corrector 2
CmdPrm.Int[-18]	Integer	Calibrate the tool probe number - номер щупа
CmdPrm.Int[-19]	Integer	Calibrate the tool probe number - Corrector 1
CmdPrm.Int[-20]	Integer	Calibrate the tool probe number - Corrector 2
CmdPrm.Int[-21]	Integer	Calibrate the part probe number - номер щупа
CmdPrm.Int[-22]	Integer	Calibrate the part probe number - Corrector 1
CmdPrm.Int[-23]	Integer	Calibrate the part probe number - Corrector 2
CmdPrm.Int[-24]	Integer	Write to report - номер компонента
CmdPrm.Int[-25]	Integer	Write to report - номер элемента

Для элемента `Custom property` на вкладке `Job assignment` операции измерительного цикла вы вручную задаёте код параметра, который будет храниться в CLData, а также можете задать тип параметра. Код пользовательского параметра не должен совпадать с другими кодами.



Probe On/Off

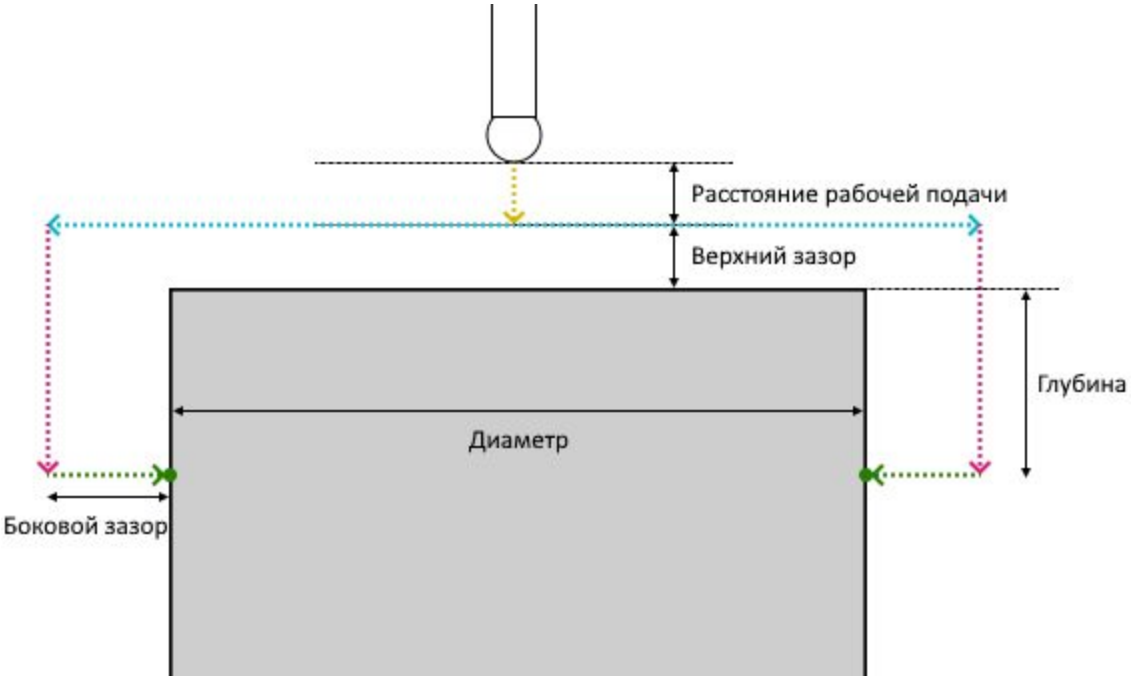
Команды включения и выключения цикла имеют параметры:

CLDArray	Тип	Описание
CmdPrm.Int[-1]	Integer	Тип измерительного цикла: значение Cycle On/Off = 0
CmdPrm.Int[-2]	Integer	Cycle On/Off SubCode = -500
CmdPrm.Bol[-3]	Boolean	Режим On/Off. True — «On», false — «Off»

Сопутствующие ссылки

- Измерительный цикл (WProbing)
- Расширенный цикл (EXTCYCLE)
- Модель доступа к CLData

Параметры измерения бобышки



Измерение бобышки состоит из следующих шагов:

1. Инструмент перемещается на расстояние, заданное в `Feed distance`, и подводится к начальной позиции цикла. Перемещение выполняется на «подаче подвода»;
2. Инструмент подводится к точке, удалённой от первой точки касания вдоль её `Target vector` на расстояние `Side clearance`, но поднятой на значение `Top clearance` + значение `Depth`. Перемещение выполняется на «подаче длинных переходов»;
3. Инструмент опускается на расстояние, равное значению `Top clearance` + значение `Depth`. Перемещение выполняется на «подаче длинных переходов»;
4. Инструмент подводится к первой точке касания (перемещаясь на «рабочей подаче»), а затем возвращается (перемещаясь на «подаче длинных переходов») в предыдущее положение;
5. Инструмент поднимается на расстояние, равное значению `Top clearance` + значение `Depth`. Перемещение выполняется на «подаче длинных переходов»;
6. Инструмент возвращается в начальную позицию;
7. Шаги 2-6 повторяются столько раз, сколько точек касания в цикле;
8. Инструмент поднимается на расстояние, равное `Feed distance`.

Параметры

CLDArray	Тип	Описание
CmdPrm.Int[-1]	Integer	Тип измерительного цикла: значение Boss probing = 5

CLDArray	Тип	Описание
CmdPrm.Int[-2]	Integer	SubCode цикла, заданный в свойстве " SubCode for postprocessor " на вкладке `Job Assignment`
CmdPrm.Flt[-50]	Double	Feed distance, расстояние до начальной позиции цикла
CmdPrm.Flt[-51]	Double	Depth, расстояние от верхней стороны до центра точек касания
CmdPrm.Flt[-52]	Double	Диаметр измеряемой бобышки
CmdPrm.Flt[-55]	Double	Top clearance, расстояние от верхней стороны
CmdPrm.Flt[-56]	Double	Side clearance, расстояние подвода к точке касания
CmdPrm.Int[-59]	Integer	Вариант цикла. 0 - прямоугольный режим, 1 - угловой режим
CmdPrm.Flt[-60]	Double	Start angle (для варианта цикла = 1). Начальный угол точек касания
CmdPrm.Flt[-61]	Double	Angular step (для варианта цикла = 1). Радиальное расстояние между точками касания
CmdPrm.Int[-62]	Integer	Step count (для варианта цикла = 1). Число точек касания
CmdPrm.Flt[-100]	Double	Значение первой точки касания по оси X

CLDArray	Тип	Описание
CmdPrm.Flt[-101]	Double	Значение первой точки касания по оси Y
CmdPrm.Flt[-102]	Double	Значение первой точки касания по оси Z
CmdPrm.Flt[-103]	Double	Значение первого целевого вектора по оси X
CmdPrm.Flt[-104]	Double	Значение первого целевого вектора по оси Y
CmdPrm.Flt[-105]	Double	Значение первого целевого вектора по оси Z
CmdPrm.Flt[-100-((N-1)*6)]	Double	Значение прочей точки касания по оси X. N - номер точки касания
CmdPrm.Flt[-101-((N-1)*6)]	Double	Значение прочей точки касания по оси Y. N - номер точки касания
CmdPrm.Flt[-102-((N-1)*6)]	Double	Значение прочей точки касания по оси Z. N - номер точки касания
CmdPrm.Flt[-103-((N-1)*6)]	Double	Значение прочего целевого вектора по оси X. N - номер точки касания
CmdPrm.Flt[-104-((N-1)*6)]	Double	Значение прочего целевого вектора по оси Y. N - номер точки касания
CmdPrm.Flt[-105-((N-1)*6)]	Double	Значение прочего целевого вектора по оси Z. N - номер точки касания
CmdPrm.Flt[-200]	Double	Значение исходной первой точки касания по оси X,

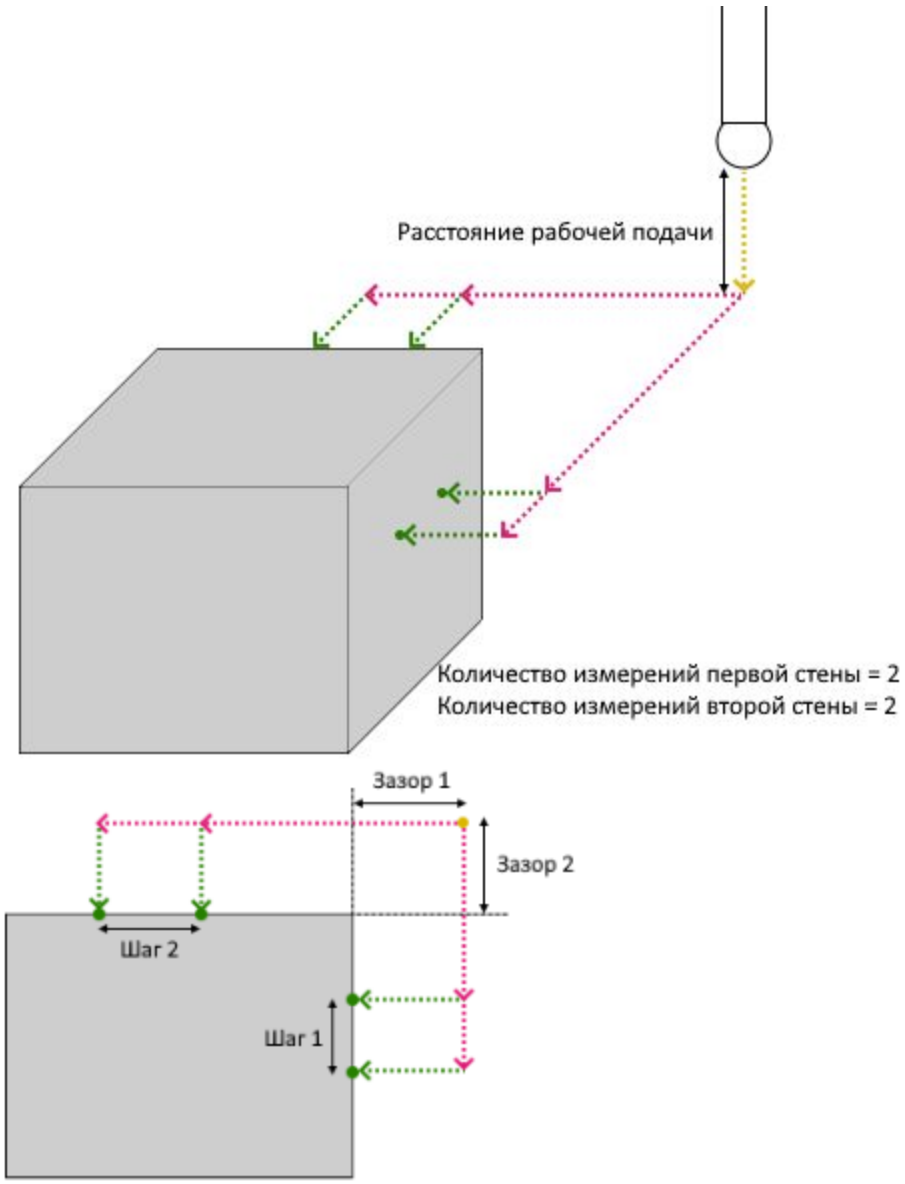
CLDArray	Тип	Описание
		заданное в Job Assignment
CmdPrm.Flt[-201]	Double	Значение исходной первой точки касания по оси Y, заданное в Job Assignment
CmdPrm.Flt[-202]	Double	Значение исходной первой точки касания по оси Z, заданное в Job Assignment
CmdPrm.Flt[-203]	Double	Значение исходного первого целевого вектора по оси X, заданное в Job Assignment
CmdPrm.Flt[-204]	Double	Значение исходного первого целевого вектора по оси Y, заданное в Job Assignment
CmdPrm.Flt[-205]	Double	Значение исходного первого целевого вектора по оси Z, заданное в Job Assignment
CmdPrm.Flt[-200-((N-1)*6)]	Double	Значение прочей точки касания от исходной точки по оси X. N - номер точки касания
CmdPrm.Flt[-201-((N-1)*6)]	Double	Значение прочей точки касания от исходной точки по оси Y. N - номер точки касания
CmdPrm.Flt[-202-((N-1)*6)]	Double	Значение прочей точки касания от исходной точки по оси Z. N - номер точки касания
CmdPrm.Flt[-203-((N-1)*6)]	Double	Значение прочего целевого вектора от исходного вектора по оси X. N - номер точки касания

CLDArray	Тип	Описание
CmdPrm.Flt[-204-((N-1)*6)]	Double	Значение прочего целевого вектора от исходного вектора по оси Y. N - номер точки касания
CmdPrm.Flt[-205-((N-1)*6)]	Double	Значение прочего целевого вектора от исходного вектора по оси Z. N - номер точки касания
CmdPrm.Flt[-66]	Integer	Compensate dimensions, возвращаемые значения: 0 - Off, 1 - On

Сопутствующие ссылки

- Измерительный цикл (WProbing)
- Расширенный цикл (EXTCYCLE)
- Модель доступа к CLData

Параметры измерения наружного угла по двум стенкам



Измерение наружного угла по двум стенкам состоит из следующих шагов:

1. Инструмент перемещается на расстояние, заданное в `Feed distance`, и подводится к начальной точке измерительного цикла — точке пересечения первой точки касания первой стороны и первой точки касания второй стороны на расстоянии, заданном в `Clearance 1`, `Clearance 2` соответственно, вдоль ортогонального направления их векторов. Перемещение на «подаче подвода»;
2. Инструмент подводится к точке, удалённой от первой точки касания первой стороны вдоль её `Target vector` на расстояние `Clearance 1` (перемещаясь на «подаче длинных переходов»), подводится к первой точке касания первой стороны (перемещаясь на

«рабочей подаче»), а затем возвращается (перемещаясь на «подаче длинных переходов») в предыдущее положение;

3. Если `First wall measure count` >1, то:
4. Инструмент перемещается к следующей точке касания на текущей стороне на расстояние, заданное в `Clearance 1`, в направлении её целевого вектора;
5. Инструмент подводится к точке касания (перемещаясь на «рабочей подаче»), а затем возвращается (перемещаясь на «подаче длинных переходов») в предыдущее положение;
6. Шаги 3 повторяются столько раз, сколько указано в `First wall measure count`;
7. Инструмент возвращается к начальной точке цикла. Перемещение на «подаче длинных переходов»;
8. Инструмент подводится к точке, удалённой от первой точки касания второй стороны вдоль её `Target vector` на расстояние `Clearance 2` (перемещаясь на «подаче длинных переходов»), подводится к первой точке касания второй стороны (перемещаясь на «рабочей подаче»), а затем возвращается (перемещаясь на «подаче длинных переходов») в предыдущее положение;
9. Если `Second wall measure count` >1, то:
10. Инструмент перемещается к следующей точке касания на текущей стороне на расстояние, заданное в `Clearance 2`, в направлении её целевого вектора;
11. Инструмент подводится к точке касания (перемещаясь на «рабочей подаче»), а затем возвращается (перемещаясь на «подаче длинных переходов») в предыдущее положение;
12. Шаги 7 повторяются столько раз, сколько указано в `Second wall measure count`;
13. Инструмент возвращается к начальной точке цикла. Перемещение на «подаче длинных переходов»;
14. Инструмент поднимается на расстояние, равное `Feed distance`.

Параметры

CLDArray	Тип	Описание
CmdPrm.Int[-1]	Integer	Тип измерительного цикла: значение Double wall external corner probing = 14
CmdPrm.Int[-2]	Integer	SubCode цикла, заданный в свойстве " SubCode for postprocessor " на вкладке `Job Assignment`
CmdPrm.Flt[-50]	Double	Feed distance, расстояние до начальной позиции цикла

CLDArray	Тип	Описание
CmdPrm.Flt[-56]	Double	Clearance 1, расстояние подвода к точкам касания на первой стороне
CmdPrm.Flt[-57]	Double	Clearance 2, расстояние подвода к точкам касания на второй стороне
CmdPrm.Int[-63]	Integer	First wall measure count, число точек касания на первой стороне
CmdPrm.Int[-64]	Integer	Second wall measure count, число точек касания на второй стороне
CmdPrm.Flt[-65]	Double	Step 1, расстояние между точками касания на первой стенке
CmdPrm.Flt[-67]	Double	Step 2, расстояние между точками касания на второй стенке
CmdPrm.Flt[-100]	Double	Значение первой точки касания на первой стороне по оси X
CmdPrm.Flt[-101]	Double	Значение первой точки касания на первой стороне по оси Y
CmdPrm.Flt[-102]	Double	Значение первой точки касания на первой стороне по оси Z
CmdPrm.Flt[-103]	Double	Значение первого целевого вектора на первой стороне по оси X

CLDArray	Тип	Описание
CmdPrm.Flt[-104]	Double	Значение первого целевого вектора на первой стороне по оси Y
CmdPrm.Flt[-105]	Double	Значение первого целевого вектора на первой стороне по оси Z
CmdPrm.Flt[-100-((N-1)*6)]	Double	Значение прочей точки касания на первой/второй стороне по оси X. N - номер точки касания
CmdPrm.Flt[-101-((N-1)*6)]	Double	Значение прочей точки касания на первой/второй стороне по оси Y. N - номер точки касания
CmdPrm.Flt[-102-((N-1)*6)]	Double	Значение прочей точки касания на первой/второй стороне по оси Z. N - номер точки касания
CmdPrm.Flt[-103-((N-1)*6)]	Double	Значение прочего целевого вектора на первой/второй стороне по оси X. N - номер точки касания
CmdPrm.Flt[-104-((N-1)*6)]	Double	Значение прочего целевого вектора на первой/второй стороне по оси Y. N - номер точки касания
CmdPrm.Flt[-105-((N-1)*6)]	Double	Значение прочего целевого вектора на первой/второй стороне по оси Z. N - номер точки касания
CmdPrm.Flt[-100-6*C]	Double	Значение первой точки касания на второй стороне

CLDArray	Тип	Описание
		по оси X. C - число точек касания на первой стороне
CmdPrm.Flt[-101-6*C]	Double	Значение первой точки касания на второй стороне по оси Y. C - число точек касания на первой стороне
CmdPrm.Flt[-102-6*C]	Double	Значение первой точки касания на второй стороне по оси Z. C - число точек касания на первой стороне
CmdPrm.Flt[-103-6*C]	Double	Значение первого целевого вектора на второй стороне по оси X. C - число точек касания на первой стороне
CmdPrm.Flt[-104-6*C]	Double	Значение первого целевого вектора на второй стороне по оси Y. C - число точек касания на первой стороне
CmdPrm.Flt[-105-6*C]	Double	Значение первого целевого вектора на второй стороне по оси Z. C - число точек касания на первой стороне
CmdPrm.Flt[-100-6*C-((N-1)*6)]	Double	Значение прочей точки касания на второй стороне по оси X. N - номер точки касания. C - число точек касания на первой стороне
CmdPrm.Flt[-101-6*C-((N-1)*6)]	Double	Значение прочей точки касания на второй стороне по оси Y. N - номер точки касания на второй стороне. C - число точек касания на первой стороне

CLDArray	Тип	Описание
CmdPrm.Flt[-102-6*C-((N-1)*6)]	Double	Значение прочей точки касания на второй стороне по оси Z. N - номер точки касания на второй стороне. C - число точек касания на первой стороне
CmdPrm.Flt[-103-6*C-((N-1)*6)]	Double	Значение прочего целевого вектора на второй стороне по оси X. N - номер точки касания на второй стороне. C - число точек касания на первой стороне
CmdPrm.Flt[-104-6*C-((N-1)*6)]	Double	Значение прочего целевого вектора на второй стороне по оси Y. N - номер точки касания на второй стороне. C - число точек касания на первой стороне
CmdPrm.Flt[-105-6*C-((N-1)*6)]	Double	Значение прочего целевого вектора на второй стороне по оси Z. N - номер точки касания на второй стороне. C - число точек касания на первой стороне
CmdPrm.Flt[-200]	Double	Значение исходной первой точки касания на первой стороне по оси X
CmdPrm.Flt[-201]	Double	Значение исходной первой точки касания на первой стороне по оси Y
CmdPrm.Flt[-202]	Double	Значение исходной первой точки касания на первой стороне по оси Z

CLDArray	Тип	Описание
CmdPrm.Flt[-203]	Double	Значение исходного первого целевого вектора на первой стороне по оси X
CmdPrm.Flt[-204]	Double	Значение исходного первого целевого вектора на первой стороне по оси Y
CmdPrm.Flt[-205]	Double	Значение исходного первого целевого вектора на первой стороне по оси Z
CmdPrm.Flt[-200-((N-1)*6)]	Double	Значение прочей точки касания на первой/второй стороне от исходной точки по оси X. N - номер точки касания
CmdPrm.Flt[-201-((N-1)*6)]	Double	Значение прочей точки касания на первой/второй стороне от исходной точки по оси Y. N - номер точки касания
CmdPrm.Flt[-202-((N-1)*6)]	Double	Значение прочей точки касания на первой/второй стороне от исходной точки по оси Z. N - номер точки касания
CmdPrm.Flt[-203-((N-1)*6)]	Double	Значение прочего целевого вектора на первой/второй стороне от исходного вектора по оси X. N - номер точки касания
CmdPrm.Flt[-204-((N-1)*6)]	Double	Значение прочего целевого вектора на первой/второй стороне от исходного

CLDArray	Тип	Описание
		вектора по оси Y. N - номер точки касания
CmdPrm.Flt[-205-((N-1)*6)]	Double	Значение прочего целевого вектора на первой/второй стороне от исходного вектора по оси Z. N - номер точки касания
CmdPrm.Flt[-200-6*C]	Double	Значение первой точки касания на второй стороне по оси X, заданное в Job Assignment. C - число точек касания на первой стороне
CmdPrm.Flt[-201-6*C]	Double	Значение первой точки касания на второй стороне по оси Y, заданное в Job Assignment. C - число точек касания на первой стороне
CmdPrm.Flt[-202-6*C]	Double	Значение первой точки касания на второй стороне по оси Z, заданное в Job Assignment. C - число точек касания на первой стороне
CmdPrm.Flt[-203-6*C]	Double	Значение первого целевого вектора на второй стороне по оси X, заданное в Job Assignment. C - число точек касания на первой стороне
CmdPrm.Flt[-204-6*C]	Double	Значение первого целевого вектора на второй стороне по оси Y, заданное в Job Assignment. C - число точек касания на первой стороне
CmdPrm.Flt[-205-6*C]	Double	Значение первого целевого вектора на второй стороне

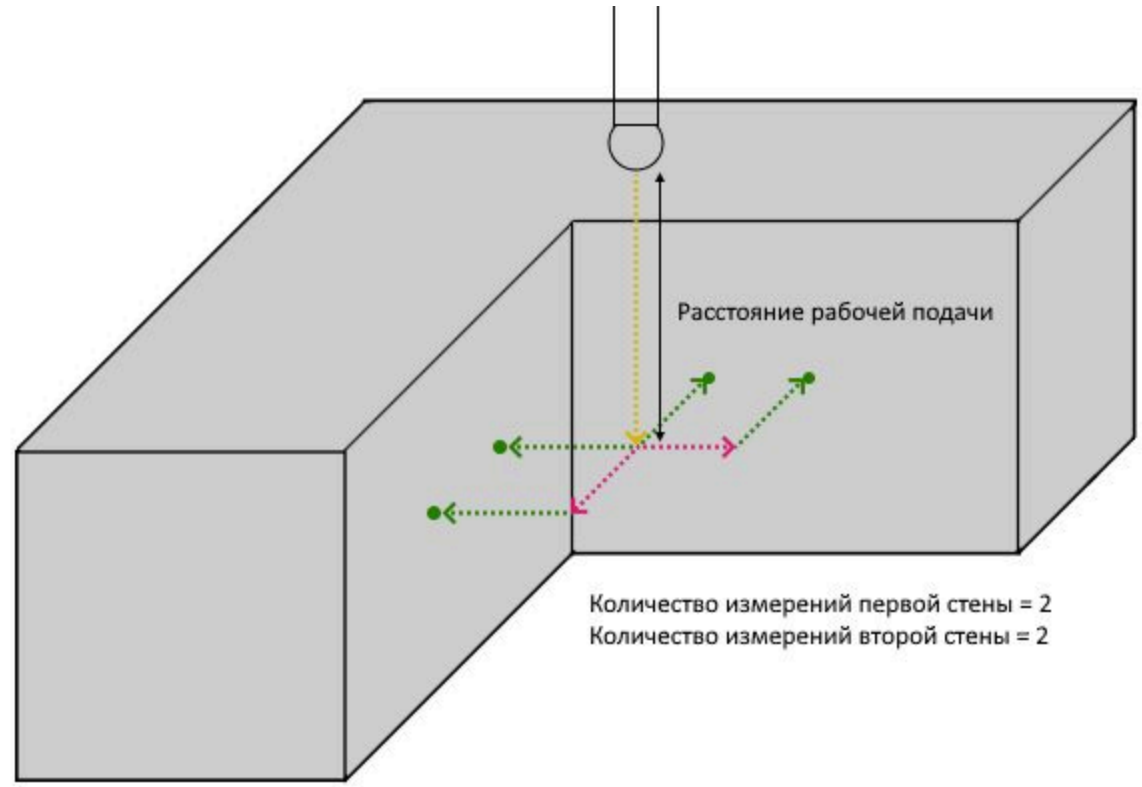
CLDArray	Тип	Описание
		по оси Z, заданное в Job Assignment. C - число точек касания на первой стороне
CmdPrm.Flt[-200-6*C-((N-1)*6)]	Double	Значение прочей точки касания на второй стороне от исходной точки по оси X. N - номер точки касания. C - число точек касания на первой стороне
CmdPrm.Flt[-201-6*C-((N-1)*6)]	Double	Значение прочей точки касания на второй стороне от исходной точки по оси Y. N - номер точки касания на второй стороне. C - число точек касания на первой стороне
CmdPrm.Flt[-202-6*C-((N-1)*6)]	Double	Значение прочей точки касания на второй стороне от исходной точки по оси Z. N - номер точки касания на второй стороне. C - число точек касания на первой стороне
CmdPrm.Flt[-203-6*C-((N-1)*6)]	Double	Значение прочего целевого вектора на второй стороне от исходного вектора по оси X. N - номер точки касания на второй стороне. C - число точек касания на первой стороне
CmdPrm.Flt[-204-6*C-((N-1)*6)]	Double	Значение прочего целевого вектора на второй стороне от исходного вектора по оси Y. N - номер точки касания на второй стороне. C - число

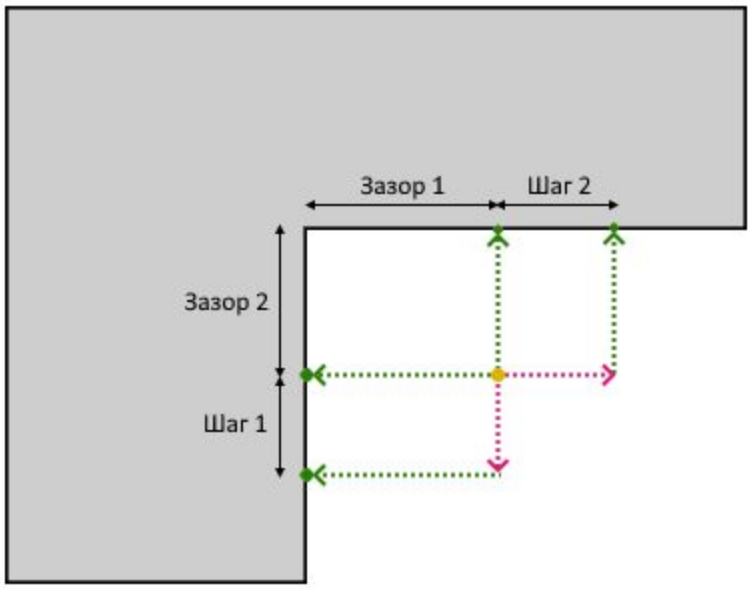
CLDArray	Тип	Описание
		точек касания на первой стороне
CmdPrm.Flt[-205-6*C-((N-1)*6)]	Double	Значение прочего целевого вектора на второй стороне от исходного вектора по оси Z. N - номер точки касания на второй стороне. C - число точек касания на первой стороне
CmdPrm.Flt[-66]	Integer	Compensate dimensions, возвращаемые значения: 0 - Off, 1 - On

Сопутствующие ссылки

- Измерительный цикл (WProbing)
- Расширенный цикл (EXTCYCLE)
- Модель доступа к CLData

Параметры измерения внутреннего угла по двум стенкам





Измерение внутреннего угла по двум стенкам состоит из следующих шагов:

1. Инструмент перемещается на расстояние, заданное в `Feed distance`, и подводится к начальной точке измерительного цикла — точке пересечения первой точки касания первой стороны и первой точки касания второй стороны на расстоянии, заданном в `Clearance 1`, `Clearance 2` соответственно, вдоль их векторов. Перемещение на «подаче подвода»;

2. Инструмент подводится к первой точке касания первой стороны (перемещаясь на «рабочей подаче»), а затем возвращается (перемещаясь на «подаче длинных переходов») в предыдущее положение;
3. Если `First wall measure count` >1, то:
4. Инструмент перемещается к следующей точке касания на текущей стороне на расстояние, заданное в `Clearance 1`, в направлении её целевого вектора;
5. Инструмент подводится к точке касания (перемещаясь на «рабочей подаче»), а затем возвращается (перемещаясь на «подаче длинных переходов») в предыдущее положение;
6. Шаги 3 повторяются столько раз, сколько указано в `First wall measure count`;
7. Инструмент возвращается к начальной точке цикла. Перемещение на «подаче длинных переходов»;
8. Инструмент подводится к первой точке касания второй стороны (перемещаясь на «рабочей подаче»), а затем возвращается (перемещаясь на «подаче длинных переходов») в предыдущее положение;
9. Если `Second wall measure count` >1, то:
10. Инструмент перемещается к следующей точке касания на текущей стороне на расстояние, заданное в `Clearance 2`, в направлении её целевого вектора;
11. Инструмент подводится к точке касания (перемещаясь на «рабочей подаче»), а затем возвращается (перемещаясь на «подаче длинных переходов») в предыдущее положение;
12. Шаги 7 повторяются столько раз, сколько указано в `Second wall measure count`;
13. Инструмент возвращается к начальной точке цикла. Перемещение на «подаче длинных переходов»;
14. Инструмент поднимается на расстояние, равное `Feed distance`.

Параметры

CLDArray	Тип	Описание
CmdPrm.Int[-1]	Integer	Тип измерительного цикла: значение Double wall internal corner probing = 13
CmdPrm.Int[-2]	Integer	SubCode цикла, заданный в свойстве " SubCode for postprocessor " на вкладке `Job Assignment`
CmdPrm.Flt[-50]	Double	Feed distance, расстояние до начальной позиции цикла
CmdPrm.Flt[-56]	Double	Clearance 1, расстояние подвода к точкам касания на

CLDArray	Тип	Описание
		первой стороне
CmdPrm.Flt[-57]	Double	Clearance 2, расстояние подвода к точкам касания на второй стороне
CmdPrm.Int[-63]	Integer	First wall measure count, число точек касания на первой стороне
CmdPrm.Int[-64]	Integer	Second wall measure count, число точек касания на второй стороне
CmdPrm.Flt[-65]	Double	Step 1, расстояние между точками касания на первой стенке
CmdPrm.Flt[-67]	Double	Step 2, расстояние между точками касания на второй стенке
CmdPrm.Flt[-100]	Double	Значение первой точки касания на первой стороне по оси X
CmdPrm.Flt[-101]	Double	Значение первой точки касания на первой стороне по оси Y
CmdPrm.Flt[-102]	Double	Значение первой точки касания на первой стороне по оси Z
CmdPrm.Flt[-103]	Double	Значение первого целевого вектора на первой стороне по оси X
CmdPrm.Flt[-104]	Double	Значение первого целевого вектора на первой стороне по оси Y

CLDArray	Тип	Описание
CmdPrm.Flt[-105]	Double	Значение первого целевого вектора на первой стороне по оси Z
CmdPrm.Flt[-100-((N-1)*6)]	Double	Значение прочей точки касания на первой стороне по оси X. N - номер точки касания на первой стороне
CmdPrm.Flt[-101-((N-1)*6)]	Double	Значение прочей точки касания на первой стороне по оси Y. N - номер точки касания на первой стороне
CmdPrm.Flt[-102-((N-1)*6)]	Double	Значение прочей точки касания на первой стороне по оси Z. N - номер точки касания на первой стороне
CmdPrm.Flt[-103-((N-1)*6)]	Double	Значение прочего целевого вектора на первой стороне по оси X. N - номер точки касания на первой стороне
CmdPrm.Flt[-104-((N-1)*6)]	Double	Значение прочего целевого вектора на первой стороне по оси Y. N - номер точки касания на первой стороне
CmdPrm.Flt[-105-((N-1)*6)]	Double	Значение прочего целевого вектора на первой стороне по оси Z. N - номер точки касания на первой стороне
CmdPrm.Flt[-100-6*C]	Double	Значение первой точки касания на второй стороне по оси X. C - число точек касания на первой стороне
CmdPrm.Flt[-101-6*C]	Double	Значение первой точки касания на второй стороне

CLDArray	Тип	Описание
		по оси Y. C - число точек касания на первой стороне
CmdPrm.Flt[-102-6*C]	Double	Значение первой точки касания на второй стороне по оси Z. C - число точек касания на первой стороне
CmdPrm.Flt[-103-6*C]	Double	Значение первого целевого вектора на второй стороне по оси X. C - число точек касания на первой стороне
CmdPrm.Flt[-104-6*C]	Double	Значение первого целевого вектора на второй стороне по оси Y. C - число точек касания на первой стороне
CmdPrm.Flt[-105-6*C]	Double	Значение первого целевого вектора на второй стороне по оси Z. C - число точек касания на первой стороне
CmdPrm.Flt[-100-6*C-((N-1)*6)]	Double	Значение прочей точки касания на второй стороне по оси X. N - номер точки касания. C - число точек касания на первой стороне
CmdPrm.Flt[-101-6*C-((N-1)*6)]	Double	Значение прочей точки касания на второй стороне по оси Y. N - номер точки касания на второй стороне. C - число точек касания на первой стороне
CmdPrm.Flt[-102-6*C-((N-1)*6)]	Double	Значение прочей точки касания на второй стороне по оси Z. N - номер точки касания на второй стороне.

CLDArray	Тип	Описание
		C - число точек касания на первой стороне
CmdPrm.Flt[-103-6*C-((N-1)*6)]	Double	Значение прочего целевого вектора на второй стороне по оси X. N - номер точки касания на второй стороне. C - число точек касания на первой стороне
CmdPrm.Flt[-104-6*C-((N-1)*6)]	Double	Значение прочего целевого вектора на второй стороне по оси Y. N - номер точки касания на второй стороне. C - число точек касания на первой стороне
CmdPrm.Flt[-105-6*C-((N-1)*6)]	Double	Значение прочего целевого вектора на второй стороне по оси Z. N - номер точки касания на второй стороне. C - число точек касания на первой стороне
CmdPrm.Flt[-200]	Double	Значение исходной первой точки касания на первой стороне по оси X, заданное в Job Assignment
CmdPrm.Flt[-201]	Double	Значение исходной первой точки касания на первой стороне по оси Y, заданное в Job Assignment
CmdPrm.Flt[-202]	Double	Значение исходной первой точки касания на первой стороне по оси Z, заданное в Job Assignment
CmdPrm.Flt[-203]	Double	Значение исходного первого целевого вектора на первой

CLDArray	Тип	Описание
		стороне по оси X, заданное в Job Assignment
CmdPrm.Flt[-204]	Double	Значение исходного первого целевого вектора на первой стороне по оси Y, заданное в Job Assignment
CmdPrm.Flt[-205]	Double	Значение исходного первого целевого вектора на первой стороне по оси Z, заданное в Job Assignment
CmdPrm.Flt[-200-((N-1)*6)]	Double	Значение прочей точки касания на первой стороне от исходной точки по оси X. N - номер точки касания на первой стороне
CmdPrm.Flt[-201-((N-1)*6)]	Double	Значение прочей точки касания на первой стороне от исходной точки по оси Y. N - номер точки касания на первой стороне
CmdPrm.Flt[-202-((N-1)*6)]	Double	Значение прочей точки касания на первой стороне от исходной точки по оси Z. N - номер точки касания на первой стороне
CmdPrm.Flt[-203-((N-1)*6)]	Double	Значение прочего целевого вектора на первой стороне от исходного вектора по оси X. N - номер точки касания на первой стороне
CmdPrm.Flt[-204-((N-1)*6)]	Double	Значение прочего целевого вектора на первой стороне от исходного вектора по оси

CLDArray	Тип	Описание
		Y. N - номер точки касания на первой стороне
CmdPrm.Flt[-205-((N-1)*6)]	Double	Значение прочего целевого вектора на первой стороне от исходного вектора по оси Z. N - номер точки касания на первой стороне
CmdPrm.Flt[-200-6*C]	Double	Значение первой точки касания на второй стороне по оси X, заданное в Job Assignment. C - число точек касания на первой стороне
CmdPrm.Flt[-201-6*C]	Double	Значение первой точки касания на второй стороне по оси Y, заданное в Job Assignment. C - число точек касания на первой стороне
CmdPrm.Flt[-202-6*C]	Double	Значение первой точки касания на второй стороне по оси Z, заданное в Job Assignment. C - число точек касания на первой стороне
CmdPrm.Flt[-203-6*C]	Double	Значение первого целевого вектора на второй стороне по оси X, заданное в Job Assignment. C - число точек касания на первой стороне
CmdPrm.Flt[-204-6*C]	Double	Значение первого целевого вектора на второй стороне по оси Y, заданное в Job Assignment. C - число точек касания на первой стороне
CmdPrm.Flt[-205-6*C]	Double	Значение первого целевого вектора на второй стороне

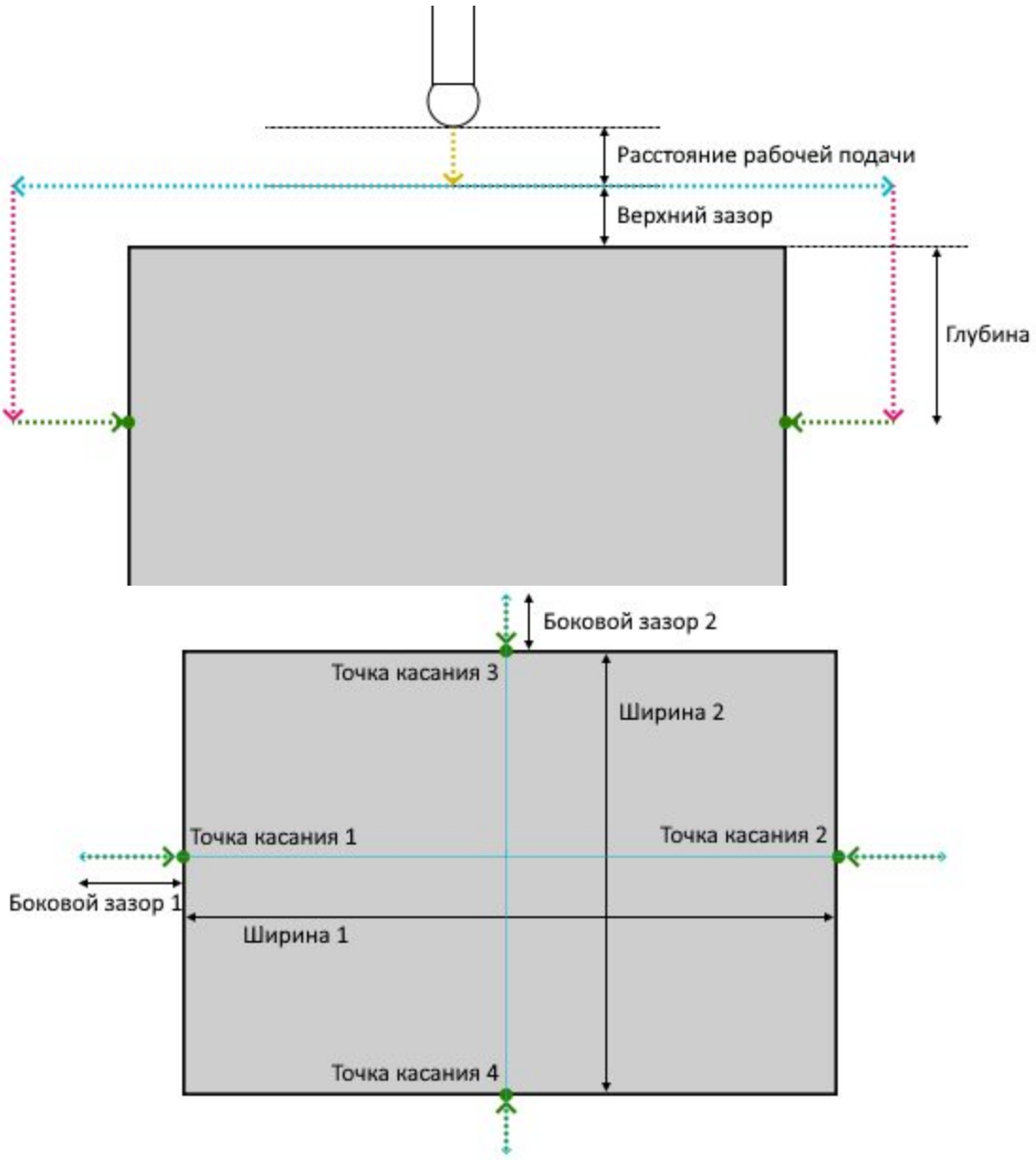
CLDArray	Тип	Описание
		по оси Z, заданное в Job Assignment. C - число точек касания на первой стороне
CmdPrm.Flt[-200-6*C-((N-1)*6)]	Double	Значение прочей точки касания на второй стороне от исходной точки по оси X. N - номер точки касания. C - число точек касания на первой стороне
CmdPrm.Flt[-201-6*C-((N-1)*6)]	Double	Значение прочей точки касания на второй стороне от исходной точки по оси Y. N - номер точки касания на второй стороне. C - число точек касания на первой стороне
CmdPrm.Flt[-202-6*C-((N-1)*6)]	Double	Значение прочей точки касания на второй стороне от исходной точки по оси Z. N - номер точки касания на второй стороне. C - число точек касания на первой стороне
CmdPrm.Flt[-203-6*C-((N-1)*6)]	Double	Значение прочего целевого вектора на второй стороне от исходного вектора по оси X. N - номер точки касания на второй стороне. C - число точек касания на первой стороне
CmdPrm.Flt[-204-6*C-((N-1)*6)]	Double	Значение прочего целевого вектора на второй стороне от исходного вектора по оси Y. N - номер точки касания на второй стороне. C - число

CLDArray	Тип	Описание
		точек касания на первой стороне
CmdPrm.Flt[-205-6*C-((N-1)*6)]	Double	Значение прочего целевого вектора на второй стороне от исходного вектора по оси Z. N - номер точки касания на второй стороне. C - число точек касания на первой стороне

Сопутствующие ссылки

- Измерительный цикл (WProbing)
- Расширенный цикл (EXTCYCLE)
- Модель доступа к CLData

Параметры измерения наружного прямоугольника



Измерение наружного прямоугольника состоит из следующих шагов:

1. Инструмент перемещается на расстояние, заданное в `Feed distance`, и подводится к начальной позиции цикла. Перемещение выполняется на «подаче подвода»;
2. Инструмент подводится к точке, удалённой от первой точки касания вдоль её `Target vector` на расстояние `Side clearance 1`, но поднятой на значение `Top clearance` + значение `Depth`. Перемещение выполняется на «подаче длинных переходов»;
3. Инструмент опускается на расстояние, равное значению `Top clearance` + значение `Depth`. Перемещение выполняется на «подаче длинных переходов»;
4. Инструмент подводится к первой точке касания (перемещаясь на «рабочей подаче»), а затем возвращается (перемещаясь на «подаче длинных переходов») в предыдущее

положение;

5. Инструмент поднимается на расстояние, равное значению `Top clearance` + значение `Depth`. Перемещение выполняется на «подаче длинных переходов»;
6. Инструмент возвращается в начальную позицию;
7. Шаги 2-6 повторяются ещё 3 раза для каждой точки касания;
8. Инструмент поднимается на расстояние, равное `Feed distance`.

Параметры

CLDArray	Тип	Описание
CmdPrm.Int[-1]	Integer	Тип измерительного цикла: значение External rectangle probing = 8
CmdPrm.Int[-2]	Integer	SubCode цикла, заданный в свойстве " SubCode for postprocessor " на вкладке `Job Assignment`
CmdPrm.Flt[-50]	Double	Feed distance, расстояние до начальной позиции цикла
CmdPrm.Flt[-51]	Double	Depth, расстояние от верхней стороны до центра точек касания
CmdPrm.Flt[-53]	Double	Width 1, расстояние от Touch point 1 до Touch point 2
CmdPrm.Flt[-54]	Double	Width 2, расстояние от Touch point 3 до Touch point 4
CmdPrm.Flt[-55]	Double	Top clearance, расстояние от верхней стороны
CmdPrm.Flt[-56]	Double	Side clearance 1, расстояние подвода к точке касания 1 / точке касания 2
CmdPrm.Flt[-57]	Double	Side clearance 2, расстояние подвода к точке касания 3 / точке касания 4

CLDArray	Тип	Описание
CmdPrm.Flt[-100]	Double	Значение первой точки касания по оси X
CmdPrm.Flt[-101]	Double	Значение первой точки касания по оси Y
CmdPrm.Flt[-102]	Double	Значение первой точки касания по оси Z
CmdPrm.Flt[-103]	Double	Значение первого целевого вектора по оси X
CmdPrm.Flt[-104]	Double	Значение первого целевого вектора по оси Y
CmdPrm.Flt[-105]	Double	Значение первого целевого вектора по оси Z
CmdPrm.Flt[-106]	Double	Значение второй точки касания по оси X
CmdPrm.Flt[-107]	Double	Значение второй точки касания по оси Y
CmdPrm.Flt[-108]	Double	Значение второй точки касания по оси Z
CmdPrm.Flt[-109]	Double	Значение второго целевого вектора по оси X
CmdPrm.Flt[-110]	Double	Значение второго целевого вектора по оси Y
CmdPrm.Flt[-111]	Double	Значение второго целевого вектора по оси Z
CmdPrm.Flt[-112]	Double	Значение третьей точки касания по оси X
CmdPrm.Flt[-113]	Double	Значение третьей точки касания по оси Y

CLDArray	Тип	Описание
CmdPrm.Flt[-114]	Double	Значение третьей точки касания по оси Z
CmdPrm.Flt[-115]	Double	Значение третьего целевого вектора по оси X
CmdPrm.Flt[-116]	Double	Значение третьего целевого вектора по оси Y
CmdPrm.Flt[-117]	Double	Значение третьего целевого вектора по оси Z
CmdPrm.Flt[-118]	Double	Значение четвёртой точки касания по оси X
CmdPrm.Flt[-119]	Double	Значение четвёртой точки касания по оси Y
CmdPrm.Flt[-120]	Double	Значение четвёртой точки касания по оси Z
CmdPrm.Flt[-121]	Double	Значение четвёртого целевого вектора по оси X
CmdPrm.Flt[-122]	Double	Значение четвёртого целевого вектора по оси Y
CmdPrm.Flt[-123]	Double	Значение четвёртого целевого вектора по оси Z
CmdPrm.Flt[-200]	Double	Значение первой исходной точки касания по оси X, заданное в Job Assignment
CmdPrm.Flt[-201]	Double	Значение первой исходной точки касания по оси Y, заданное в Job Assignment
CmdPrm.Flt[-202]	Double	Значение первой исходной точки касания по оси Z, заданное в Job Assignment

CLDArray	Тип	Описание
CmdPrm.Flt[-203]	Double	Значение первого исходного целевого вектора по оси X, заданное в Job Assignment
CmdPrm.Flt[-204]	Double	Значение первого исходного целевого вектора по оси Y, заданное в Job Assignment
CmdPrm.Flt[-205]	Double	Значение первого исходного целевого вектора по оси Z, заданное в Job Assignment
CmdPrm.Flt[-206]	Double	Значение второй исходной точки касания по оси X, заданное в Job Assignment
CmdPrm.Flt[-207]	Double	Значение второй исходной точки касания по оси Y, заданное в Job Assignment
CmdPrm.Flt[-208]	Double	Значение второй исходной точки касания по оси Z, заданное в Job Assignment
CmdPrm.Flt[-209]	Double	Значение второго исходного целевого вектора по оси X, заданное в Job Assignment
CmdPrm.Flt[-210]	Double	Значение второго исходного целевого вектора по оси Y, заданное в Job Assignment
CmdPrm.Flt[-211]	Double	Значение второго исходного целевого вектора по оси Z, заданное в Job Assignment
CmdPrm.Flt[-212]	Double	Значение третьей исходной точки касания по оси X, заданное в Job Assignment

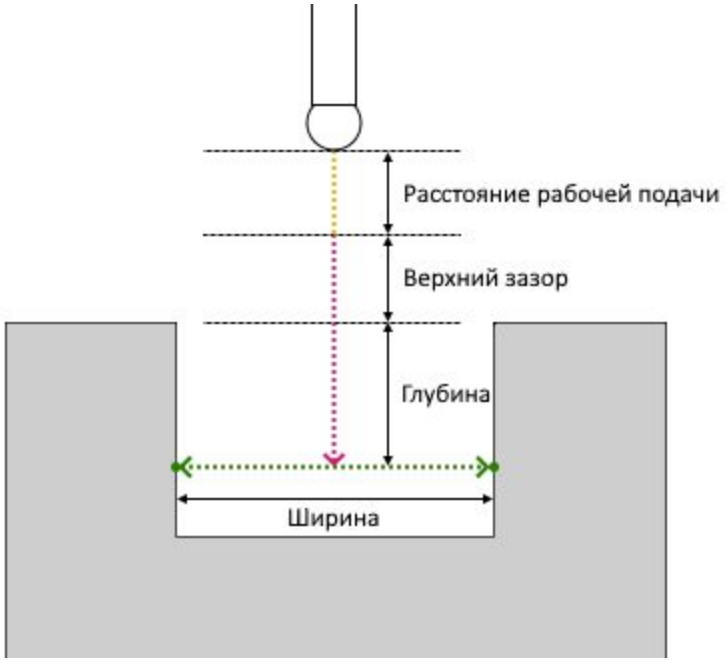
CLDArray	Тип	Описание
CmdPrm.Flt[-213]	Double	Значение третьей исходной точки касания по оси Y, заданное в Job Assignment
CmdPrm.Flt[-214]	Double	Значение третьей исходной точки касания по оси Z, заданное в Job Assignment
CmdPrm.Flt[-215]	Double	Значение третьего исходного целевого вектора по оси X, заданное в Job Assignment
CmdPrm.Flt[-216]	Double	Значение третьего исходного целевого вектора по оси Y, заданное в Job Assignment
CmdPrm.Flt[-217]	Double	Значение третьего исходного целевого вектора по оси Z, заданное в Job Assignment
CmdPrm.Flt[-218]	Double	Значение четвёртой исходной точки касания по оси X, заданное в Job Assignment
CmdPrm.Flt[-219]	Double	Значение четвёртой исходной точки касания по оси Y, заданное в Job Assignment
CmdPrm.Flt[-220]	Double	Значение четвёртой исходной точки касания по оси Z, заданное в Job Assignment
CmdPrm.Flt[-221]	Double	Значение четвёртого исходного целевого вектора по оси X, заданное в Job Assignment

CLDArray	Тип	Описание
CmdPrm.Flt[-222]	Double	Значение четвёртого исходного целевого вектора по оси Y, заданное в Job Assignment
CmdPrm.Flt[-223]	Double	Значение четвёртого исходного целевого вектора по оси Z, заданное в Job Assignment
CmdPrm.Flt[-66]	Integer	Compensate dimensions, возвращаемые значения: 0 - Off, 1 - On

Сопутствующие ссылки

- Измерительный цикл (WProbing)
- Расширенный цикл (EXTCYCLE)
- Модель доступа к CLData

Параметры измерения паза



Измерение паза состоит из следующих шагов:

1. Если параметр `Top clearance` отмечен:
2. Инструмент перемещается на расстояние, заданное в `Feed distance`, и подводится к начальной точке позиции `Top clearance`. Перемещение выполняется на «подаче подвода»;
3. Инструмент перемещается на расстояние, заданное в `Top clearance` плюс значение `Depth`, и подводится к центру точек касания (начальной точке цикла). Перемещение выполняется на «подаче длинных переходов»;
4. Если параметр `Top clearance` не отмечен:
5. Инструмент перемещается на расстояние, заданное в `Feed distance`, и подводится к центру точек касания (начальной точке цикла) согласно глубине. Перемещение выполняется на «подаче подвода»;
6. Инструмент подводится к первой точке касания (перемещаясь на «рабочей подаче») вдоль своего инвертированного `Target vector`, а затем возвращается (перемещаясь на «подаче длинных переходов») к центральной точке. Это действие повторяется для второй точки касания;
7. Если параметр `Top clearance` отмечен: инструмент поднимается на расстояние, равное значению `Top clearance` + значение `Depth`. Перемещение выполняется на «подаче длинных переходов»;
8. Инструмент отводится на расстояние до позиции `Feed distance`. Перемещение выполняется на «подаче возврата».

Параметры

CLDArray	Тип	Описание
CmdPrm.Int[-1]	Integer	Тип измерительного цикла: значение Groove probing = 11
CmdPrm.Int[-2]	Integer	SubCode цикла, заданный в свойстве " SubCode for postprocessor " на вкладке `Job Assignment`
CmdPrm.Flt[-50]	Double	Feed distance, расстояние до начальной позиции цикла
CmdPrm.Flt[-51]	Double	Depth, расстояние от верхней стороны до центра точек касания
CmdPrm.Flt[-53]	Double	Width, расстояние от Touch point 1 до Touch point 2
CmdPrm.Flt[-55]	Double	Top clearance, расстояние от верхней стороны
CmdPrm.Flt[-100]	Double	Значение первой точки касания по оси X
CmdPrm.Flt[-101]	Double	Значение первой точки касания по оси Y
CmdPrm.Flt[-102]	Double	Значение первой точки касания по оси Z
CmdPrm.Flt[-103]	Double	Значение первого целевого вектора по оси X
CmdPrm.Flt[-104]	Double	Значение первого целевого вектора по оси Y
CmdPrm.Flt[-105]	Double	Значение первого целевого вектора по оси Z
CmdPrm.Flt[-106]	Double	Значение второй точки касания по оси X

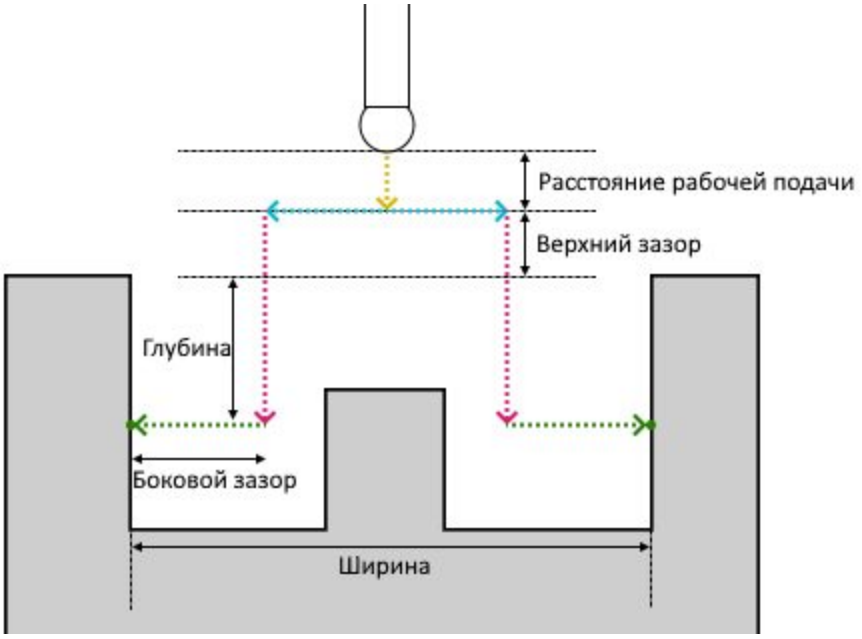
CLDArray	Тип	Описание
CmdPrm.Flt[-107]	Double	Значение второй точки касания по оси Y
CmdPrm.Flt[-108]	Double	Значение второй точки касания по оси Z
CmdPrm.Flt[-109]	Double	Значение второго целевого вектора по оси X
CmdPrm.Flt[-110]	Double	Значение второго целевого вектора по оси Y
CmdPrm.Flt[-111]	Double	Значение второго целевого вектора по оси Z
CmdPrm.Flt[-200]	Double	Значение исходной первой точки касания по оси X, заданное в Job Assignment
CmdPrm.Flt[-201]	Double	Значение исходной первой точки касания по оси Y, заданное в Job Assignment
CmdPrm.Flt[-202]	Double	Значение исходной первой точки касания по оси Z, заданное в Job Assignment
CmdPrm.Flt[-203]	Double	Значение исходного первого целевого вектора по оси X, заданное в Job Assignment
CmdPrm.Flt[-204]	Double	Значение исходного первого целевого вектора по оси Y, заданное в Job Assignment
CmdPrm.Flt[-205]	Double	Значение исходного первого целевого вектора по оси Z, заданное в Job Assignment
CmdPrm.Flt[-206]	Double	Значение исходной второй точки касания по оси X,

CLDArray	Тип	Описание
		заданное в Job Assignment
CmdPrm.Flt[-207]	Double	Значение исходной второй точки касания по оси Y, заданное в Job Assignment
CmdPrm.Flt[-208]	Double	Значение исходной второй точки касания по оси Z, заданное в Job Assignment
CmdPrm.Flt[-209]	Double	Значение исходного второго целевого вектора по оси X, заданное в Job Assignment
CmdPrm.Flt[-210]	Double	Значение исходного второго целевого вектора по оси Y, заданное в Job Assignment
CmdPrm.Flt[-211]	Double	Значение исходного второго целевого вектора по оси Z, заданное в Job Assignment

Сопутствующие ссылки

- Измерительный цикл (WProbing)
- Расширенный цикл (EXTCYCLE)
- Модель доступа к CLData

Параметры защищённого измерения паза



Защищённое измерение паза состоит из следующих шагов:

1. Инструмент перемещается на расстояние, заданное в `Feed distance`, и подводится к начальной позиции цикла. Перемещение выполняется на «подаче подвода»;
2. Инструмент подводится к точке, удалённой от первой точки касания вдоль её `Target vector` на расстояние `Side clearance`, но поднятой на значение `Top clearance` + значение `Depth`. Перемещение выполняется на «подаче длинных переходов»;
3. Инструмент опускается на расстояние, равное значению `Top clearance` + значение `Depth`. Перемещение выполняется на «подаче длинных переходов»;
4. Инструмент подводится к первой точке касания (перемещаясь на «рабочей подаче»), а затем возвращается (перемещаясь на «подаче длинных переходов») в предыдущее положение;
5. Инструмент поднимается на расстояние, равное значению `Top clearance` + значение `Depth`. Перемещение выполняется на «подаче длинных переходов»;
6. Инструмент возвращается в начальную позицию;
7. Шаги 2-6 повторяются для второй точки касания;
8. Инструмент поднимается на расстояние, равное `Feed distance`.

Параметры

CLDArray	Тип	Описание
CmdPrm.Int[-1]	Integer	Тип измерительного цикла: значение Groove probing protected = 12

CLDArray	Тип	Описание
CmdPrm.Int[-2]	Integer	SubCode цикла, заданный в свойстве " SubCode for postprocessor " на вкладке `Job Assignment`
CmdPrm.Flt[-50]	Double	Feed distance, расстояние до начальной позиции цикла
CmdPrm.Flt[-51]	Double	Depth, расстояние от верхней стороны до центра точек касания
CmdPrm.Flt[-53]	Double	Width, расстояние от Touch point 1 до Touch point 2
CmdPrm.Flt[-55]	Double	Top clearance, расстояние от верхней стороны
CmdPrm.Flt[-56]	Double	Side clearance, расстояние подвода к точке касания 1 / точке касания 2
CmdPrm.Flt[-100]	Double	Значение первой точки касания по оси X
CmdPrm.Flt[-101]	Double	Значение первой точки касания по оси Y
CmdPrm.Flt[-102]	Double	Значение первой точки касания по оси Z
CmdPrm.Flt[-103]	Double	Значение первого целевого вектора по оси X
CmdPrm.Flt[-104]	Double	Значение первого целевого вектора по оси Y
CmdPrm.Flt[-105]	Double	Значение первого целевого вектора по оси Z

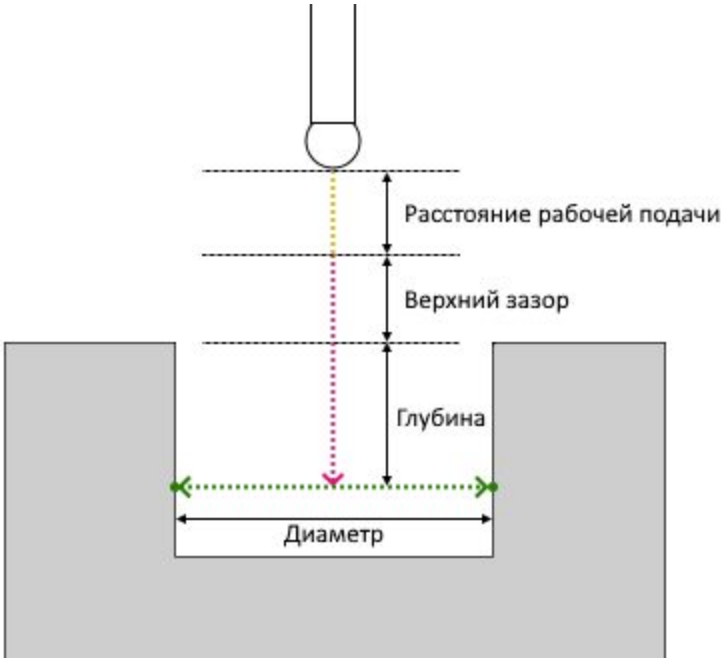
CLDArray	Тип	Описание
CmdPrm.Flt[-106]	Double	Значение второй точки касания по оси X
CmdPrm.Flt[-107]	Double	Значение второй точки касания по оси Y
CmdPrm.Flt[-108]	Double	Значение второй точки касания по оси Z
CmdPrm.Flt[-109]	Double	Значение второго целевого вектора по оси X
CmdPrm.Flt[-110]	Double	Значение второго целевого вектора по оси Y
CmdPrm.Flt[-111]	Double	Значение второго целевого вектора по оси Z
CmdPrm.Flt[-200]	Double	Значение исходной первой точки касания по оси X, заданное в Job Assignment
CmdPrm.Flt[-201]	Double	Значение исходной первой точки касания по оси Y, заданное в Job Assignment
CmdPrm.Flt[-202]	Double	Значение исходной первой точки касания по оси Z, заданное в Job Assignment
CmdPrm.Flt[-203]	Double	Значение исходного первого целевого вектора по оси X, заданное в Job Assignment
CmdPrm.Flt[-204]	Double	Значение исходного первого целевого вектора по оси Y, заданное в Job Assignment
CmdPrm.Flt[-205]	Double	Значение исходного первого целевого вектора по оси Z, заданное в Job Assignment

CLDArray	Тип	Описание
CmdPrm.Flt[-206]	Double	Значение исходной второй точки касания по оси X, заданное в Job Assignment
CmdPrm.Flt[-207]	Double	Значение исходной второй точки касания по оси Y, заданное в Job Assignment
CmdPrm.Flt[-208]	Double	Значение исходной второй точки касания по оси Z, заданное в Job Assignment
CmdPrm.Flt[-209]	Double	Значение исходного второго целевого вектора по оси X, заданное в Job Assignment
CmdPrm.Flt[-210]	Double	Значение исходного второго целевого вектора по оси Y, заданное в Job Assignment
CmdPrm.Flt[-211]	Double	Значение исходного второго целевого вектора по оси Z, заданное в Job Assignment
CmdPrm.Flt[-66]	Integer	Compensate dimensions, возвращаемые значения: 0 - Off, 1 - On

Сопутствующие ссылки

- Измерительный цикл (WProbing)
- Расширенный цикл (EXTCYCLE)
- Модель доступа к CLData

Параметры измерения отверстия



Измерение отверстия состоит из следующих шагов:

1. Если параметр `Top clearance` отмечен:
2. Инструмент перемещается на расстояние, заданное в `Feed distance`, и подводится к начальной точке позиции `Top clearance`. Перемещение выполняется на «подаче подвода»;
3. Инструмент перемещается на расстояние, заданное в `Top clearance` плюс значение `Depth`, и подводится к центру отверстия (начальной точке цикла). Перемещение выполняется на «подаче длинных переходов»;
4. Если параметр `Top clearance` не отмечен:
5. Инструмент перемещается на расстояние, заданное в `Feed distance`, и подводится к центру отверстия (начальной точке цикла) согласно глубине. Перемещение выполняется на «подаче подвода»;
6. Инструмент подводится к первой точке касания (перемещаясь на «рабочей подаче») вдоль своего инвертированного `Target vector`, а затем возвращается (перемещаясь на «подаче длинных переходов») к центральной точке. Это действие повторяется столько раз, сколько точек касания;
7. Если параметр `Top clearance` отмечен: инструмент поднимается на расстояние, равное значению `Top clearance` + значение `Depth`. Перемещение на «подаче длинных переходов»;
8. Инструмент отводится на расстояние до позиции `Feed distance`. Перемещение на «подаче возврата».

Параметры

CLDArray	Тип	Описание
CmdPrm.Int[-1]	Integer	Тип измерительного цикла: значение Hole probing = 3
CmdPrm.Int[-2]	Integer	SubCode цикла, заданный в свойстве " SubCode for postprocessor " на вкладке `Job Assignment`
CmdPrm.Flt[-50]	Double	Feed distance, расстояние до начальной позиции цикла
CmdPrm.Flt[-51]	Double	Depth, расстояние от верхней стороны до центра точек касания
CmdPrm.Flt[-52]	Double	Диаметр измеряемого отверстия
CmdPrm.Flt[-55]	Double	Top clearance, расстояние от верхней стороны
CmdPrm.Int[-59]	Integer	Вариант цикла. 0 - прямоугольный режим, 1 - угловой режим
CmdPrm.Flt[-60]	Double	Start angle (для варианта цикла = 1). Начальный угол точек касания
CmdPrm.Flt[-61]	Double	Angular step (для варианта цикла = 1). Радиальное расстояние между точками касания
CmdPrm.Int[-62]	Integer	Step count (для варианта цикла = 1). Число точек касания
CmdPrm.Flt[-100]	Double	Значение первой точки касания по оси X

CLDArray	Тип	Описание
CmdPrm.Flt[-101]	Double	Значение первой точки касания по оси Y
CmdPrm.Flt[-102]	Double	Значение первой точки касания по оси Z
CmdPrm.Flt[-103]	Double	Значение первого целевого вектора по оси X
CmdPrm.Flt[-104]	Double	Значение первого целевого вектора по оси Y
CmdPrm.Flt[-105]	Double	Значение первого целевого вектора по оси Z
CmdPrm.Flt[-100-((N-1)*6)]	Double	Значение прочей точки касания по оси X. N - номер точки касания
CmdPrm.Flt[-101-((N-1)*6)]	Double	Значение прочей точки касания по оси Y. N - номер точки касания
CmdPrm.Flt[-102-((N-1)*6)]	Double	Значение прочей точки касания по оси Z. N - номер точки касания
CmdPrm.Flt[-103-((N-1)*6)]	Double	Значение прочего целевого вектора по оси X. N - номер точки касания
CmdPrm.Flt[-104-((N-1)*6)]	Double	Значение прочего целевого вектора по оси Y. N - номер точки касания
CmdPrm.Flt[-105-((N-1)*6)]	Double	Значение прочего целевого вектора по оси Z. N - номер точки касания
CmdPrm.Flt[-200]	Double	Значение исходной первой точки касания по оси X,

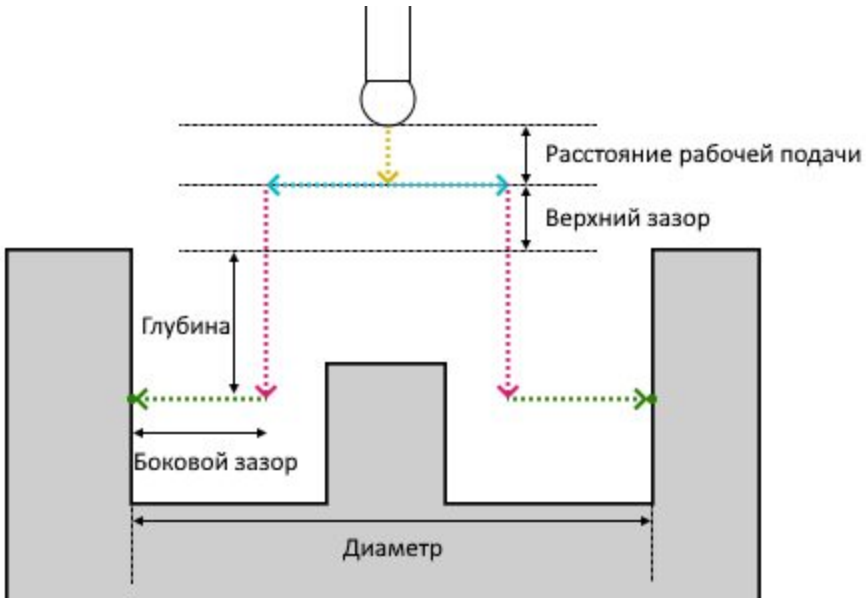
CLDArray	Тип	Описание
		заданное в Job Assignment
CmdPrm.Flt[-201]	Double	Значение исходной первой точки касания по оси Y, заданное в Job Assignment
CmdPrm.Flt[-202]	Double	Значение исходной первой точки касания по оси Z, заданное в Job Assignment
CmdPrm.Flt[-203]	Double	Значение исходного первого целевого вектора по оси X, заданное в Job Assignment
CmdPrm.Flt[-204]	Double	Значение исходного первого целевого вектора по оси Y, заданное в Job Assignment
CmdPrm.Flt[-205]	Double	Значение исходного первого целевого вектора по оси Z, заданное в Job Assignment
CmdPrm.Flt[-200-((N-1)*6)]	Double	Значение прочей точки касания от исходной точки по оси X. N - номер точки касания
CmdPrm.Flt[-201-((N-1)*6)]	Double	Значение прочей точки касания от исходной точки по оси Y. N - номер точки касания
CmdPrm.Flt[-202-((N-1)*6)]	Double	Значение прочей точки касания от исходной точки по оси Z. N - номер точки касания
CmdPrm.Flt[-203-((N-1)*6)]	Double	Значение прочего целевого вектора от исходного вектора по оси X. N - номер точки касания

CLDArray	Тип	Описание
CmdPrm.Flt[-204-((N-1)*6)]	Double	Значение прочего целевого вектора от исходного вектора по оси Y. N - номер точки касания
CmdPrm.Flt[-205-((N-1)*6)]	Double	Значение прочего целевого вектора от исходного вектора по оси Z. N - номер точки касания

Сопутствующие ссылки

- Измерительный цикл (WProbing)
- Расширенный цикл (EXTCYCLE)
- Модель доступа к CLData

Параметры защищённого измерения отверстия



Защищённое измерение отверстия состоит из следующих шагов:

1. Инструмент перемещается на расстояние, заданное в `Feed distance`, и подводится к начальной позиции цикла. Перемещение выполняется на «подаче подвода»;
2. Инструмент подводится к точке, удалённой от первой точки касания вдоль её `Target vector` на расстояние `Side clearance`, но поднятой на значение `Top clearance` + значение `Depth`. Перемещение выполняется на «подаче длинных переходов»;
3. Инструмент опускается на расстояние, равное значению `Top clearance` + значение `Depth`. Перемещение выполняется на «подаче длинных переходов»;
4. Инструмент подводится к первой точке касания (перемещаясь на «рабочей подаче»), а затем возвращается (перемещаясь на «подаче длинных переходов») в предыдущее положение;
5. Инструмент поднимается на расстояние, равное значению `Top clearance` + значение `Depth`. Перемещение выполняется на «подаче длинных переходов»;
6. Инструмент возвращается в начальную позицию;
7. Шаги 2-6 повторяются столько раз, сколько точек касания в цикле;
8. Инструмент поднимается на расстояние, равное `Feed distance`.

Параметры

CLDArray	Тип	Описание
CmdPrm.Int[-1]	Integer	Тип измерительного цикла: значение Hole protected probing = 4
CmdPrm.Int[-2]	Integer	SubCode цикла, заданный в свойстве " SubCode for postprocessor " на вкладке `Job Assignment`
CmdPrm.Flt[-50]	Double	Feed distance, расстояние до начальной позиции цикла
CmdPrm.Flt[-51]	Double	Depth, расстояние от верхней стороны до центра точек касания
CmdPrm.Flt[-52]	Double	Диаметр измеряемого отверстия
CmdPrm.Flt[-55]	Double	Top clearance, расстояние от верхней стороны
CmdPrm.Flt[-56]	Double	Side clearance, расстояние подвода к точке касания
CmdPrm.Int[-59]	Integer	Вариант цикла. 0 - прямоугольный режим, 1 - угловой режим
CmdPrm.Flt[-60]	Double	Start angle (для варианта цикла = 1). Начальный угол точек касания
CmdPrm.Flt[-61]	Double	Angular step (для варианта цикла = 1). Радиальное расстояние между точками касания
CmdPrm.Int[-62]	Integer	Step count (для варианта цикла = 1). Число точек касания

CLDArray	Тип	Описание
CmdPrm.Flt[-100]	Double	Значение первой точки касания по оси X
CmdPrm.Flt[-101]	Double	Значение первой точки касания по оси Y
CmdPrm.Flt[-102]	Double	Значение первой точки касания по оси Z
CmdPrm.Flt[-103]	Double	Значение первого целевого вектора по оси X
CmdPrm.Flt[-104]	Double	Значение первого целевого вектора по оси Y
CmdPrm.Flt[-105]	Double	Значение первого целевого вектора по оси Z
CmdPrm.Flt[-100-((N-1)*6)]	Double	Значение прочей точки касания по оси X. N - номер точки касания
CmdPrm.Flt[-101-((N-1)*6)]	Double	Значение прочей точки касания по оси Y. N - номер точки касания
CmdPrm.Flt[-102-((N-1)*6)]	Double	Значение прочей точки касания по оси Z. N - номер точки касания
CmdPrm.Flt[-103-((N-1)*6)]	Double	Значение прочего целевого вектора по оси X. N - номер точки касания
CmdPrm.Flt[-104-((N-1)*6)]	Double	Значение прочего целевого вектора по оси Y. N - номер точки касания
CmdPrm.Flt[-105-((N-1)*6)]	Double	Значение прочего целевого вектора по оси Z. N - номер точки касания

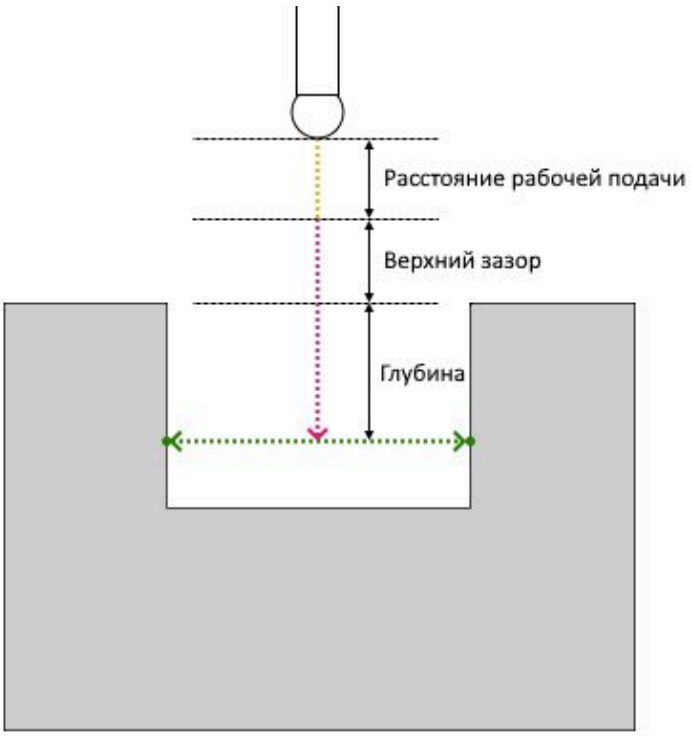
CLDArray	Тип	Описание
CmdPrm.Flt[-200]	Double	Значение исходной первой точки касания по оси X, заданное в Job Assignment
CmdPrm.Flt[-201]	Double	Значение исходной первой точки касания по оси Y, заданное в Job Assignment
CmdPrm.Flt[-202]	Double	Значение исходной первой точки касания по оси Z, заданное в Job Assignment
CmdPrm.Flt[-203]	Double	Значение исходного первого целевого вектора по оси X, заданное в Job Assignment
CmdPrm.Flt[-204]	Double	Значение исходного первого целевого вектора по оси Y, заданное в Job Assignment
CmdPrm.Flt[-205]	Double	Значение исходного первого целевого вектора по оси Z, заданное в Job Assignment
CmdPrm.Flt[-200-((N-1)*6)]	Double	Значение прочей точки касания от исходной точки по оси X. N - номер точки касания
CmdPrm.Flt[-201-((N-1)*6)]	Double	Значение прочей точки касания от исходной точки по оси Y. N - номер точки касания
CmdPrm.Flt[-202-((N-1)*6)]	Double	Значение прочей точки касания от исходной точки по оси Z. N - номер точки касания
CmdPrm.Flt[-203-((N-1)*6)]	Double	Значение прочего целевого вектора от исходного

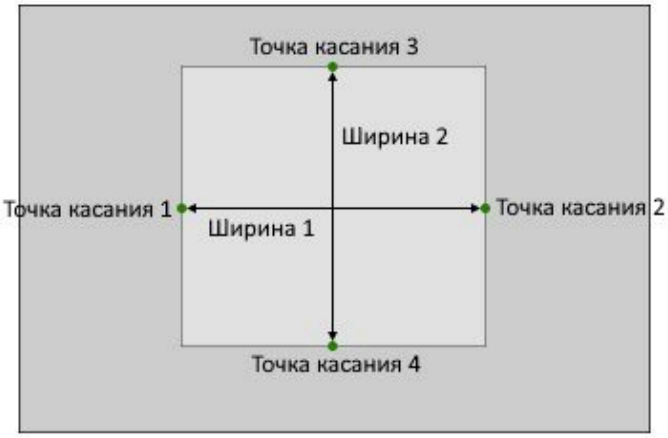
CLDArray	Тип	Описание
		вектора по оси X. N - номер точки касания
CmdPrm.Flt[-204-((N-1)*6)]	Double	Значение прочего целевого вектора от исходного вектора по оси Y. N - номер точки касания
CmdPrm.Flt[-205-((N-1)*6)]	Double	Значение прочего целевого вектора от исходного вектора по оси Z. N - номер точки касания
CmdPrm.Flt[-66]	Integer	Compensate dimensions, возвращаемые значения: 0 - Off, 1 - On

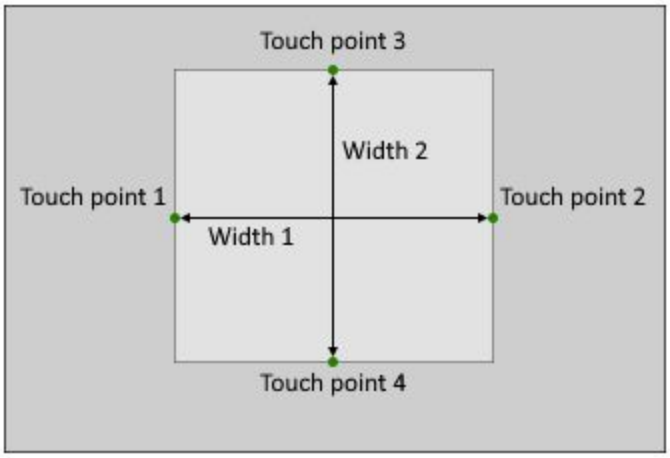
Сопутствующие ссылки

- Измерительный цикл (WProbing)
- Расширенный цикл (EXTCYCLE)
- Модель доступа к CLData

Параметры измерения внутреннего прямоугольника







Измерение внутреннего прямоугольника состоит из следующих шагов:

1. Если параметр `Top clearance` отмечен:
2. Инструмент перемещается на расстояние, заданное в `Feed distance`, и подводится к начальной точке позиции `Top clearance`. Перемещение выполняется на «подаче подвода»;
3. Инструмент перемещается на расстояние, заданное в `Top clearance` плюс значение `Depth`, и подводится к центру сторон (начальной точке цикла). Перемещение выполняется на «подаче длинных переходов»;
4. Если параметр `Top clearance` не отмечен:

5. Инструмент перемещается на расстояние, заданное в `Feed distance`, и подводится к центру сторон (начальной точке цикла) согласно глубине. Перемещение на «подаче подвода»;
6. Инструмент подводится к первой точке касания (перемещаясь на «рабочей подаче») вдоль своего инвертированного `Target vector`, а затем возвращается (перемещаясь на «подаче длинных переходов») к центральной точке. Это действие повторяется 3 раза для каждой точки касания;
7. Если параметр `Top clearance` отмечен: инструмент поднимается на расстояние, равное значению `Top clearance` + значение `Depth`. Перемещение на «подаче длинных переходов»;
8. Инструмент отводится на расстояние до позиции `Feed distance`. Перемещение на «подаче возврата».

Параметры

CLDArray	Тип	Описание
CmdPrm.Int[-1]	Integer	Тип измерительного цикла: значение Internal rectangle probing = 6
CmdPrm.Int[-2]	Integer	SubCode цикла, заданный в свойстве " SubCode for postprocessor " на вкладке `Job Assignment`
CmdPrm.Flt[-50]	Double	Feed distance, расстояние до начальной позиции цикла
CmdPrm.Flt[-51]	Double	Depth, расстояние от верхней стороны до центра точек касания
CmdPrm.Flt[-53]	Double	Width 1, расстояние от Touch point 1 до Touch point 2
CmdPrm.Flt[-54]	Double	Width 2, расстояние от Touch point 3 до Touch point 4
CmdPrm.Flt[-55]	Double	Top clearance, расстояние от верхней стороны
CmdPrm.Flt[-100]	Double	Значение первой точки касания по оси X

CLDArray	Тип	Описание
CmdPrm.Flt[-101]	Double	Значение первой точки касания по оси Y
CmdPrm.Flt[-102]	Double	Значение первой точки касания по оси Z
CmdPrm.Flt[-103]	Double	Значение первого целевого вектора по оси X
CmdPrm.Flt[-104]	Double	Значение первого целевого вектора по оси Y
CmdPrm.Flt[-105]	Double	Значение первого целевого вектора по оси Z
CmdPrm.Flt[-106]	Double	Значение второй точки касания по оси X
CmdPrm.Flt[-107]	Double	Значение второй точки касания по оси Y
CmdPrm.Flt[-108]	Double	Значение второй точки касания по оси Z
CmdPrm.Flt[-109]	Double	Значение второго целевого вектора по оси X
CmdPrm.Flt[-110]	Double	Значение второго целевого вектора по оси Y
CmdPrm.Flt[-111]	Double	Значение второго целевого вектора по оси Z
CmdPrm.Flt[-112]	Double	Значение третьей точки касания по оси X
CmdPrm.Flt[-113]	Double	Значение третьей точки касания по оси Y
CmdPrm.Flt[-114]	Double	Значение третьей точки касания по оси Z

CLDArray	Тип	Описание
CmdPrm.Flt[-115]	Double	Значение третьего целевого вектора по оси X
CmdPrm.Flt[-116]	Double	Значение третьего целевого вектора по оси Y
CmdPrm.Flt[-117]	Double	Значение третьего целевого вектора по оси Z
CmdPrm.Flt[-118]	Double	Значение четвёртой точки касания по оси X
CmdPrm.Flt[-119]	Double	Значение четвёртой точки касания по оси Y
CmdPrm.Flt[-120]	Double	Значение четвёртой точки касания по оси Z
CmdPrm.Flt[-121]	Double	Значение четвёртого целевого вектора по оси X
CmdPrm.Flt[-122]	Double	Значение четвёртого целевого вектора по оси Y
CmdPrm.Flt[-123]	Double	Значение четвёртого целевого вектора по оси Z
CmdPrm.Flt[-200]	Double	Значение первой исходной точки касания по оси X, заданное в Job Assignment
CmdPrm.Flt[-201]	Double	Значение первой исходной точки касания по оси Y, заданное в Job Assignment
CmdPrm.Flt[-202]	Double	Значение первой исходной точки касания по оси Z, заданное в Job Assignment

CLDArray	Тип	Описание
CmdPrm.Flt[-203]	Double	Значение первого исходного целевого вектора по оси X, заданное в Job Assignment
CmdPrm.Flt[-204]	Double	Значение первого исходного целевого вектора по оси Y, заданное в Job Assignment
CmdPrm.Flt[-205]	Double	Значение первого исходного целевого вектора по оси Z, заданное в Job Assignment
CmdPrm.Flt[-206]	Double	Значение второй исходной точки касания по оси X, заданное в Job Assignment
CmdPrm.Flt[-207]	Double	Значение второй исходной точки касания по оси Y, заданное в Job Assignment
CmdPrm.Flt[-208]	Double	Значение второй исходной точки касания по оси Z, заданное в Job Assignment
CmdPrm.Flt[-209]	Double	Значение второго исходного целевого вектора по оси X, заданное в Job Assignment
CmdPrm.Flt[-210]	Double	Значение второго исходного целевого вектора по оси Y, заданное в Job Assignment
CmdPrm.Flt[-211]	Double	Значение второго исходного целевого вектора по оси Z, заданное в Job Assignment
CmdPrm.Flt[-212]	Double	Значение третьей исходной точки касания по оси X, заданное в Job Assignment

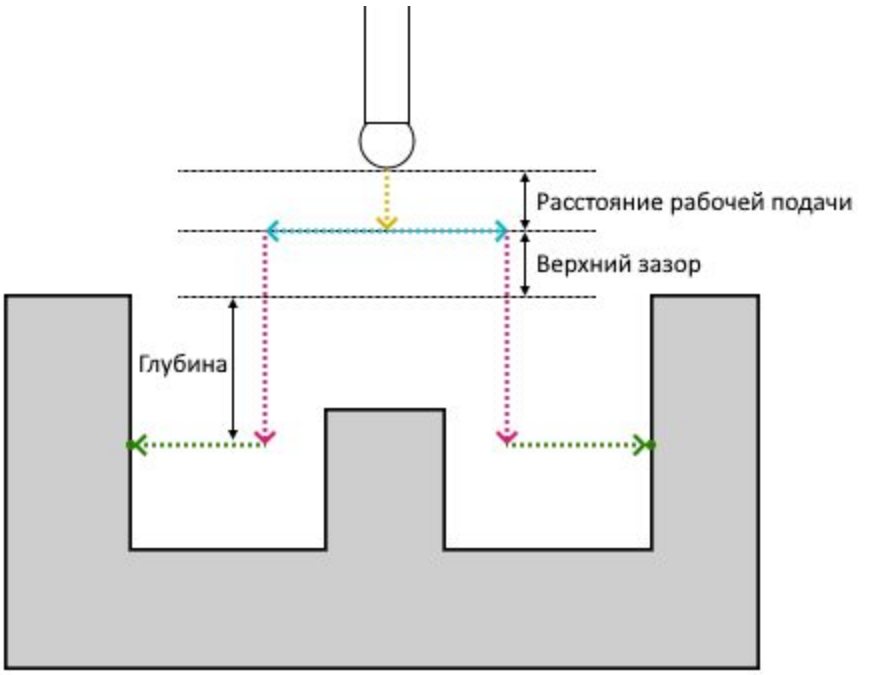
CLDArray	Тип	Описание
CmdPrm.Flt[-213]	Double	Значение третьей исходной точки касания по оси Y, заданное в Job Assignment
CmdPrm.Flt[-214]	Double	Значение третьей исходной точки касания по оси Z, заданное в Job Assignment
CmdPrm.Flt[-215]	Double	Значение третьего исходного целевого вектора по оси X, заданное в Job Assignment
CmdPrm.Flt[-216]	Double	Значение третьего исходного целевого вектора по оси Y, заданное в Job Assignment
CmdPrm.Flt[-217]	Double	Значение третьего исходного целевого вектора по оси Z, заданное в Job Assignment
CmdPrm.Flt[-218]	Double	Значение четвёртой исходной точки касания по оси X, заданное в Job Assignment
CmdPrm.Flt[-219]	Double	Значение четвёртой исходной точки касания по оси Y, заданное в Job Assignment
CmdPrm.Flt[-220]	Double	Значение четвёртой исходной точки касания по оси Z, заданное в Job Assignment
CmdPrm.Flt[-221]	Double	Значение четвёртого исходного целевого вектора по оси X, заданное в Job Assignment

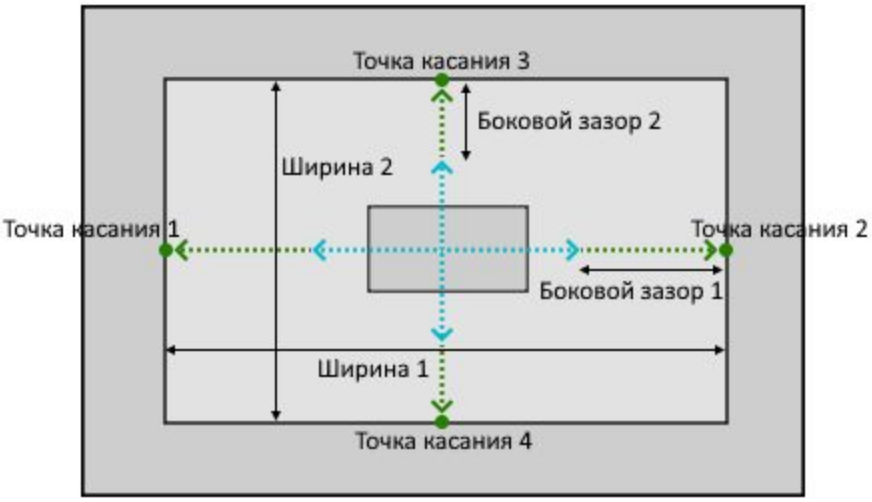
CLDArray	Тип	Описание
CmdPrm.Flt[-222]	Double	Значение четвёртого исходного целевого вектора по оси Y, заданное в Job Assignment
CmdPrm.Flt[-223]	Double	Значение четвёртого исходного целевого вектора по оси Z, заданное в Job Assignment

Сопутствующие ссылки

- Измерительный цикл (WProbing)
- Расширенный цикл (EXTCYCLE)
- Модель доступа к CLData

Параметры защищённого измерения внутреннего прямоугольника





Защищённое измерение внутреннего прямоугольника состоит из следующих шагов:

1. Инструмент перемещается на расстояние, заданное в `Feed distance`, и подводится к начальной позиции цикла. Перемещение выполняется на «подаче подвода»;
2. Инструмент подводится к точке, удалённой от первой точки касания вдоль её `Target vector` на расстояние `Side clearance 1`, но поднятой на значение `Top clearance` + значение `Depth`. Перемещение выполняется на «подаче длинных переходов»;
3. Инструмент опускается на расстояние, равное значению `Top clearance` + значение `Depth`. Перемещение на «подаче длинных переходов»;
4. Инструмент подводится к первой точке касания (перемещаясь на «рабочей подаче»), а затем возвращается (перемещаясь на «подаче длинных переходов») в предыдущее

положение;

5. Инструмент поднимается на расстояние, равное значению `Top clearance` + значение `Depth`. Перемещение выполняется на «подаче длинных переходов»;
6. Инструмент возвращается в начальную позицию;
7. Шаги 2-6 повторяются 3 раза для каждой точки касания;
8. Инструмент поднимается на расстояние, равное `Feed distance`.

Параметры

CLDArray	Тип	Описание
CmdPrm.Int[-1]	Integer	Тип измерительного цикла: значение Internal rectangle probing protected = 7
CmdPrm.Int[-2]	Integer	SubCode цикла, заданный в свойстве " SubCode for postprocessor " на вкладке `Job Assignment`
CmdPrm.Flt[-50]	Double	Feed distance, расстояние до начальной позиции цикла
CmdPrm.Flt[-51]	Double	Depth, расстояние от верхней стороны до центра точек касания
CmdPrm.Flt[-53]	Double	Width 1, расстояние от Touch point 1 до Touch point 2
CmdPrm.Flt[-54]	Double	Width 2, расстояние от Touch point 3 до Touch point 4
CmdPrm.Flt[-55]	Double	Top clearance, расстояние от верхней стороны
CmdPrm.Flt[-56]	Double	Side clearance 1, расстояние подвода к точке касания 1 / точке касания 2
CmdPrm.Flt[-57]	Double	Side clearance 2, расстояние подвода к точке касания 3 / точке касания 4

CLDArray	Тип	Описание
CmdPrm.Flt[-100]	Double	Значение первой точки касания по оси X
CmdPrm.Flt[-101]	Double	Значение первой точки касания по оси Y
CmdPrm.Flt[-102]	Double	Значение первой точки касания по оси Z
CmdPrm.Flt[-103]	Double	Значение первого целевого вектора по оси X
CmdPrm.Flt[-104]	Double	Значение первого целевого вектора по оси Y
CmdPrm.Flt[-105]	Double	Значение первого целевого вектора по оси Z
CmdPrm.Flt[-106]	Double	Значение второй точки касания по оси X
CmdPrm.Flt[-107]	Double	Значение второй точки касания по оси Y
CmdPrm.Flt[-108]	Double	Значение второй точки касания по оси Z
CmdPrm.Flt[-109]	Double	Значение второго целевого вектора по оси X
CmdPrm.Flt[-110]	Double	Значение второго целевого вектора по оси Y
CmdPrm.Flt[-111]	Double	Значение второго целевого вектора по оси Z
CmdPrm.Flt[-112]	Double	Значение третьей точки касания по оси X
CmdPrm.Flt[-113]	Double	Значение третьей точки касания по оси Y

CLDArray	Тип	Описание
CmdPrm.Flt[-114]	Double	Значение третьей точки касания по оси Z
CmdPrm.Flt[-115]	Double	Значение третьего целевого вектора по оси X
CmdPrm.Flt[-116]	Double	Значение третьего целевого вектора по оси Y
CmdPrm.Flt[-117]	Double	Значение третьего целевого вектора по оси Z
CmdPrm.Flt[-118]	Double	Значение четвёртой точки касания по оси X
CmdPrm.Flt[-119]	Double	Значение четвёртой точки касания по оси Y
CmdPrm.Flt[-120]	Double	Значение четвёртой точки касания по оси Z
CmdPrm.Flt[-121]	Double	Значение четвёртого целевого вектора по оси X
CmdPrm.Flt[-122]	Double	Значение четвёртого целевого вектора по оси Y
CmdPrm.Flt[-123]	Double	Значение четвёртого целевого вектора по оси Z
CmdPrm.Flt[-200]	Double	Значение первой исходной точки касания по оси X, заданное в Job Assignment
CmdPrm.Flt[-201]	Double	Значение первой исходной точки касания по оси Y, заданное в Job Assignment
CmdPrm.Flt[-202]	Double	Значение первой исходной точки касания по оси Z, заданное в Job Assignment

CLDArray	Тип	Описание
CmdPrm.Flt[-203]	Double	Значение первого исходного целевого вектора по оси X, заданное в Job Assignment
CmdPrm.Flt[-204]	Double	Значение первого исходного целевого вектора по оси Y, заданное в Job Assignment
CmdPrm.Flt[-205]	Double	Значение первого исходного целевого вектора по оси Z, заданное в Job Assignment
CmdPrm.Flt[-206]	Double	Значение второй исходной точки касания по оси X, заданное в Job Assignment
CmdPrm.Flt[-207]	Double	Значение второй исходной точки касания по оси Y, заданное в Job Assignment
CmdPrm.Flt[-208]	Double	Значение второй исходной точки касания по оси Z, заданное в Job Assignment
CmdPrm.Flt[-209]	Double	Значение второго исходного целевого вектора по оси X, заданное в Job Assignment
CmdPrm.Flt[-210]	Double	Значение второго исходного целевого вектора по оси Y, заданное в Job Assignment
CmdPrm.Flt[-211]	Double	Значение второго исходного целевого вектора по оси Z, заданное в Job Assignment
CmdPrm.Flt[-212]	Double	Значение третьей исходной точки касания по оси X, заданное в Job Assignment

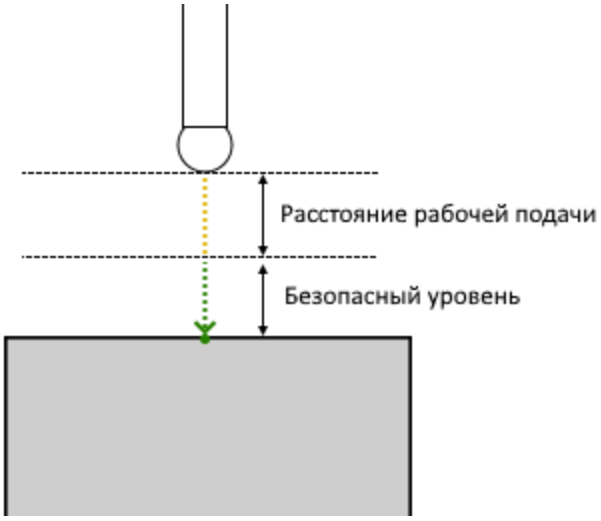
CLDArray	Тип	Описание
CmdPrm.Flt[-213]	Double	Значение третьей исходной точки касания по оси Y, заданное в Job Assignment
CmdPrm.Flt[-214]	Double	Значение третьей исходной точки касания по оси Z, заданное в Job Assignment
CmdPrm.Flt[-215]	Double	Значение третьего исходного целевого вектора по оси X, заданное в Job Assignment
CmdPrm.Flt[-216]	Double	Значение третьего исходного целевого вектора по оси Y, заданное в Job Assignment
CmdPrm.Flt[-217]	Double	Значение третьего исходного целевого вектора по оси Z, заданное в Job Assignment
CmdPrm.Flt[-218]	Double	Значение четвёртой исходной точки касания по оси X, заданное в Job Assignment
CmdPrm.Flt[-219]	Double	Значение четвёртой исходной точки касания по оси Y, заданное в Job Assignment
CmdPrm.Flt[-220]	Double	Значение четвёртой исходной точки касания по оси Z, заданное в Job Assignment
CmdPrm.Flt[-221]	Double	Значение четвёртого исходного целевого вектора по оси X, заданное в Job Assignment

CLDArray	Тип	Описание
CmdPrm.Flt[-222]	Double	Значение четвёртого исходного целевого вектора по оси Y, заданное в Job Assignment
CmdPrm.Flt[-223]	Double	Значение четвёртого исходного целевого вектора по оси Z, заданное в Job Assignment

Сопутствующие ссылки

- Измерительный цикл (WProbing)
- Расширенный цикл (EXTCYCLE)
- Модель доступа к CLData

Параметры измерения поверхности



Измерение поверхности состоит из следующих шагов:

1. Инструмент перемещается на расстояние, заданное в `Feed distance`, и подводится к начальной точке измерительного цикла. Перемещение выполняется на «подаче подвода»;
2. Инструмент перемещается на расстояние, заданное в `Clearance`, и подводится к точке касания поверхности. Перемещение выполняется на «рабочей подаче»;
3. Инструмент возвращается к начальной точке на расстояние, заданное в `Clearance`. Перемещение выполняется на «подаче отвода»;
4. Инструмент возвращается на расстояние, заданное в `Feed distance`. Перемещение выполняется на «подаче возврата».

Параметры

CLDArray	Тип	Описание
CmdPrm.Int[-1]	Integer	Тип измерительного цикла: значение Surface probing = 2
CmdPrm.Int[-2]	Integer	SubCode цикла, заданный в свойстве "SubCode for postprocessor" на вкладке `Job Assignment`
CmdPrm.Flt[-50]	Double	Feed distance, расстояние до начальной позиции цикла
CmdPrm.Flt[-100]	Double	Значение точки касания по оси X

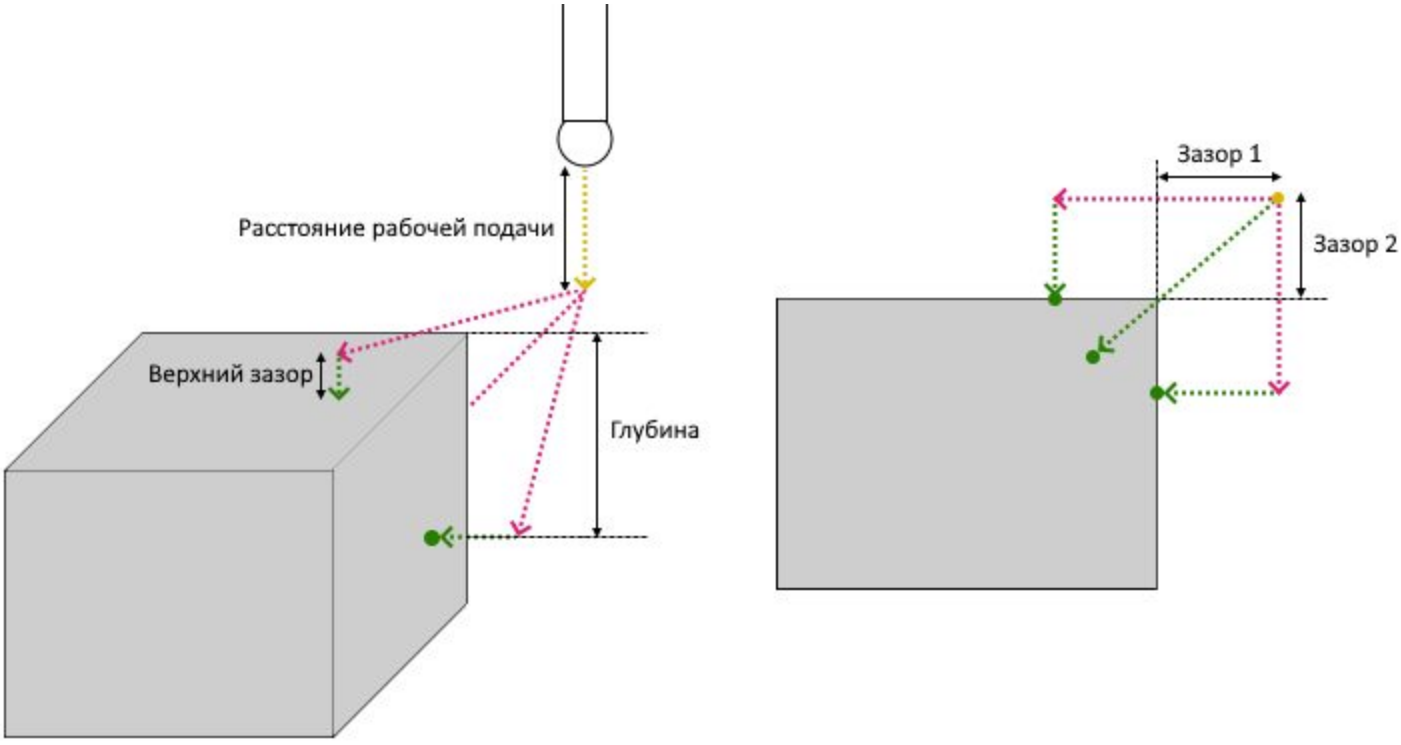
CLDArray	Тип	Описание
CmdPrm.Flt[-101]	Double	Значение точки касания по оси Y
CmdPrm.Flt[-102]	Double	Значение точки касания по оси Z
CmdPrm.Flt[-103]	Double	Значение целевого вектора по оси X
CmdPrm.Flt[-104]	Double	Значение целевого вектора по оси Y
CmdPrm.Flt[-105]	Double	Значение целевого вектора по оси Z
CmdPrm.Flt[-200]	Double	Значение исходной точки касания по оси X, заданное в Job Assignment
CmdPrm.Flt[-201]	Double	Значение исходной точки касания по оси Y, заданное в Job Assignment
CmdPrm.Flt[-202]	Double	Значение исходной точки касания по оси Z, заданное в Job Assignment
CmdPrm.Flt[-203]	Double	Значение исходного целевого вектора по оси X, заданное в Job Assignment
CmdPrm.Flt[-204]	Double	Значение исходного целевого вектора по оси Y, заданное в Job Assignment
CmdPrm.Flt[-205]	Double	Значение исходного целевого вектора по оси Z, заданное в Job Assignment
CmdPrm.Flt[-56]	Double	Clearance, расстояние подвода к точке касания

CLDArray	Тип	Описание
CmdPrm.Flt[-66]	Integer	Compensate dimensions, возвращаемые значения: 0 - Off, 1 - On

Сопутствующие ссылки

- Измерительный цикл (WProbing)
- Расширенный цикл (EXTCYCLE)
- Модель доступа к CLData

Параметры измерения наружного угла по трём стенкам



Измерение наружного угла по трём стенкам состоит из следующих шагов:

1. Инструмент перемещается на расстояние, заданное в `Feed distance`, и подводится к начальной точке измерительного цикла — точке пересечения точки касания на первой стенке и точки касания на второй стенке на расстоянии, заданном в `Clearance 1`, `Clearance 2` соответственно, вдоль ортогонального направления их векторов, причём точка пересечения поднята на расстояние, заданное в `Top clearance` + `Depth`. Перемещение на «подаче подвода»;
2. Инструмент подводится к точке, удалённой от точки касания на первой стороне вдоль её `Target vector` на расстояние `Clearance 1` (перемещаясь на «подаче длинных переходов»), подводится к точке касания первой стороны (перемещаясь на «рабочей подаче»), а затем возвращается (перемещаясь на «подаче длинных переходов») в предыдущее положение;
3. Инструмент возвращается к начальной точке. Перемещение на «подаче длинных переходов»;
4. Инструмент подводится к точке, удалённой от точки касания на второй стороне вдоль её `Target vector` на расстояние `Clearance 2` (перемещаясь на «подаче длинных переходов»), подводится к точке касания второй стороны (перемещаясь на «рабочей подаче»), а затем возвращается (перемещаясь на «подаче длинных переходов») в предыдущее положение;
5. Инструмент возвращается к начальной точке. Перемещение на «подаче длинных переходов»;

6. Инструмент подводится к точке, удалённой от точки касания на верхней стороне вдоль её `Target vector` на расстояние `Top clearance` (перемещаясь на «подаче длинных переходов»), подводится к точке касания верхней стороны (перемещаясь на «рабочей подаче»), а затем возвращается (перемещаясь на «подаче длинных переходов») в предыдущее положение;
7. Инструмент возвращается к начальной точке. Перемещение на «подаче длинных переходов»;
8. Инструмент поднимается на расстояние, равное `Feed distance`.

Параметры

CLDArray	Тип	Описание
CmdPrm.Int[-1]	Integer	Тип измерительного цикла: значение Triple wall external corner probing = 16
CmdPrm.Int[-2]	Integer	SubCode цикла, заданный в свойстве " SubCode for postprocessor " на вкладке `Job Assignment`
CmdPrm.Flt[-50]	Double	Feed distance, расстояние до начальной позиции цикла
CmdPrm.Flt[-51]	Double	Depth, расстояние от верхней стороны до точки касания первой и второй стенки
CmdPrm.Flt[-56]	Double	Clearance 1, расстояние подвода к точке касания на первой стороне
CmdPrm.Flt[-57]	Double	Clearance 2, расстояние подвода к точке касания на второй стороне
CmdPrm.Flt[-58]	Double	Top clearance, расстояние подвода к точке касания на верхней стороне
CmdPrm.Flt[-100]	Double	Значение точки касания на первой стенке по оси X

CLDArray	Тип	Описание
CmdPrm.Flt[-101]	Double	Значение точки касания на первой стенке по оси Y
CmdPrm.Flt[-102]	Double	Значение точки касания на первой стенке по оси Z
CmdPrm.Flt[-103]	Double	Значение целевого вектора точки касания на первой стенке по оси X
CmdPrm.Flt[-104]	Double	Значение целевого вектора точки касания на первой стенке по оси Y
CmdPrm.Flt[-105]	Double	Значение целевого вектора точки касания на первой стенке по оси Z
CmdPrm.Flt[-106]	Double	Значение точки касания на второй стенке по оси X
CmdPrm.Flt[-107]	Double	Значение точки касания на второй стенке по оси Y
CmdPrm.Flt[-108]	Double	Значение точки касания на второй стенке по оси Z
CmdPrm.Flt[-109]	Double	Значение целевого вектора точки касания на второй стенке по оси X
CmdPrm.Flt[-110]	Double	Значение целевого вектора точки касания на второй стенке по оси Y
CmdPrm.Flt[-111]	Double	Значение целевого вектора точки касания на второй стенке по оси Z
CmdPrm.Flt[-112]	Double	Значение точки касания на верхней стенке по оси X

CLDArray	Тип	Описание
CmdPrm.Flt[-113]	Double	Значение точки касания на верхней стенке по оси Y
CmdPrm.Flt[-114]	Double	Значение точки касания на верхней стенке по оси Z
CmdPrm.Flt[-115]	Double	Значение целевого вектора точки касания на верхней стенке по оси X
CmdPrm.Flt[-116]	Double	Значение целевого вектора точки касания на верхней стенке по оси Y
CmdPrm.Flt[-117]	Double	Значение целевого вектора точки касания на верхней стенке по оси Z
CmdPrm.Flt[-200]	Double	Значение исходной точки касания на первой стенке по оси X, заданное в Job Assignment
CmdPrm.Flt[-201]	Double	Значение исходной точки касания на первой стенке по оси Y, заданное в Job Assignment
CmdPrm.Flt[-202]	Double	Значение исходной точки касания на первой стенке по оси Z, заданное в Job Assignment
CmdPrm.Flt[-203]	Double	Значение исходного целевого вектора точки касания на первой стенке по оси X, заданное в Job Assignment
CmdPrm.Flt[-204]	Double	Значение исходного целевого вектора точки

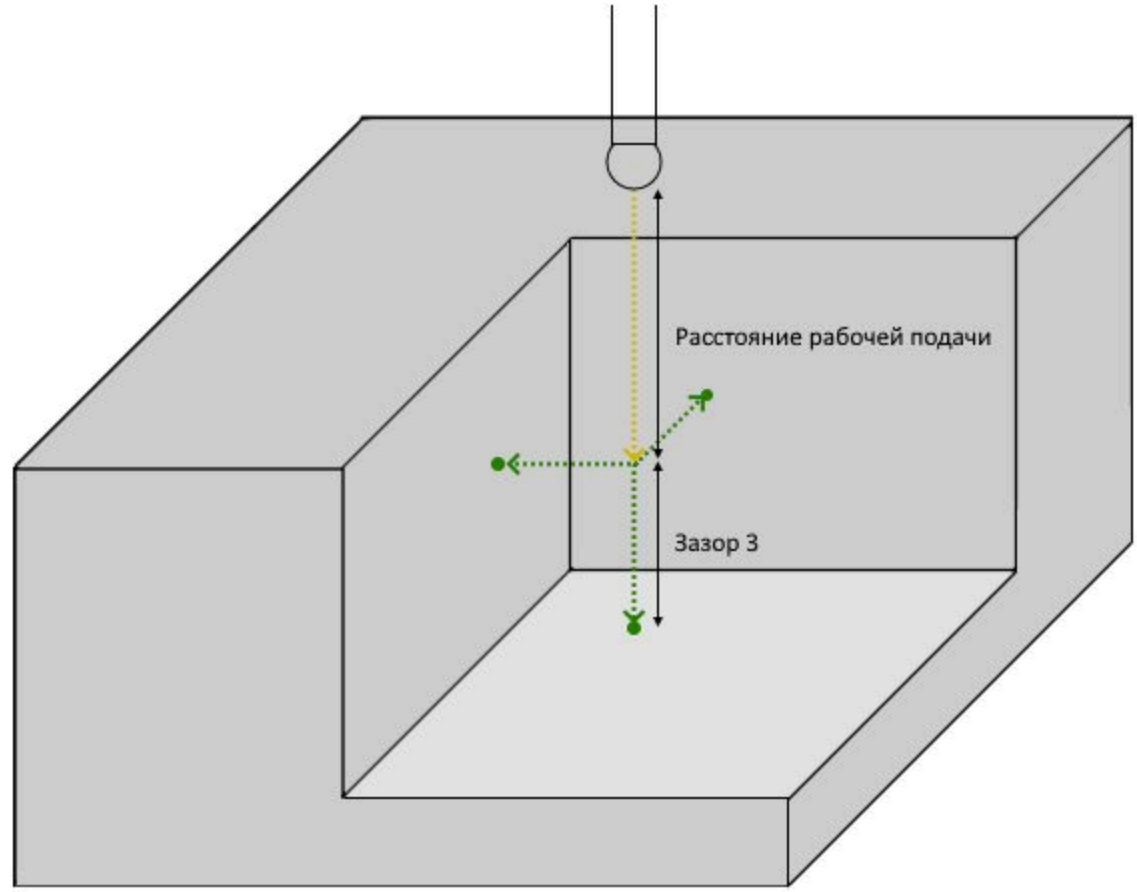
CLDArray	Тип	Описание
		касания на первой стенке по оси Y, заданное в Job Assignment
CmdPrm.Flt[-205]	Double	Значение исходного целевого вектора точки касания на первой стенке по оси Z, заданное в Job Assignment
CmdPrm.Flt[-206]	Double	Значение исходной точки касания на второй стенке по оси X, заданное в Job Assignment
CmdPrm.Flt[-207]	Double	Значение исходной точки касания на второй стенке по оси Y, заданное в Job Assignment
CmdPrm.Flt[-208]	Double	Значение исходной точки касания на второй стенке по оси Z, заданное в Job Assignment
CmdPrm.Flt[-209]	Double	Значение исходного целевого вектора точки касания на второй стенке по оси X, заданное в Job Assignment
CmdPrm.Flt[-210]	Double	Значение исходного целевого вектора точки касания на второй стенке по оси Y, заданное в Job Assignment
CmdPrm.Flt[-211]	Double	Значение исходного целевого вектора точки касания на второй стенке по

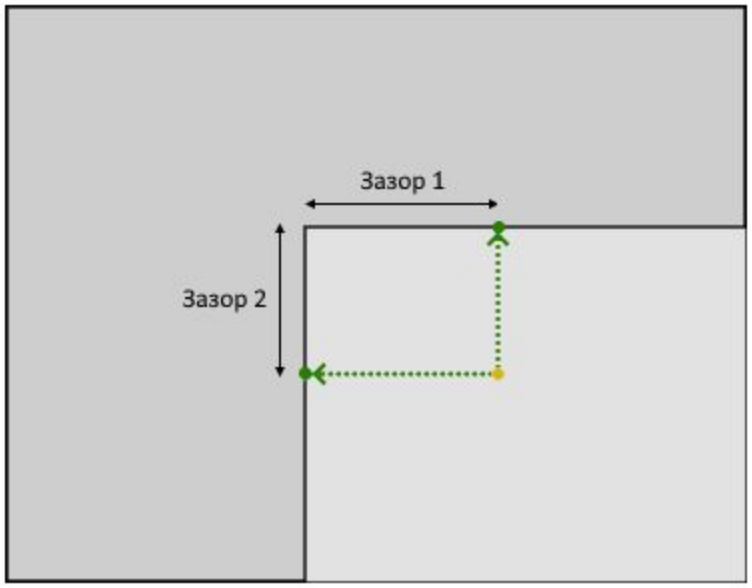
CLDArray	Тип	Описание
		оси Z, заданное в Job Assignment
CmdPrm.Flt[-212]	Double	Значение исходной точки касания на верхней стенке по оси X, заданное в Job Assignment
CmdPrm.Flt[-213]	Double	Значение исходной точки касания на верхней стенке по оси Y, заданное в Job Assignment
CmdPrm.Flt[-214]	Double	Значение исходной точки касания на верхней стенке по оси Z, заданное в Job Assignment
CmdPrm.Flt[-215]	Double	Значение исходного целевого вектора точки касания на верхней стенке по оси X, заданное в Job Assignment
CmdPrm.Flt[-216]	Double	Значение исходного целевого вектора точки касания на верхней стенке по оси Y, заданное в Job Assignment
CmdPrm.Flt[-217]	Double	Значение исходного целевого вектора точки касания на верхней стенке по оси Z, заданное в Job Assignment

Сопутствующие ссылки

- Измерительный цикл (WProbing)
- Расширенный цикл (EXTCYCLE)
- Модель доступа к CLData

Параметры измерения внутреннего угла по трём стенкам





Измерение внутреннего угла по трём стенкам состоит из следующих шагов:

1. Инструмент перемещается на расстояние, заданное в `Feed distance`, и подводится к начальной точке измерительного цикла — точке пересечения трёх точек касания на

расстоянии, заданном в `Clearance 1`, `Clearance 2`, `Clearance 3` соответственно, вдоль их векторов. Перемещение на «подаче подвода»;
2. Инструмент подводится к точке касания первой стороны (перемещаясь на «рабочей подаче»), а затем возвращается (перемещаясь на «подаче длинных переходов») в предыдущее положение;
3. Инструмент подводится к точке касания второй стороны (перемещаясь на «рабочей подаче»), а затем возвращается (перемещаясь на «подаче длинных переходов») в предыдущее положение;
4. Инструмент подводится к точке касания третьей стороны (перемещаясь на «рабочей подаче»), а затем возвращается (перемещаясь на «подаче длинных переходов») в предыдущее положение;
5. Инструмент поднимается на расстояние, равное `Feed distance`.

Параметры

CLDArray	Тип	Описание
CmdPrm.Int[-1]	Integer	Тип измерительного цикла: значение Triple wall internal corner probing = 15
CmdPrm.Int[-2]	Integer	SubCode цикла, заданный в свойстве " SubCode for postprocessor " на вкладке `Job Assignment`
CmdPrm.Flt[-50]	Double	Feed distance, расстояние до начальной позиции цикла
CmdPrm.Flt[-56]	Double	Clearance 1, расстояние подвода к точке касания на первой стороне
CmdPrm.Flt[-57]	Double	Clearance 2, расстояние подвода к точке касания на второй стороне
CmdPrm.Flt[-58]	Double	Clearance 3, расстояние подвода к точке касания на третьей стороне

CLDArray	Тип	Описание
CmdPrm.Flt[-100]	Double	Значение точки касания на первой стенке по оси X
CmdPrm.Flt[-101]	Double	Значение точки касания на первой стенке по оси Y
CmdPrm.Flt[-102]	Double	Значение точки касания на первой стенке по оси Z
CmdPrm.Flt[-103]	Double	Значение целевого вектора точки касания на первой стенке по оси X
CmdPrm.Flt[-104]	Double	Значение целевого вектора точки касания на первой стенке по оси Y
CmdPrm.Flt[-105]	Double	Значение целевого вектора точки касания на первой стенке по оси Z
CmdPrm.Flt[-106]	Double	Значение точки касания на второй стенке по оси X
CmdPrm.Flt[-107]	Double	Значение точки касания на второй стенке по оси Y
CmdPrm.Flt[-108]	Double	Значение точки касания на второй стенке по оси Z
CmdPrm.Flt[-109]	Double	Значение целевого вектора точки касания на второй стенке по оси X
CmdPrm.Flt[-110]	Double	Значение целевого вектора точки касания на второй стенке по оси Y
CmdPrm.Flt[-111]	Double	Значение целевого вектора точки касания на второй стенке по оси Z

CLDArray	Тип	Описание
CmdPrm.Flt[-112]	Double	Значение точки касания на третьей стенке по оси X
CmdPrm.Flt[-113]	Double	Значение точки касания на третьей стенке по оси Y
CmdPrm.Flt[-114]	Double	Значение точки касания на третьей стенке по оси Z
CmdPrm.Flt[-115]	Double	Значение целевого вектора точки касания на третьей стенке по оси X
CmdPrm.Flt[-116]	Double	Значение целевого вектора точки касания на третьей стенке по оси Y
CmdPrm.Flt[-117]	Double	Значение целевого вектора точки касания на третьей стенке по оси Z
CmdPrm.Flt[-200]	Double	Значение исходной точки касания на первой стенке по оси X, заданное в Job Assignment
CmdPrm.Flt[-201]	Double	Значение исходной точки касания на первой стенке по оси Y, заданное в Job Assignment
CmdPrm.Flt[-202]	Double	Значение исходной точки касания на первой стенке по оси Z, заданное в Job Assignment
CmdPrm.Flt[-203]	Double	Значение исходного целевого вектора точки касания на первой стенке по оси X, заданное в Job Assignment

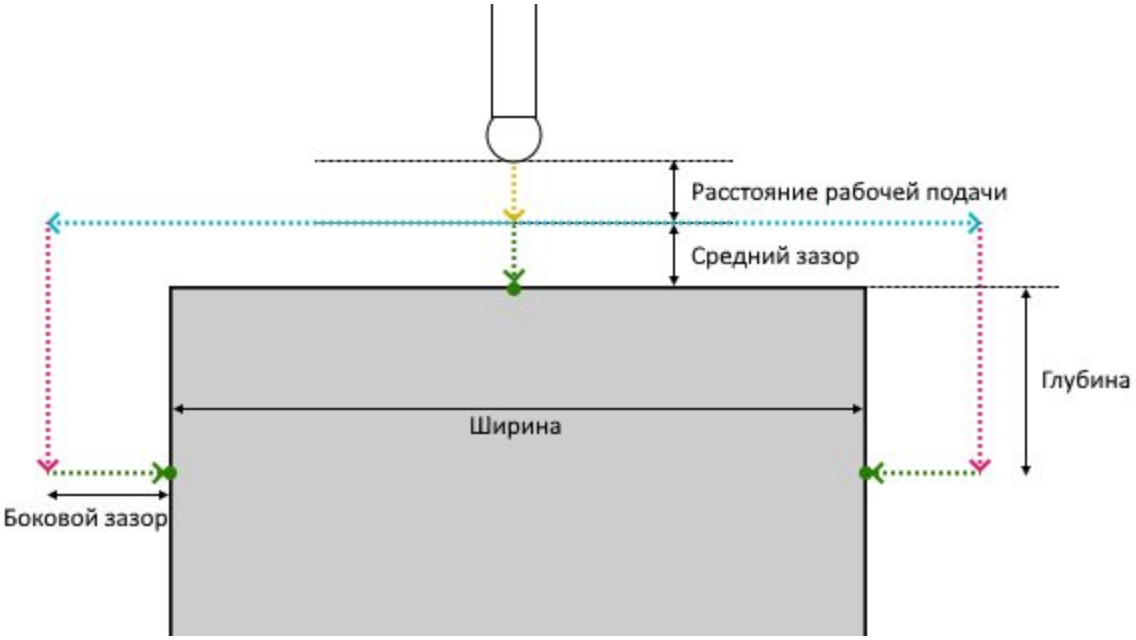
CLDArray	Тип	Описание
CmdPrm.Flt[-204]	Double	Значение исходного целевого вектора точки касания на первой стенке по оси Y, заданное в Job Assignment
CmdPrm.Flt[-205]	Double	Значение исходного целевого вектора точки касания на первой стенке по оси Z, заданное в Job Assignment
CmdPrm.Flt[-206]	Double	Значение исходной точки касания на второй стенке по оси X, заданное в Job Assignment
CmdPrm.Flt[-207]	Double	Значение исходной точки касания на второй стенке по оси Y, заданное в Job Assignment
CmdPrm.Flt[-208]	Double	Значение исходной точки касания на второй стенке по оси Z, заданное в Job Assignment
CmdPrm.Flt[-209]	Double	Значение исходного целевого вектора точки касания на второй стенке по оси X, заданное в Job Assignment
CmdPrm.Flt[-210]	Double	Значение исходного целевого вектора точки касания на второй стенке по оси Y, заданное в Job Assignment

CLDArray	Тип	Описание
CmdPrm.Flt[-211]	Double	Значение исходного целевого вектора точки касания на второй стенке по оси Z, заданное в Job Assignment
CmdPrm.Flt[-212]	Double	Значение исходной точки касания на третьей стенке по оси X, заданное в Job Assignment
CmdPrm.Flt[-213]	Double	Значение исходной точки касания на третьей стенке по оси Y, заданное в Job Assignment
CmdPrm.Flt[-214]	Double	Значение исходной точки касания на третьей стенке по оси Z, заданное в Job Assignment
CmdPrm.Flt[-215]	Double	Значение исходного целевого вектора точки касания на третьей стенке по оси X, заданное в Job Assignment
CmdPrm.Flt[-216]	Double	Значение исходного целевого вектора точки касания на третьей стенке по оси Y, заданное в Job Assignment
CmdPrm.Flt[-217]	Double	Значение исходного целевого вектора точки касания на третьей стенке по оси Z, заданное в Job Assignment

Сопутствующие ссылки

- Измерительный цикл (WProbing)
- Расширенный цикл (EXTCYCLE)
- Модель доступа к CLData

Параметры измерения перемычки (три точки)



Измерение перемычки (три точки) состоит из следующих шагов:

1. Инструмент перемещается на расстояние, заданное в `Feed distance`, и подводится к начальной позиции цикла. Перемещение на «подаче подвода»;
2. Инструмент подводится к точке, удалённой от левой точки касания вдоль её `Target vector` на расстояние `Side clearance`, но поднятой на значение `Top clearance` + значение `Depth`. Перемещение на «подаче длинных переходов»;
3. Инструмент опускается на расстояние, равное значению `Top clearance` + значение `Depth`. Перемещение на «подаче длинных переходов»;
4. Инструмент подводится к левой точке касания (перемещаясь на «рабочей подаче»), а затем возвращается (перемещаясь на «подаче длинных переходов») в предыдущее положение;
5. Инструмент поднимается на расстояние, равное значению `Top clearance` + значение `Depth`. Перемещение на «подаче длинных переходов»;
6. Инструмент возвращается в начальную позицию;
7. Шаги 2-6 повторяются для правой точки касания;
8. Инструмент подводится к средней точке касания (перемещаясь на «рабочей подаче»), а затем возвращается (перемещаясь на «подаче длинных переходов») в предыдущее положение;
9. Инструмент поднимается на расстояние, равное `Feed distance`.

Параметры

CLDArray	Тип	Описание
CmdPrm.Int[-1]	Integer	Тип измерительного цикла: значение Web probing (three points) = 10
CmdPrm.Int[-2]	Integer	SubCode цикла, заданный в свойстве " SubCode for postprocessor " на вкладке `Job Assignment`
CmdPrm.Flt[-50]	Double	Feed distance, расстояние до начальной позиции цикла
CmdPrm.Flt[-51]	Double	Depth, расстояние от средней стороны до центра левой и правой точек
CmdPrm.Flt[-53]	Double	Width, расстояние от левой точки до правой точки
CmdPrm.Flt[-56]	Double	Side clearance, расстояние подвода к правой / левой точке касания
CmdPrm.Flt[-57]	Double	Middle clearance, расстояние подвода к средней точке касания
CmdPrm.Flt[-100]	Double	Значение левой точки касания по оси X
CmdPrm.Flt[-101]	Double	Значение левой точки касания по оси Y
CmdPrm.Flt[-102]	Double	Значение левой точки касания по оси Z
CmdPrm.Flt[-103]	Double	Значение левого целевого вектора по оси X
CmdPrm.Flt[-104]	Double	Значение левого целевого вектора по оси Y

CLDArray	Тип	Описание
CmdPrm.Flt[-105]	Double	Значение левого целевого вектора по оси Z
CmdPrm.Flt[-106]	Double	Значение правой точки касания по оси X
CmdPrm.Flt[-107]	Double	Значение правой точки касания по оси Y
CmdPrm.Flt[-108]	Double	Значение правой точки касания по оси Z
CmdPrm.Flt[-109]	Double	Значение правого целевого вектора по оси X
CmdPrm.Flt[-110]	Double	Значение правого целевого вектора по оси Y
CmdPrm.Flt[-111]	Double	Значение правого целевого вектора по оси Z
CmdPrm.Flt[-112]	Double	Значение средней точки касания по оси X
CmdPrm.Flt[-113]	Double	Значение средней точки касания по оси Y
CmdPrm.Flt[-114]	Double	Значение средней точки касания по оси Z
CmdPrm.Flt[-115]	Double	Значение среднего целевого вектора по оси X
CmdPrm.Flt[-116]	Double	Значение среднего целевого вектора по оси Y
CmdPrm.Flt[-117]	Double	Значение среднего целевого вектора по оси Z
CmdPrm.Flt[-200]	Double	Значение исходной левой точки касания по оси X, заданное в Job Assignment

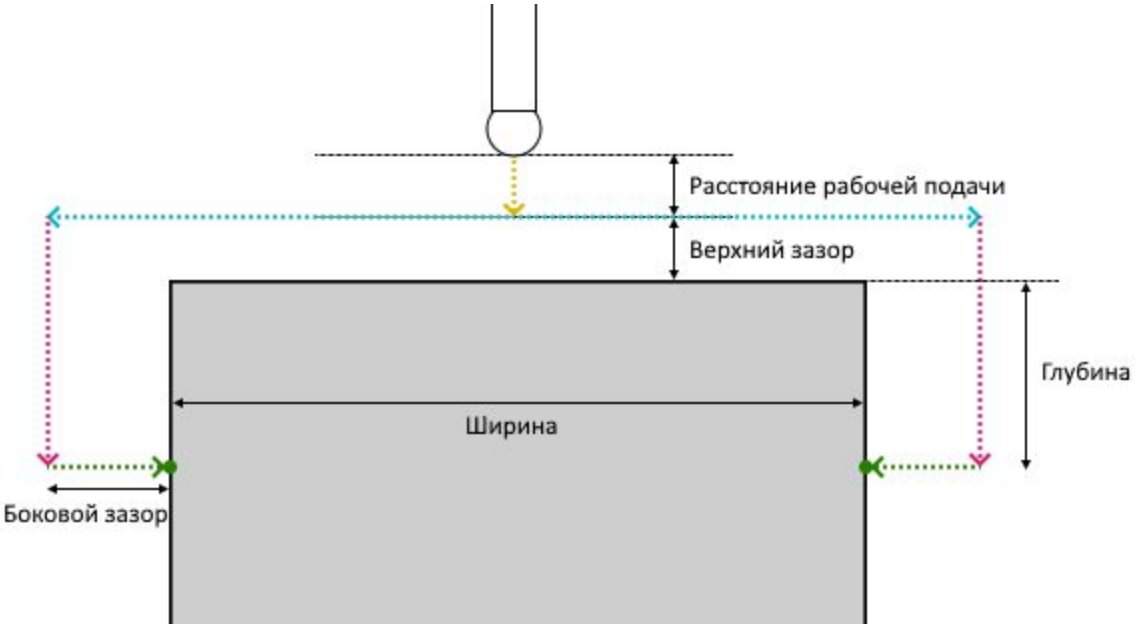
CLDArray	Тип	Описание
CmdPrm.Flt[-201]	Double	Значение исходной левой точки касания по оси Y, заданное в Job Assignment
CmdPrm.Flt[-202]	Double	Значение исходной левой точки касания по оси Z, заданное в Job Assignment
CmdPrm.Flt[-203]	Double	Значение исходного левого целевого вектора по оси X, заданное в Job Assignment
CmdPrm.Flt[-204]	Double	Значение исходного левого целевого вектора по оси Y, заданное в Job Assignment
CmdPrm.Flt[-205]	Double	Значение исходного левого целевого вектора по оси Z, заданное в Job Assignment
CmdPrm.Flt[-206]	Double	Значение исходной правой точки касания по оси X, заданное в Job Assignment
CmdPrm.Flt[-207]	Double	Значение исходной правой точки касания по оси Y, заданное в Job Assignment
CmdPrm.Flt[-208]	Double	Значение исходной правой точки касания по оси Z, заданное в Job Assignment
CmdPrm.Flt[-209]	Double	Значение исходного правого целевого вектора по оси X, заданное в Job Assignment
CmdPrm.Flt[-210]	Double	Значение исходного правого целевого вектора по оси Y, заданное в Job Assignment

CLDArray	Тип	Описание
CmdPrm.Flt[-211]	Double	Значение исходного правого целевого вектора по оси Z, заданное в Job Assignment
CmdPrm.Flt[-212]	Double	Значение исходной средней точки касания по оси X, заданное в Job Assignment
CmdPrm.Flt[-213]	Double	Значение исходной средней точки касания по оси Y, заданное в Job Assignment
CmdPrm.Flt[-214]	Double	Значение исходной средней точки касания по оси Z, заданное в Job Assignment
CmdPrm.Flt[-215]	Double	Значение исходного среднего целевого вектора по оси X, заданное в Job Assignment
CmdPrm.Flt[-216]	Double	Значение исходного среднего целевого вектора по оси Y, заданное в Job Assignment
CmdPrm.Flt[-217]	Double	Значение исходного среднего целевого вектора по оси Z, заданное в Job Assignment
CmdPrm.Flt[-66]	Integer	Compensate dimensions, возвращаемые значения: 0 - Off, 1 - On

Сопутствующие ссылки

- Измерительный цикл (WProbing)
- Расширенный цикл (EXTCYCLE)
- Модель доступа к CLData

Параметры измерения перемычки



Измерение перемычки состоит из следующих шагов:

1. Инструмент перемещается на расстояние, заданное в `Feed distance`, и подводится к начальной позиции цикла. Перемещение на «подаче подвода»;
2. Инструмент подводится к точке, удалённой от первой точки касания вдоль её `Target vector` на расстояние `Side clearance`, но поднятой на значение `Top clearance` + значение `Depth`. Перемещение на «подаче длинных переходов»;
3. Инструмент опускается на расстояние, равное значению `Top clearance` + значение `Depth`. Перемещение на «подаче длинных переходов»;
4. Инструмент подводится к первой точке касания (перемещаясь на «рабочей подаче»), а затем возвращается (перемещаясь на «подаче длинных переходов») в предыдущее положение;
5. Инструмент поднимается на расстояние, равное значению `Top clearance` + значение `Depth`. Перемещение на «подаче длинных переходов»;
6. Инструмент возвращается в начальную позицию;
7. Шаги 2-6 повторяются для второй точки касания;
8. Инструмент поднимается на расстояние, равное `Feed distance`.

Параметры

CLDArray	Тип	Описание
CmdPrm.Int[-1]	Integer	Тип измерительного цикла: значение Web probing = 9

CLDArray	Тип	Описание
CmdPrm.Int[-2]	Integer	SubCode цикла, заданный в свойстве " SubCode for postprocessor " на вкладке `Job Assignment`
CmdPrm.Flt[-50]	Double	Feed distance, расстояние до начальной позиции цикла
CmdPrm.Flt[-51]	Double	Depth, расстояние от верхней стороны до центра точек касания
CmdPrm.Flt[-53]	Double	Width, расстояние от Touch point 1 до Touch point 2
CmdPrm.Flt[-55]	Double	Top clearance, расстояние от верхней стороны
CmdPrm.Flt[-56]	Double	Side clearance, расстояние подвода к точке касания 1 / точке касания 2
CmdPrm.Flt[-100]	Double	Значение первой точки касания по оси X
CmdPrm.Flt[-101]	Double	Значение первой точки касания по оси Y
CmdPrm.Flt[-102]	Double	Значение первой точки касания по оси Z
CmdPrm.Flt[-103]	Double	Значение первого целевого вектора по оси X
CmdPrm.Flt[-104]	Double	Значение первого целевого вектора по оси Y
CmdPrm.Flt[-105]	Double	Значение первого целевого вектора по оси Z

CLDArray	Тип	Описание
CmdPrm.Flt[-106]	Double	Значение второй точки касания по оси X
CmdPrm.Flt[-107]	Double	Значение второй точки касания по оси Y
CmdPrm.Flt[-108]	Double	Значение второй точки касания по оси Z
CmdPrm.Flt[-109]	Double	Значение второго целевого вектора по оси X
CmdPrm.Flt[-110]	Double	Значение второго целевого вектора по оси Y
CmdPrm.Flt[-111]	Double	Значение второго целевого вектора по оси Z
CmdPrm.Flt[-200]	Double	Значение исходной первой точки касания по оси X, заданное в Job Assignment
CmdPrm.Flt[-201]	Double	Значение исходной первой точки касания по оси Y, заданное в Job Assignment
CmdPrm.Flt[-202]	Double	Значение исходной первой точки касания по оси Z, заданное в Job Assignment
CmdPrm.Flt[-203]	Double	Значение исходного первого целевого вектора по оси X, заданное в Job Assignment
CmdPrm.Flt[-204]	Double	Значение исходного первого целевого вектора по оси Y, заданное в Job Assignment
CmdPrm.Flt[-205]	Double	Значение исходного первого целевого вектора по оси Z, заданное в Job Assignment

CLDArray	Тип	Описание
CmdPrm.Flt[-206]	Double	Значение исходной второй точки касания по оси X, заданное в Job Assignment
CmdPrm.Flt[-207]	Double	Значение исходной второй точки касания по оси Y, заданное в Job Assignment
CmdPrm.Flt[-208]	Double	Значение исходной второй точки касания по оси Z, заданное в Job Assignment
CmdPrm.Flt[-209]	Double	Значение исходного второго целевого вектора по оси X, заданное в Job Assignment
CmdPrm.Flt[-210]	Double	Значение исходного второго целевого вектора по оси Y, заданное в Job Assignment
CmdPrm.Flt[-211]	Double	Значение исходного второго целевого вектора по оси Z, заданное в Job Assignment
CmdPrm.Flt[-66]	Integer	Compensate dimensions, возвращаемые значения: 0 - Off, 1 - On

Сопутствующие ссылки

- Измерительный цикл (WProbing)
- Расширенный цикл (EXTCYCLE)
- Модель доступа к CLData

FEDRAT - Подача

Отображение команды

Как команда отображается в списке команд CLData:

```
FEDRAT NM nm, K k, MMPM(315) | MMPR(316)
```

Команда FEDRAT задаёт значение и тип подачи инструмента. Значение подачи сохраняется до следующей команды FEDRAT или команды RAPID.

CAM-система использует FEDRAT для передачи кода условий обработки.

Доступ из sppx

Обработчик: `program Fedrat`

Параметр	Массив CLD	Имя CLD	Описание
nm	CLD[1]	CLD.NM	Значение подачи.
k	CLD[2]	CLD.K	Тип подачи (рабочая подача, подача подвода, подача отвода, подача перехода, подача врезания и т. д.). 0 - рабочая, 2 - подача первого прохода, 4 - подача подвода, 8 - подача отвода, 16 - подача врезания, 32 - подача чистового прохода 64 - переход на следующий проход, 128 - подача возврата, 256 - подача подхода. Для операций

Параметр	Массив CLD	Имя CLD	Описание
			электроэрозионной обработки CAM-система использует этот параметр для передачи кода условий обработки.
MMPM(315) MMPR(316)	CLD[3]	CLD.MMPM	Единицы измерения подачи: MMPM(315) - мм в минуту, MMPR(316) - мм на оборот

Пример (из постпроцессора Fanuc (30i)_Mill):

```
program Fedrat
  FEED_ = cld[1]   ! заполнение регистра FEED_
  if INTERP_ = 0 then INTERP_ = 1   ! G1
  if cld[3]=315 then GFeed = 94
                else GFeed = 95
end
```

Доступ из .NET

Обработчики:

```
public override void OnMoveVelocity(ICLDMoveVelocityCommand cmd, CLDArray cld)   //
общий обработчик скорости (FEDRAT и RAPID)
public override void OnFeedrate(ICLDFeedrateCommand cmd, CLDArray cld)   //
рабочая подача
```

Наследование: `ICLDFeedrateCommand : ICLDMoveVelocityCommand : ICLDCommand`.

Читайте параметры из строго типизированного объекта `cmd` (рекомендуется); также доступны «сырой» массив `cld[i]` (те же индексы, что и в sppx) и именованные помощники `cmd["..."]`. В таблице перечислены члены команды, включая унаследованные от базовых интерфейсов; общие члены `ICLDCommand` (навигация, `CLD`, `CLDFile`, `TechOperation`, ...) описаны в разделе Модель доступа к CLData.

Член	Тип	Описание
`cmd.IsRapid`	`bool`	Содержит "true", если команда является командой RAPID, иначе - "false".
`cmd.FeedKind`	`CLDFeedKind`	Вид подачи (рабочая, ускоренная, подвод, возврат, чистовая и т. д.). Возможные значения: `Working` (0) - Вид подачи для базовых рабочих перемещений. `Rapid` (1) - Вид подачи для ускоренных перемещений. `First` (2) - Вид подачи для рабочих перемещений на первом (черновом) проходе. `Engage` (4) - Вид подачи для перемещений, входящих в первую точку обработки (резания). `Retract` (8) - Вид подачи для перемещений, покидающих последнюю точку обработки (резания). `Plunge` (16) - Вид подачи для перемещений, вертикально врезающихся в зону обработки. `Finish` (32) - Вид подачи для рабочих перемещений на последнем (чистовом) проходе. `Next` (64) - Вид подачи для коротких связующих перемещений. `Return` (128) - Вид подачи для перемещений, уводящих инструмент от места обработки.

Член	Тип	Описание
		`Approach` (256) - Вид подачи для перемещений, приближающих инструмент к месту обработки. `Rapid5D` (512) - Значение вида подачи, зарезервированное для внутреннего использования. `TransitionOnSafe` (1024) - Вид подачи для перемещений, выполняемых в безопасной зоне. `ApproachFromSafe` (2048) - Вид подачи для перемещений из безопасной зоны к месту обработки. `ReturnToSafe` (4096) - Вид подачи для перемещений от места обработки в безопасную зону. `LongNext` (8192) - Вид подачи для длинных связующих перемещений. `ThreadPitch` (16384) - Вид подачи для перемещений при нарезании резьбы. `Overload` (1073741824) - Значение вида подачи, зарезервированное для внутреннего использования.
`cmd.FeedCode`	`int`	Числовое представление флага "FeedKind", определяющего вид подачи (рабочая, ускоренная, подвод, возврат, чистовая и т. д.). Для проволочной электроэрозионной обработки содержит

Член	Тип	Описание
		числовой код строки в таблице условий обработки.
`cmd.FeedUnits`	`CLDFeedUnits`	Единицы измерения подачи (мм/мин, мм/об и т. д.) Возможные значения: `MPM` (0) - Текущие линейные единицы (мм или дюйм) в минуту. `MPR` (1) - Текущие линейные единицы (мм или дюйм) на оборот. `ConditionCode` (2) - Числовой код строки в таблице условий обработки (для проволочной электроэрозионной обработки).
`cmd.FeedValue`	`double`	Значение подачи.

Пример - `OnMoveVelocity` (из постпроцессора Fanuc (30i)_Mill_DN):

```
public override void OnMoveVelocity(ICLDMoveVelocityCommand cmd, CLDArray cld)
{
    if (cmd.IsRapid) {
        nc.GInterp.v = 0;
    } else {
        if (nc.GInterp == 0)
            nc.GInterp.v = 1;
        nc.F.v = cmd.FeedValue;
        if (cmd.FeedUnits==CLDFeedUnits.MPR)
            nc.GFeed.v = 95;
        else
            nc.GFeed.v = 94;
        if (IsFirstFeed)
        {
            nc.F.Show();
            nc.GFeed.Show();
            IsFirstFeed = false;
        }
```

```
    }
}
```

Сопутствующие ссылки

- Модель доступа к CLData

Filter - Выходной фильтр (глобальный обработчик)

> Не является командой CLData. `Filter` - это глобальный обработчик, вызываемый для каждой строки управляющей программы, независимо от того, какая команда CLData (или какой ещё код) её сформировала. Он описан здесь, чтобы все обработчики постпроцессора можно было найти в одном месте и в единообразном виде.

Доступ из sppx

Каждый постпроцессор включает предопределённую подпрограмму Filter, которую можно найти в списке подпрограмм. Filter вызывается сразу после того, как кадр управляющей программы сформирован, но до того, как он будет добавлен в текст управляющей программы. Таким образом, эта подпрограмма позволяет пользователю редактировать каждый кадр управляющей программы в момент его вывода. Например, можно заменять символы или добавлять метки в конец или начало строки.

По умолчанию система создаёт для Filter следующий прототип подпрограммы:

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

Сформированный кадр УП передаётся через параметр S. Все изменения этой строки попадут в текст управляющей программы.

Пример:

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

Выполнение этого кода для кадра "17 L Z-10 R0 F0" даст в результате строку управляющей программы "17 L Z-10 R0 FMAX".

Доступ из .NET

Аналог в .NET - обработчик `OnFilterString`:

```
public override void OnFilterString(ref string s, TNCFile ncFile, INCLabel label)
```

Строка передаётся по ссылке в параметре `s` - изменяйте её на месте. `ncFile` - целевой файл УП, `label` - текущая метка.

Пример (из постпроцессора Goodway SW):

```
public override void OnFilterString(ref string s, TNCFile ncFile, INCLabel label)
{
    SynchPoints.CheckForOutput(s);
    if ((ncFile is NCFile ncx) && (ncx.NeedSlash) && (!String.IsNullOrEmpty(s)))
        s= "/"+s;
}
```

Сопутствующие ссылки

- Модель доступа к CLData

FINI - Завершающая запись

Отображение команды

Как команда отображается в списке команд CLData:

```
FINI
```

FINI — эта команда является последней в списке технологических команд проекта. Используйте FINI для формирования завершающих блоков управляющей программы.

Доступ из sppx

Обработчик: `program Fini`

Пример (из постпроцессора Fanuc (30i)_Mill):

```
program Fini
  OutBlock                          ! output
  OutPut "M30"
  OutPut "%"
  OutPut ""
  NCSub.Output

  s$ = "G53Z0."
  CrLf$ = Chr(13) + Chr(10)
  if (HasAxisA<>0) or (HasAxisB<>0) or (HasAxisC<>0) then begin
    s$ = s$ + CrLf$ + "G53"
    if (HasAxisA<>0) then
      s$ = s$ + "A0."
    if (HasAxisB<>0) then
      s$ = s$ + "B0."
    if (HasAxisC<>0) then
      s$ = s$ + "C0."
  end
  ReplNCStr("<StartGoHome>", s$)
end
```

Доступ из .NET

Обработчики:

```
public override void OnFini(ICLDFiniCommand cmd, CLDArray cld)
public override void OnFinishProject(ICLDProject prj)   // project-level, raised
on FINI
```

Наследование: `ICLDFiniCommand : ICLDCommand`.

Читайте параметры из строго типизированного объекта `cmd` (рекомендуется); доступны также «сырой» массив `cld[i]` (те же индексы, что и в sppx) и именованные обёртки `cmd["..."]`. В таблице перечислены члены команды, включая унаследованные от базовых интерфейсов; общие члены `ICLDCommand` (навигация, `CLD`, `CLDFile`, `TechOperation`, ...) описаны в разделе Модель доступа к CLData.

Пример - `OnFinishProject` (из постпроцессора Fanuc (30i)_Mill_DN):

```
public override void OnFinishProject(ICLDProject prj)
{
    nc.Block.Out();
    nc.Output("M30");
}
```

Сопутствующие ссылки

- Модель доступа к CLData

FROM - Начальная точка

Отображение команды

Как команда отображается в списке команд CLData:

```
FROM COUNT 5, MACHINE, X 0, Y 0, Z 254, A 0, C 0, GEOM X 10, Y 0, Z 194, NX 0, NY 0,
NZ 1, NW 0, MSF 0, Time 0
```

Команда FROM задаёт координаты исходной точки. Обычно команда FROM вызывается только один раз, в начале обработки, и координаты первого перемещения, как правило, совпадают с координатами команды FROM. Если требуется избежать вывода исходной точки в управляющую программу, используйте команду FROM для инициализации значений регистров без их вывода в код управляющей программы. Это предотвращает вывод команд перемещения, координаты которых совпадают с координатами команды FROM.

Доступ из sppx

Обработчик: `program From`

Параметр	Массив CLD	Имя CLD	Описание
X Y Z	CLD[1] CLD[2] CLD[3]	CLD.X CLD.Y CLD.Z	Пространственные координаты исходной точки кончика инструмента относительно активной системы координат заготовки
NX NY NZ NW	CLD[4] CLD[5] CLD[6] CLD[7]	CLD.NX CLD.NY CLD.NZ CLD.NW	Ориентация системы координат инструмента в исходной точке относительно активной системы координат заготовки. Способ задания ориентации зависит от настройки в схеме станка. Это могут быть углы Эйлера в

Параметр	Массив CLD	Имя CLD	Описание
			различных соглашениях, вектор нормали инструмента, кватернион или повёрнутый вектор.
MSF	CLD[8]	CLD.MSF	Флаги состояния станка (флипы). Актуально для роботов и станков, которые могут иметь несколько конфигураций физических осей для одной и той же пространственной позиции. См. описание в следующей таблице.
Time	CLD[9]	CLD.Time	Время перемещения.
AxesCount	CLD[10]	CLD.AxesCount	Количество физических координат станка, задействованных в этой команде.
r1	CLD[11]		Индекс оси с именем Axis1Pos в списке координат.
Val1	CLD[12]		Значение координаты станка Axis1Pos.
r2	CLD[13]		Индекс оси с именем Axis2Pos в списке координат.

Параметр	Массив CLD	Имя CLD	Описание
Val2	CLD[14]		Значение координаты станка Axis2Pos.
...	...		...
rN	CLD[2*N+9]		Индекс оси с именем AxisNPos в списке координат.
ValN	CLD[2*N+10]		Значение координаты станка AxisNPos.

Параметры, доступные через оператор cmd

TCLDFrom: ComplexType	Команда координат исходной точки
Axes: Array, Key="AxisID"	cmd.Ptr["Axes"] - массив структур типа Axis. Одна команда может включать перемещение по нескольким осям, положение каждой из которых хранится в этом массиве.
Axis: ComplexType	cmd.Ptr["Axes"].Item[Index] или cmd.Ptr["Axes(AxisName)"] - отдельный элемент массива Axes. Содержит информацию о перемещении по одной оси станка. Доступ к элементам массива возможен как по индексу, так и по ключевому полю. Здесь AxisName - значение ключевого поля, которое должно совпадать со значением поля AxisID.
AxisID: String	cmd.Str["Axes(AxisName).AxisID"] - идентификатор оси станка, для которой задана новая позиция. Определяется схемой станка.
Value: Double	cmd.Flt["Axes(AxisName).Value"] - новая позиция оси станка, в которую она перемещается.

TCLDFrom: ComplexType	Команда координат исходной точки
Pos5D: ComplexType	cmd.Ptr["Pos5D"] - структура, содержащая координаты, определяющие конечное положение инструмента в пространстве. Определяет не только положение настроечной точки инструмента (поля X, Y и Z - декартовы координаты), но и ориентацию инструмента относительно текущей системы координат (поля NX, NY, NZ и NW). В зависимости от конфигурации системы (заданной в схеме станка) ориентация инструмента может задаваться: вектором нормали n = (NX, NY, NZ); пространственными углами поворота вокруг осей координат - углами Эйлера (порядок поворотов также задаётся настройками) A = NX, B = NY, C = NZ; кватернионом q = (NX, NY, NZ, NW).
X: Double	cmd.Flt["Pos5D.X"] - декартова координата X точки кончика инструмента.
Y: Double	cmd.Flt["Pos5D.Y"] - декартова координата Y точки кончика инструмента.
Z: Double	cmd.Flt["Pos5D.Z"] - декартова координата Z точки кончика инструмента.
NX: Double	cmd.Flt["Pos5D.NX"] - в зависимости от конфигурации системы может быть компонентой NX вектора нормали инструмента, углом поворота вокруг одной из осей координат или коэффициентом кватерниона.
NY: Double	cmd.Flt["Pos5D.NY"] - в зависимости от конфигурации системы может быть компонентой NY вектора нормали инструмента, углом поворота вокруг одной из осей координат или коэффициентом кватерниона.

TCLDFrom: ComplexType	Команда координат исходной точки
NZ: Double	cmd.Flt["Pos5D.NZ"] - в зависимости от конфигурации системы может быть компонентой NZ вектора нормали инструмента, углом поворота вокруг одной из осей координат или коэффициентом кватерниона.
NW: Double	cmd.Flt["Pos5D.NW"] - если выбран способ ориентации инструмента в виде кватерниона, это четвёртый коэффициент кватерниона.
MachineStateFlags: Integer	cmd.Ptr["MachineStateFlags"].Bit[i], i=(0..31). Целое число, являющееся битовым полем из 32 бит от 0 до 31. Если для заданного положения инструмента в пространстве существует несколько вариантов расположения осей станка, каждый бит этого поля определяет выбор одного из возможных решений. Например, для большинства положений инструмента у пятиосевого станка с осями A и C есть два возможных решения - с положительным значением оси A и с отрицательным значением оси A. Нулевой бит этого поля выберет одно из этих решений. Для стандартного шестиосевого сочленённого робота многие положения инструмента в пространстве могут быть обеспечены разными комбинациями положений трёх его ключевых сочленений (основания, локтя и запястья), что даёт в сумме 8 возможных решений. В этом случае нулевой бит этого поля будет определять положение основания, первый бит - положение локтя, а третий бит будет задавать положение запястья робота. Таким образом, значение каждого из этих битов полностью определяется используемой схемой станка.

TCLDFrom: ComplexType	Команда координат исходной точки
	При использовании станков, для которых эти флаги не нужны, вывод этого параметра в CLData можно отключить в схеме станка.
Time: Double	Время перемещения в минутах.

Список координат, передаваемых через CLData команды FROM, определяется схемой станка CAM-системы.

Используйте массив GMA для обработки команды FROM. Свойствам массива GMA автоматически присваиваются предварительно рассчитанные значения параметров координатного перемещения. Параметры команды также доступны через оператор cmd.

Пример (из постпроцессора Heidenhain (iTNC530)_Mill):

```
program From
  !X = CLD[1]; X@ = X; XT_ = X
  !Y = CLD[2]; Y@ = Y; YT_ = Y
  !Z = CLD[3]; Z@ = Z; ZT_ = Z
  !A = 0; A@ = A; AT_ = A
  !B = 0; B@ = B; BT_ = B
  !C = 0; C@ = C; CT_ = C
End
```

Доступ из .NET

Обработчики:

```
public override void OnBeforeMovement(ICLDMotionCommand cmd, CLDArray cld)   //
групповой обработчик - перед любой командой перемещения
public override void OnFrom(ICLDFromCommand cmd, CLDArray cld)
public override void OnAfterMovement(ICLDMotionCommand cmd, CLDArray cld)   //
групповой обработчик - после любой команды перемещения
```

Наследование: `ICLDFromCommand : ICLDMultiMotionCommand : ICLDMotionCommand : ICLDCommand`.

Читайте параметры из строго типизированного объекта `cmd` (рекомендуется); доступны также «сырой» массив `cld[i]` (те же индексы, что и в sppx) и именованные обёртки `cmd["..."]`. В таблице перечислены члены команды, включая унаследованные от базовых интерфейсов;

общие члены `ICLDCommand` (навигация, `CLD`, `CLDFile`, `TechOperation`, ...) описаны в разделе Модель доступа к CLData.

Член	Тип	Описание
`cmd.EN`	`TInpRotation`	Конечная нормаль или ориентация перемещения в декартовой системе координат. В зависимости от текущих настроек схемы станка это может быть вектор нормали, углы Эйлера, кватернион и т. д.
`cmd.MSFlags`	`BitArray`	Флаги состояния станка или флипы робота, которые вместе с декартовыми координатами помогают определить конкретное состояние механизма в случае избыточности. То, какой бит отвечает за какой флаг, зависит от типа и конфигурации оборудования.
`cmd.Time`	`double`	Продолжительность перемещения в минутах.
`cmd.AxesCount`	`int`	Количество осей, которые перемещение содержит в списке Axis[].
`cmd.Axis`	`ICLDMultiMotionAxesIndexer`	Список осей перемещения [0..(AxesCount-1)].
`cmd.Axes`	`IEnumerable`	Объект-перечислитель осей перемещения для удобного использования синтаксиса foreach.

Член	Тип	Описание
`cmd.HasAxis(...)`	`bool`	Возвращает "true", если перемещение содержит ось с указанным текстовым идентификатором. Иначе возвращает "false".
`cmd.EP`	`TInp3DPoint`	Конечная точка (X, Y, Z) перемещения

Пример - `OnFrom` (из постпроцессора Sinumerik (840D)_Mill_DN):

```
public override void OnFrom(ICLDFromCommand cmd, CLDArray cld)
{
    nc.X.Hide(cld[1]); nc.Y.Hide(cld[2]); nc.Z.Hide(cld[3]);
    LastPnt.X = cld[1]; LastPnt.Y = cld[2]; LastPnt.Z = cld[3];
}
```

Сопутствующие ссылки

- Модель доступа к CLData

GOHOME - Возврат в исходное положение

Отображение команды

Как команда отображается в списке команд CLData:

```
GOHOME COUNT 2, MACHINE, X 34.725, Y 32, GEOM X 34.725, Y 32, Z 194, NX 0, NY 0, NZ 1, NW 0, MSF 0, Time 0.003
```

GOHOME — эта команда используется для перемещения инструмента в ноль станка. Обычно команда GOHOME транслируется в код G28. Ноль станка определяется системой ЧПУ, постпроцессор получает только код команды.

Доступ из sppx

Обработчик: `program GoHome`

Параметр	Массив CLD	Описание
X Y Z	CLD[1] CLD[2] CLD[3]	Пространственные координаты точки инструмента относительно активной СК заготовки
NX NY NZ NW	CLD[4] CLD[5] CLD[6] CLD[7]	Ориентация системы координат инструмента относительно активной СК заготовки. Способ задания ориентации зависит от настроек схемы станка. Это могут быть углы Эйлера в разных соглашениях, нормальный вектор инструмента, кватернион или повёрнутый вектор.
MSF	CLD[8]	Флаги состояния станка (flips). Актуально для роботов и станков, у которых одному и тому же положению в пространстве может соответствовать несколько

Параметр	Массив CLD	Описание
		конфигураций физических осей. См. описание в следующей таблице.
Time	CLD[9]	Время перемещения.
AxesCount	CLD[10]	Количество физических координат станка, задействованных в этой команде.
r1	CLD[11]	Индекс оси с именем Axis1Pos в списке координат.
Val1	CLD[12]	Значение координаты станка Axis1Pos.
r2	CLD[13]	Индекс оси с именем Axis2Pos в списке координат.
Val2	CLD[14]	Значение координаты станка Axis2Pos.
...	...	...
rN	CLD[2*N+9]	Индекс оси с именем AxisNPos в списке координат.
ValN	CLD[2*N+10]	Значение координаты станка AxisNPos.

Параметры, доступные через оператор cmd

TCLDGoHome: ComplexType	Команда возврата станка в исходное положение
Axes: Array, Key="AxisID"	cmd.Ptr["Axes"] - массив структур типа Axis. Одна команда может включать перемещение по нескольким осям, положение каждой из них хранится в этом массиве.

TCLDGoHome: ComplexType	Команда возврата станка в исходное положение
Axis: ComplexType	cmd.Ptr["Axes"].Item[Index] или cmd.Ptr["Axes(AxisName)"] - отдельный элемент массива Axes. Содержит информацию о перемещении по одной оси станка. Доступ к элементам массива возможен как по индексу, так и по ключевому полю. Здесь AxisName - значение ключевого поля, которое должно совпадать со значением поля AxisID.
AxisID: String	cmd.Str["Axes(AxisName).AxisID"] - идентификатор оси станка, для которой задано новое положение. Определяется схемой станка.
Value: Double	cmd.Flt["Axes(AxisName).Value"] - новое положение оси станка, в которое она перемещается.
Pos5D: ComplexType	cmd.Ptr["Pos5D"] - структура, содержащая координаты, определяющие конечное положение инструмента в пространстве. Определяет не только положение точки настройки инструмента (поля X, Y и Z - декартовы координаты), но и ориентацию инструмента относительно текущей системы координат (поля NX, NY, NZ и NW). В зависимости от конфигурации системы (заданной в схеме станка) ориентация инструмента может задаваться: нормальным вектором n = (NX, NY, NZ), пространственными углами поворота вокруг осей координат - углами Эйлера (порядок поворотов также задаётся в настройках) A = NX, B = NY, C = NZ, кватернионом q = (NX, NY, NZ, NW).

TCLDGoHome: ComplexType	Команда возврата станка в исходное положение
X: Double	cmd.Flt["Pos5D.X"] - декартова координата X положения точки инструмента.
Y: Double	cmd.Flt["Pos5D.Y"] - декартова координата Y положения точки инструмента.
Z: Double	cmd.Flt["Pos5D.Z"] - декартова координата Z положения точки инструмента.
NX: Double	cmd.Flt["Pos5D.NX"] - в зависимости от конфигурации системы может быть компонентом NX нормального вектора инструмента, углом поворота вокруг одной из осей координат либо коэффициентом кватерниона.
NY: Double	cmd.Flt["Pos5D.NY"] - в зависимости от конфигурации системы может быть компонентом NY нормального вектора инструмента, углом поворота вокруг одной из осей координат либо коэффициентом кватерниона.
NZ: Double	cmd.Flt["Pos5D.NZ"] - в зависимости от конфигурации системы может быть компонентом NZ нормального вектора инструмента, углом поворота вокруг одной из осей координат либо коэффициентом кватерниона.
NW: Double	cmd.Flt["Pos5D.NW"] - если выбран способ ориентации инструмента в виде кватерниона, это четвёртый коэффициент кватерниона.
MachineStateFlags: Integer	cmd.Ptr["MachineStateFlags"].Bit[i], i=(0..31). Целое число, являющееся битовым полем из 32 бит, от 0 до 31. Если существует несколько вариантов расположения осей

TCLDGoHome: ComplexType	Команда возврата станка в исходное положение
	станка, обеспечивающих заданное положение инструмента в пространстве, каждый бит этого поля определяет выбор одного из возможных решений. Например, у пятиосевого станка с осями A и C для большинства положений инструмента есть два возможных решения - с положительным значением оси A и с отрицательным. Нулевой бит этого поля выбирает одно из этих решений. Для стандартного шестиосевого шарнирного робота многие положения инструмента в пространстве могут обеспечиваться разными сочетаниями положений трёх его ключевых шарниров (основания, локтя и запястья), что даёт в сумме 8 возможных решений. В этом случае нулевой бит этого поля определит положение основания, первый бит - положение локтя, а третий бит - положение запястья робота. Таким образом, значение каждого из этих битов полностью определяется используемой схемой станка. При использовании станков, которым эти флаги не требуются, вывод этого параметра в CLData можно отключить в схеме станка.
Time: Double	Время перемещения в минутах.

Список координат, передаваемых через CLData команды GOHOME, определяется схемой станка CAM-системы.

Используйте массив GMA для обработки команды GOHOME. Свойствам массива GMA автоматически присваиваются предварительно рассчитанные значения параметров перемещения по координатам. Параметры команды также доступны через оператор cmd.

Пример (из постпроцессора Fanuc (30i)_Mill):

```
program GoHome
  if CycleOn=1 then begin
```

```
    CycleOn = 0
    Cycle = 80
  end
  !OutBlock                 ! Output in block

  if Cmd.Ptr["Axes(AxisXPos)"]<>0 then begin
    X = 0; X@ = MaxReal
  end
  if Cmd.Ptr["Axes(AxisYPos)"]<>0 then begin
    Y = 0; Y@ = MaxReal
  end
  if Cmd.Ptr["Axes(AxisZPos)"]<>0 then begin
    Z = 0; Z@ = MaxReal
  end
  if Cmd.Ptr["Axes(AxisAPos)"]<>0 then begin
    A = 0; A@ = MaxReal
    HasAxisA = 1
  end
  if Cmd.Ptr["Axes(AxisBPos)"]<>0 then begin
    B = 0; B@ = MaxReal
    HasAxisB = 1
  end
  if Cmd.Ptr["Axes(AxisCPos)"]<>0 then begin
    C = 0; C@ = MaxReal
    HasAxisC = 1
  end
  if (X<>X@) or (Y<>Y@) or (Z<>Z@) or
     (A<>A@) or (B<>B@) or (C<>C@)
  then begin
    GFunc = 28; GFunc@ = MaxReal
    GAbsInc = 91
    OutBlock
    if NextCode<>CodeOfCmd("GoHome") then begin
      GAbsInc = 90
      OutBlock
    end
  end
end
```

Доступ из .NET

Обработчики:

```
public override void OnBeforeMovement(ICLDMotionCommand cmd, CLDArray cld)   //
group handler - before any motion command
```

```
public override void OnGoHome(ICLDGoHomeCommand cmd, CLDArray cld)
public override void OnAfterMovement(ICLDMotionCommand cmd, CLDArray cld)   // group
handler - after any motion command
```

Наследование: `ICLDGoHomeCommand : ICLDMultiMotionCommand : ICLDMotionCommand : ICLDCommand`.

Читайте параметры из строго типизированного объекта `cmd` (рекомендуется); доступны также «сырой» массив `cld[i]` (те же индексы, что и в sppx) и именованные обёртки `cmd["..."]`. В таблице перечислены члены команды, включая унаследованные от базовых интерфейсов; общие члены `ICLDCommand` (навигация, `CLD`, `CLDFile`, `TechOperation`, ...) описаны в разделе Модель доступа к CLData.

Член	Тип	Описание
`cmd.EN`	`TInpRotation`	Конечная нормаль или ориентация перемещения в декартовой системе координат. В зависимости от настроек текущей схемы станка это может быть нормальный вектор, углы Эйлера, кватернион и т. д.
`cmd.MSFlags`	`BitArray`	Флаги состояния станка или flips робота, которые вместе с декартовыми координатами помогают определить конкретное состояние механизма в случае избыточности. Какой бит отвечает за какой флаг, зависит от типа и конфигурации оборудования.
`cmd.Time`	`double`	Продолжительность перемещения в минутах.
`cmd.AxesCount`	`int`	Количество осей, которые перемещение включает в список Axis[].

Член	Тип	Описание
`cmd.Axis`	`ICLDMultiMotionAxesIndexer`	Список осей перемещения [0..(AxesCount-1)].
`cmd.Axes`	`IEnumerable`	Объект-перечислитель осей перемещения, позволяющий использовать удобный синтаксис foreach.
`cmd.HasAxis(...)`	`bool`	Возвращает "true", если перемещение содержит ось с указанным текстовым идентификатором. Иначе возвращает "false".
`cmd.EP`	`TInp3DPoint`	Конечная точка (X, Y, Z) перемещения

Пример - `OnGoHome` (из постпроцессора Fanuc (30i)_Mill_DN):

```
public override void OnGoHome(ICLDGoHomeCommand cmd, CLDArray cld)
{
    foreach(CLDMultiMotionAxis ax in cmd.Axes) {
        if (ax.IsX) {
            nc.U.Show(0);
        } else if (ax.IsY) {
            nc.V.Show(0);
        } else if (ax.IsZ) {
            nc.W.Show(0);
        }
    }
    if (nc.U.Changed || nc.V.Changed || nc.W.Changed) {
        nc.GHome.Show(28);
        nc.Block.Out();
    }
}
```

Сопутствующие ссылки

- Модель доступа к CLData

GOTO - Линейный переход

Отображение команды

Как команда отображается в списке команд CLData:

```
GOTO.abs X x, Y y, Z z
```

CAM-система передаёт данные линейного перехода командой ABSMOV (`GOTO.abs`). Параметрами команды являются новые абсолютные координаты по осям X, Y и Z.

Доступ из sppx

Обработчик: `program AbsMov`

Параметр	Массив CLD	Имя CLD	Описание
x y z	CLD[1] CLD[2] CLD[3]	CLD.X CLD.Y CLD.Z	Новые координаты инструмента (абсолютные)

Пример (из постпроцессора Fanuc (30i)_Mill):

```
program AbsMov
  if (INTERP_ > 1) and (INTERP_ <> 33) then
    INTERP_ = 1    ! G1
  X = cld[1]                          ! X,Y,Z в абсолютных значениях
  Y = cld[2]
  if IsRotaryInterpolation = 2 then begin
    Y = -Y/CylR/Pi*180
  end
  Z = cld[3]
  if (Z<>Z@) and (NeedKorDl<>0) then begin
    KorDl = NeedKorDl; KorDl@ = MaxReal
    H@ = MaxReal
    NeedKorDl = 0
  end
  if (CycleOn=0) and ((X<>X@) or (Y<>Y@) or (Z<>Z@)) then begin
    if KorEcv<>KorEcv@ then begin
      if Interp_<>Interp_@ then begin
        X@ = X
        Y@ = Y
        Z@ = Z
```

```
        KorEcv@ = KorEcv
        D@ = D
        OutBlock
        KorEcv@ = MaxReal
        D@ = MaxReal
      end
      X@ = MaxReal !XT_
      Y@ = MaxReal !YT_
      Z@ = MaxReal !ZT_
    end
    if IsMultiaxisInterpolation = 1 then begin
      X@ = MaxReal;
      Y@ = MaxReal;
      Z@ = MaxReal;
    end;
    OutBlock
  end
  XT_ = X;  YT_ = Y;  ZT_ = Z        ! сохранить текущие координаты
end
```

Доступ из .NET

Обработчики:

```
public override void OnBeforeMovement(ICLDMotionCommand cmd, CLDArray cld)   //
групповой обработчик - перед любой командой перемещения
public override void OnGoto(ICLDGotoCommand cmd, CLDArray cld)
public override void OnAfterMovement(ICLDMotionCommand cmd, CLDArray cld)   //
групповой обработчик - после любой команды перемещения
```

Наследование: `ICLDGotoCommand : ICLDMotionCommand : ICLDCommand`.

Читайте параметры из строго типизированного объекта `cmd` (рекомендуется); доступны также «сырой» массив `cld[i]` (те же индексы, что и в sppx) и именованные обёртки `cmd["..."]`. В таблице перечислены члены команды, включая унаследованные от базовых интерфейсов; общие члены `ICLDCommand` (навигация, `CLD`, `CLDFile`, `TechOperation`, ...) описаны в разделе Модель доступа к CLData.

Член	Тип	Описание
`cmd.EP`	`TInp3DPoint`	Конечная точка (X, Y, Z) перемещения

Пример - `OnGoto` (из постпроцессора Fanuc (30i)_Mill_DN):

```
public override void OnGoto(ICLDGotoCommand cmd, CLDArray cld)
{
    if (nc.GInterp > 1)
        nc.GInterp.v = 1;
    nc.X.v = cmd.EP.X;
    nc.Y.v = cmd.EP.Y;
    nc.Z.v = cmd.EP.Z;

    if (!cycleIsOn) {
        if (nc.Z.Changed && nc.GLCompens.ValuesDiffer) {
            nc.Block.Hide(nc.X, nc.Y);
            nc.Block.Show(nc.GLCompens, nc.HLCompens, nc.Z);
            nc.Block.Out();
            nc.Block.UpdateState(nc.X, nc.Y);
        }
        // Раскомментируйте, если нужны обязательные XYZABC в режиме TCPM
        // if (nc.GLCompens == 43.4) {
        //     nc.Block.Show(nc.X, nc.Y, nc.Z);
        //     if (CLDProject.Machine.HasAAxis)
        //         nc.A.Show();
        //     if (CLDProject.Machine.HasBAxis)
        //         nc.B.Show();
        //     if (CLDProject.Machine.HasCAxis)
        //         nc.C.Show();
        // }
        if(nc.X.Changed || nc.Y.Changed || nc.Z.Changed )
            nc.Block.Out();
    }
    nc.LastP = cmd.EP;
}
```

Сопутствующие ссылки

- Модель доступа к CLData

HEAD - Номер шпиндельной головки

> Устарела. Эта команда является устаревшей - текущая версия CAM-системы не генерирует её автоматически. Она документирована здесь для постпроцессоров, которые всё ещё обрабатывают устаревший CLData.

Отображение команды

Как команда отображается в списке команд CLData:

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

Команда HEAD используется для выбора активной шпиндельной головки. Номер головки передаётся в качестве параметра.

Доступ из sppx

Обработчик: `program Head`

Параметр	Массив CLD	Имя CLD	Описание
BOTH(83) n	CLD[1]	CLD.N	83 (BOTH) - одновременная работа обеих головок, n - номер головки.

Сопутствующие ссылки

- Модель доступа к CLData

INSERT - Непосредственный вывод в кадр

Отображение команды

Как команда отображается в списке команд CLData:

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

Команда INSERT используется для передачи текстовой информации непосредственно в управляющую программу. Текст помещается в переменную CLDATA$.

Доступ из sppx

Обработчик: `program Insert`

Пример (из постпроцессора Fanuc (30i)_Mill):

```
program Insert
  OutBlock
  FormBlock
  Output OutStr$ + CLData$
end
```

Доступ из .NET

Обработчик:

```
public override void OnInsert(ICLDInsertCommand cmd, CLDArray cld)
```

Наследование: `ICLDInsertCommand : ICLDCommand`.

Читайте параметры из строго типизированного объекта `cmd` (рекомендуется); доступны также «сырой» массив `cld[i]` (те же индексы, что и в sppx) и именованные обёртки `cmd["..."]`. В таблице перечислены члены команды, включая унаследованные от базовых интерфейсов; общие члены `ICLDCommand` (навигация, `CLD`, `CLDFile`, `TechOperation`, ...) описаны в разделе Модель доступа к CLData.

Член	Тип	Описание
`cmd.Text`	`string`	Текст для вывода в файл.

Пример - `OnInsert` (из постпроцессора Heidenhain_Mill_iTNC530_DN):

```
public override void OnInsert(ICLDInsertCommand cmd, CLDArray cld)
{
    nc.WriteLine(cmd.Text);
}
```

Сопутствующие ссылки

- Модель доступа к CLData

INTERPOLATION - Режим интерполяции

Отображение команды

Как команда отображается в списке команд CLData:

```
INTERPOLATION ON(71) | OFF(72), CARTESIAN(9020) | POLAR(9021) |
CYLINDRICAL(9022) | MULTIAXIS(9023), P1 p1, P2 p2, P3 p3
```

INTERPOLATION — эта команда активирует или деактивирует режимы интерполяции, такие как цилиндрическая или полярная интерполяция.

Режим интерполяции, заданный параметром `CLD[2]` (`CLD.Mode`), активируется, если параметр `CLD[1]` (`CLD.SubCmd`) равен `ON(71)`, иначе указанная интерполяция деактивируется.

Параметры `CLD[3]` (`CLD.P1`), `CLD[4]` (`CLD.P2`), `CLD[5]` (`CLD.P3`) задают дополнительные параметры режимов интерполяции.

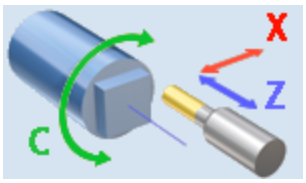
Доступ из sppx

Обработчик: `program Interpolation`

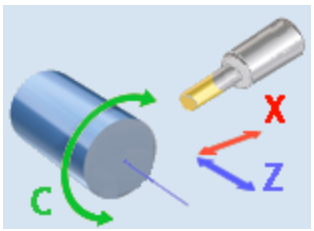
Параметр	Массив CLD	Имя CLD	Описание
ON(71) OFF(72)	CLD[1]	CLD.SubCmd	ON(71) – активировать, OFF(72) – деактивировать.
CARTESIAN(9020) POLAR(9021) CYLINDRICAL(9022) MULTIAXIS(9023)	CLD[2]	CLD.Mode	CARTESIAN(9020) – декартова интерполяция; POLAR(9021) – полярная интерполяция; CYLINDRICAL(9022) – цилиндрическая интерполяция; MULTIAXIS(9023) – многоосевая интерполяция

Параметр	Массив CLD	Имя CLD	Описание
P1	CLD[3]	CLD.P1	Задаёт радиус цилиндра для режима цилиндрической интерполяции
P2	CLD[4]	CLD.P2	Зарезервировано
P3	CLD[5]	CLD.P3	Зарезервировано

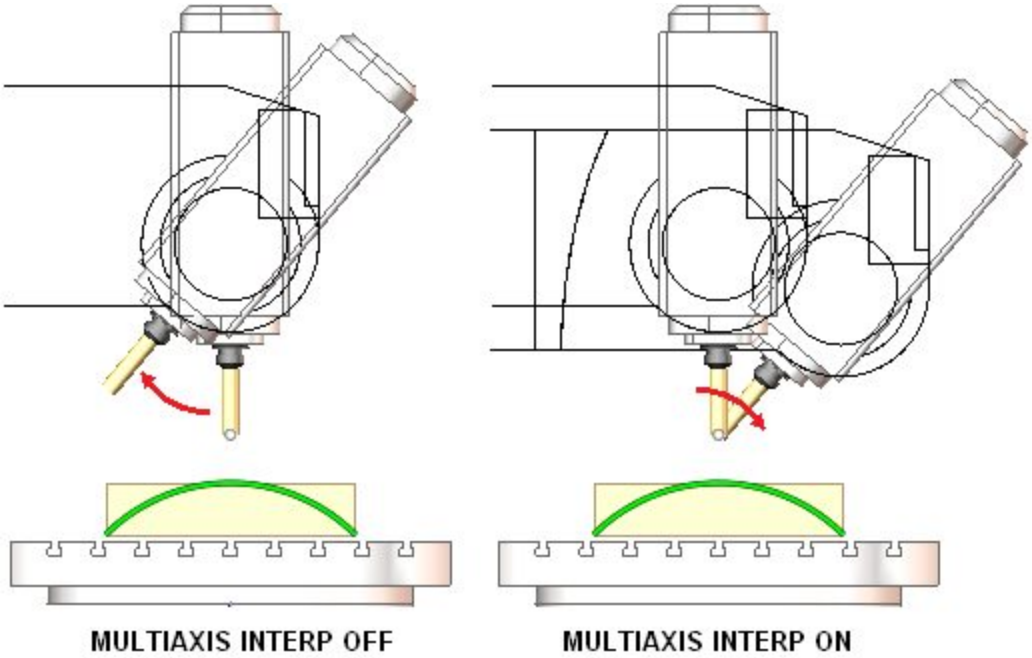
Режим полярной интерполяции обычно используется на токарно-фрезерных станках без оси Y для обработки деталей сбоку. В этом режиме перемещение инструмента в направлении Y заменяется одновременным поворотом заготовки вокруг поворотной оси (C) и перемещением инструмента вдоль радиальной оси (X). В зависимости от используемой системы ЧПУ координата Y в управляющей программе может заменяться координатой C.



Режим цилиндрической интерполяции используется для обработки цилиндрических карманов на цилиндрических деталях. Управляющая программа содержит 2D-контур кармана в координатах XY, где координата X — это координата вдоль оси цилиндра, а координата Y равна радиусу цилиндра, умноженному на угол поворота заготовки вокруг поворотной оси. В зависимости от используемой системы ЧПУ координата Y в управляющей программе может заменяться координатой C.



Режим многоосевой интерполяции (MULTIAXIS) используется для одновременного пятиосевого фрезерования. В этом режиме координаты XYZ кадра управляющей программы — это координаты центральной точки инструмента относительно системы координат заготовки, вращающейся вместе с поворотным столом. В отличие от стандартного режима декартовой интерполяции, в режиме многоосевой интерполяции, если кадр УП содержит только команды позиционирования поворотных осей, положение кончика инструмента относительно заготовки остаётся неизменным. Это означает, что инструмент поворачивается вокруг своей центральной точки (см. рисунок ниже).



Если запрограммировать перемещение наклоняемого стола при активной многоосевой интерполяции, система ЧПУ соответствующим образом поворачивает систему координат. Если, например, повернуть ось C на 90° (командой позиционирования или сдвигом базы) и затем запрограммировать перемещение по оси X, система ЧПУ выполнит это перемещение по оси станка Y.

Разные системы ЧПУ используют разные команды для включения и отключения режима многоосевой интерполяции. В Heidenhain этот режим называется управлением центральной точкой инструмента и управляется функциями TCPM, M128 и M129. В Sinumeric режим включается и выключается командами TRAORI и TRAFOOF.

Пример (из постпроцессора Fanuc (30i)_Mill):

```
program Interpolation
  case cld[2] of
    9023: begin ! 5-axis
      if CLD.SubCmd=71 then begin ! On
        NeedKorDl = 43.4
        IsMultiaxisInterpolation = 1
      end else begin              ! Off
        KorDl = 49;! KorDl@ = MaxReal
        IsMultiaxisInterpolation = 0
      end
      OutBlock
    end
    9021: begin ! POLAR
      OutBlock
      if CLD.SubCmd=71 then begin ! On
```

```
        GInterpRot = 12.1
        YAddress = "A"
        IsRotaryInterpolation = 1
      end else begin              ! Off
        GInterpRot = 13.1
        YAddress = "Y"
        IsRotaryInterpolation = 0
      end
      GInterpRot@ = MaxReal
      OutBlock
    end
    9022: begin ! CYLINDRICAL
      OutBlock
      if CLD.SubCmd=71 then begin ! On
        GInterpRot = 07.1
        A = CLD.P1; A@ = MaxReal
        YAddress = "A"
        IsRotaryInterpolation = 2
        CylR = CLD.P1
        CirclesThroughRadius = 1
      end else begin              ! Off
        GInterpRot = 07.1
        A = 0; A@ = MaxReal
        YAddress = "Y"
        IsRotaryInterpolation = 0
        CirclesThroughRadius = 0
      end
      GInterpRot@ = MaxReal
      OutBlock
    end
  end
end
```

Доступ из .NET

Обработчики:

```
public override void OnInterpolation(ICLDInterpolationCommand cmd, CLDArray cld)
public override void OnInterpPolar(ICLDInterpolationCommand cmd, CLDArray cld)
// POLAR(9021)
public override void OnInterpCylindrical(ICLDInterpolationCommand cmd, CLDArray cld)
// CYLINDRICAL(9022)
public override void OnInterp5x(ICLDInterpolationCommand cmd, CLDArray cld)
// MULTIAXIS(9023)
```

Наследование: `ICLDInterpolationCommand : ICLDCommand`.

Читайте параметры из строго типизированного объекта `cmd` (рекомендуется); доступны также «сырой» массив `cld[i]` (те же индексы, что и в sppx) и именованные обёртки `cmd["..."]`. В таблице перечислены члены команды, включая унаследованные от базовых интерфейсов; общие члены `ICLDCommand` (навигация, `CLD`, `CLDFile`, `TechOperation`, ...) описаны в разделе Модель доступа к CLData.

Член	Тип	Описание
`cmd.IsOn`	`bool`	Возвращает "True", если это команда включения интерполяции, иначе возвращает "False".
`cmd.IsOff`	`bool`	Возвращает "True", если это команда выключения интерполяции, иначе возвращает "False".
`cmd.InterpMode`	`CLDInterpMode`	Определяет тип интерполяции (полярная - G112, цилиндрическая - G107 или непрерывная многоосевая - G43.4, TCPM, TRAORI).
`cmd.InterpType`	`int`	Числовой код, определяющий тип интерполяции (полярная, цилиндрическая или непрерывная многоосевая).
`cmd.CylRadius`	`double`	Задаёт радиус цилиндра для режима цилиндрической интерполяции.

Пример - `OnInterpolation` (из постпроцессора Sinumerik (840D)_Mill_DN):

```
public override void OnInterpolation(ICLDInterpolationCommand cmd, CLDArray cld)
{
    if (cmd.InterpType == 9023){// MULTIAXIS interpolation
```

```
        if (cmd.IsOn){ // Switch on
            nc.Block.Out();
            nc.WriteLineWithBlockN($"TRAORI");
            nc.CoordSys.v0 = double.MaxValue; nc.Block.Out();
            nc.WriteLineWithBlockN($"ORIWKS");
            nc.WriteLineWithBlockN($"ORIAXES");
        }else{          // Switch off
            nc.Block.Out();
            nc.WriteLineWithBlockN($"TRAFOOF");
        }
    }else if (cmd.InterpType == 9021) {// Polar interpolation
        if (cmd.IsOn) { // Switch on
            nc.Block.Out();
            nc.WriteLine("TRANSMIT");
            Cycle.SetPolarInterpolationStatus(true);
            csC = Cos(nc.A.v);
            snC = -Sin(nc.A.v);
        }else {          // Switch off
            nc.Block.Out();
            nc.WriteLine("TRANSMIT");
            Cycle.SetPolarInterpolationStatus(true);
            nc.A.RestoreDefaultValue(false);
        }
    }else if (cmd.InterpType == 9022) {// Cylindrical interpolation
        if (cmd.IsOn) { // Switch on
            nc.Block.Out();
            nc.WriteLine("TRACYL(" + Str(2*cld[3]) + ")");
            Cycle.SetCilindInterpolationStatus(true);
        }else{             // Switch off
            nc.Block.Out();
            nc.WriteLine("TRAFOOF");
            //Output "TMCOFF"
            Cycle.SetCilindInterpolationStatus(false);
            nc.A.RestoreDefaultValue(false);
        }
    }
}
```

Сопутствующие ссылки

- Модель доступа к CLData

LOADTL - Загрузка инструмента

Отображение команды

Как команда отображается в списке команд CLData:

```
LOADTL N n, Tool changed 1, X x, Y y, Z z, D d, M m, K k, L l, P p, A a, R r, H h,
RC rc,
       PLANE XY(33)|YZ(37)|XZ(41)|YX(133)|ZY(137)|ZX(141),
       DUR d2, HID hid, NX nx, NY ny, NZ nz, NW nw, RevolverID rid
```

Команда LOADTL загружает инструмент с номером, указанным в первом параметре. Используйте команду LOADTL для формирования кадров управляющей программы, перемещающих револьверную головку или выбирающих инструмент в магазине инструментов. Также эта команда используется для формирования кадров управляющей программы, отводящих инструмент в безопасную позицию для смены инструмента.

Доступ из sppx

Обработчик: `program Loadtl`

Массив `cld` читается по индексу или по имени - в sppx по типизированному короткому имени (например, `cld.X`), в .NET по тому же имени как строковому ключу (например, `cld["X"]`).

Параметр	Массив CLD	Имя CLD	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 X, Y, Z относительно точки закрепления инструмента (система координат фланца).
d	CLD[5]	CLD.D	Диаметр инструмента.
m	CLD[6]	CLD.M	Номер корректора на длину. Если настроечная точка -

Параметр	Массив CLD	Имя CLD	Описание
			кончик инструмента, то M < 0.
k	CLD[7]	CLD.K	Номер корректора на радиус. Если настроечная точка - кончик инструмента, то K < 0.
L	CLD[8]	CLD.L	Длина 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.
h	CLD[12]	CLD.H	Высота инструмента.
rc	CLD[13]	CLD.Rc	Радиус цилиндрической части осевого инструмента.
XY(33) YZ(37) XZ(41) YX(133) ZY(137) ZX(141)	CLD[14]		Плоскость инструмента. Плоскость, перпендикулярная оси инструмента для осевого инструмента. Плоскость обработки для токарного инструмента. 33 - XY,

Параметр	Массив CLD	Имя CLD	Описание
			37 - YZ, 41 - XZ, 133 - YX, 137 - ZY, 141 - ZX.
	CLD[15] CLD[16]		Зарезервировано.
d2	CLD[17]	CLD.Dur	Стойкость инструмента (в минутах). 0 - не учитывается.
HID	CLD[18]	CLD.HID	Идентификатор (номер) держателя инструмента
nx ny nz nw	CLD[19] CLD[20] CLD[21] CLD[22]	CLD.NX CLD.NY CLD.NZ CLD.NW	Значения, определяющие ориентацию инструмента относительно точки закрепления (системы координат фланца). В зависимости от настроек системы (задаются в схеме станка) это могут быть либо углы поворота вокруг осей координат X, Y и Z в градусах (углы Эйлера A = NX, B = NY, C = NZ), либо коэффициенты кватерниона q = (NX, NY, NZ, NW).

Доступ по имени параметра (помощники)

Параметры, доступные через оператор `cmd` (одинаково в sppx и .NET):

TCLDLoadTool: ComplexType	Команда загрузки инструмента
ToolChanged: bool (integer)	cmd.Int["ToolChanged"] - ToolChanged - это флаг, определяющий, производится ли в данном месте смена инструмента относительно предыдущей операции. Если он равен 1 (true), это означает, что инструмент в этой операции отличается от инструмента предыдущей операции. Если флаг равен 0 (false), это означает, что инструменты в текущей и предыдущей операциях одинаковы и смены инструмента нет. От этого флага зависит опция «Пропускать LoadTL, если нет смены инструмента» в окне настроек, и постпроцессор может пропускать вызов процедуры LoadTL.
ToolID: Integer	cmd.Int["ToolID"] - Идентификатор (номер) инструмента
RadCorrNum: Integer	cmd.Int["RadCorrNum"] - Номер корректора на радиус
LenCorrNum: Integer	cmd.Int["LenCorrNum"] - Номер корректора на длину
HolderID: Integer	cmd.Int["HolderID"] - Идентификатор (номер) держателя инструмента
RevolverID: String	cmd.Str["RevolverID"] - Строковый идентификатор револьверной головки

Пример (из постпроцессора Fanuc (30i)_Mill):

```
program LoadTl
  OutBlock

  H = abs(cld[6]); H@ = H

  GFunc = 53; GFunc@ = MaxReal
  Z = 0; Z@ = MaxReal
```

```
  OutBlock
  GFunc = 53; GFunc@ = MaxReal
  X = 0; X@ = MaxReal
  Y = 0; Y@ = MaxReal
  OutBlock

  M = 06; M@ = M-1
  Tool = cld[1]; Tool@ = MaxReal    ! заполнение регистра Tool
  FormBlock
  if NextCode=CodeOfCmd("Comment") then begin
    s$ = CLDFile[CLDFile.CurrentFile].Cmd[CLDFile.CurrentCmd+1].Data
    call TransStr(s$)
    OutStr$ = OutStr$ + " (" + s$ + ")"
  end
  Output OutStr$
  WasLoadTool = 1

  CoordSys = WCSNumber
  CoordSys@ = MaxReal
  OutBlock

  X = MaxReal; X@ = X
  Y = MaxReal; Y@ = Y
  Z = MaxReal; Z@ = Z
!  A = MaxReal; A@ = A
!  B = MaxReal; B@ = B
!  C = MaxReal; C@ = C
end
```

Доступ из .NET

Обработчик:

```
public override void OnLoadTool(ICLDLoadToolCommand cmd, CLDArray cld)
```

Наследование: `ICLDLoadToolCommand : ICLDCommand`.

Читайте параметры из строго типизированного объекта `cmd` (рекомендуется); также доступны «сырой» массив `cld[i]` (те же индексы, что и в sppx) и именованные помощники `cmd[...]`. В таблице перечислены члены команды, включая унаследованные от базовых интерфейсов; общие члены `ICLDCommand` (навигация, `CLD`, `CLDFile`, `TechOperation`, ...) описаны в разделе Модель доступа к CLData.

Свойство	Тип	Описание
`cmd.ToolChanged`	bool	True, если инструмент изменился по сравнению с предыдущей операцией; false, если операция использует тот же инструмент.
`cmd.Number`	int	Номер инструмента.
`cmd.LCorNum`	int	Первый номер корректора (на длину) инструмента.
`cmd.RCorNum`	int	Второй номер корректора (на радиус) инструмента.
`cmd.ConnectorIndex`	int	Индекс коннектора инструмента в схеме станка.
`cmd.RevolverID`	string	Идентификатор револьверной головки для станков с револьверной головкой; идентификатор коннектора инструмента для других типов станков.
`cmd.Overhang`	`TInpLocation`	Вылет кончика инструмента (X, Y, Z + углы ориентации) относительно шпинделя; тип ориентации зависит от схемы станка (углы Эйлера, вектор нормали или кватернион).
`cmd.Plane`	`CLDPlaneType`	Плоскость ориентации инструмента.
`cmd.Durability`	double	Стойкость инструмента в минутах.
`cmd.Geom`	`IToolGeometry`	Основные геометрические параметры инструмента: `D`

Свойство	Тип	Описание
		(диаметр резания), `R` (радиус скругления вершины), `Rc` (радиус скругления у цилиндрической части), `L` (длина), `P` (ширина), `A` (угол), `H` (высота фасонной части).

Пример (из постпроцессора Sinumerik 840D mill):

```
public override void OnLoadTool(ICLDLoadToolCommand cmd, CLDArray cld)
{
    nc.Tool.Show(cmd.Number);
    nc.DTool.Reset(Abs(cmd.LCorNum));
    nc.OutText("M6");
}
```

Сопутствующие ссылки

- Модель доступа к CLData

MULTIARC - Многокоординатное перемещение по окружности

Отображение команды

Как команда отображается в списке команд CLData:

```
MULTIARC EPX ep.X, EPY ep.Y, EPZ ep.Z, EPRX ep.Rx, EPRY ep.Ry, EPRZ ep.Rz,
EPRW ep.Rw,
         MPX mp.X, MPY mp.Y, MPZ mp.Z, MPRX mp.Rx, MPRY mp.Ry, MPRZ mp.Rz,
MPRW mp.Rw
```

Команда перемещения MULTIARC задаёт дугу окружности, проходящую через три точки, с заданной ориентацией инструмента в пространстве для каждой из этих точек. На рисунке ниже SP - начальное положение, в котором находился инструмент перед выполнением команды, MP - промежуточное положение инструмента на дуге, EP - конечное положение инструмента.

Доступ из sppx

Обработчик: `program MultiArc`

Массив `cld` читается по индексу или по имени - в sppx по типизированному короткому имени (например, `cld.X`), в .NET по тому же имени как строковому ключу (например, `cld["X"]`).

Параметр	Массив CLD	Имя CLD	Описание
ep.X ep.Y ep.Z	CLD[1] CLD[2] CLD[3]	CLD.EPX CLD.EPY CLD.EPZ	Декартовы координаты X, Y и Z конечной точки дуги
ep.Rx ep.Ry ep.Rz ep.Rw	CLD[4] CLD[5] CLD[6] CLD[7]	CLD.EPNX CLD.EPNY CLD.EPNZ CLD.EPNW	Параметры, задающие ориентацию инструмента в конечной точке дуги относительно текущей системы координат (поля RX, RY, RZ и RW). В зависимости от

Параметр	Массив CLD	Имя CLD	Описание
			конфигурации системы (заданной в схеме станка) ориентация инструмента может задаваться: вектором нормали n = (RX, RY, RZ), пространственными углами поворота вокруг осей координат - углами Эйлера (порядок поворотов также задаётся в настройках) A = RX, B = RY, C = RZ, кватернионом q = (RX, RY, RZ, RW).
mp.X mp.Y mp.Z	CLD[8] CLD[9] CLD[10]	CLD.MPX CLD.MPY CLD.MPZ	Декартовы координаты X, Y и Z промежуточной точки дуги
mp.Rx mp.Ry mp.Rz mp.Rw	CLD[11] CLD[12] CLD[13] CLD[14]	CLD.MPNX CLD.MPNY CLD.MPNZ CLD.MPNW	Параметры, задающие ориентацию инструмента в промежуточной точке дуги относительно текущей системы координат (поля RX, RY, RZ и RW). В зависимости от конфигурации системы (заданной в схеме станка)

Параметр	Массив CLD	Имя CLD	Описание
			ориентация инструмента может задаваться: вектором нормали n = (RX, RY, RZ), пространственными углами поворота вокруг осей координат - углами Эйлера (порядок поворотов также задаётся в настройках) A = RX, B = RY, C = RZ, кватернионом q = (RX, RY, RZ, RW).

Доступ по имени параметра (вспомогательные обёртки)

Параметры, доступные через оператор `cmd` (одинаковый в sppx и .NET):

TCLDMultiArc: ComplexType	Команда перемещения по дуге окружности, заданной тремя точками в пространстве.
EndPos: ComplexType	cmd.Ptr["EndPos"] - структура, поля которой определяют положение инструмента в конечной точке дуги.
Axes: Array, Key="AxisID"	cmd.Ptr["EndPos.Axes"] - массив структур типа Axis. Одна команда может включать перемещение по нескольким осям, положение каждой из них хранится в этом массиве.
Axis: ComplexType	cmd.Ptr["EndPos.Axes"].Item[Index] или cmd.Ptr["EndPos.Axes(AxisName)"] - отдельный элемент массива Axes. Содержит информацию о перемещении по одной оси станка. Доступ к элементам массива возможен как по индексу, так и по

TCLDMultiArc: ComplexType	Команда перемещения по дуге окружности, заданной тремя точками в пространстве.
	ключевому полю. Здесь AxisName - значение ключевого поля, которое должно совпадать со значением поля AxisID.
AxisID: String	cmd.Str["EndPos.Axes(AxisName).AxisID"] - идентификатор оси станка, для которой задано новое положение. Определяется схемой станка.
Value: Double	cmd.Flt["EndPos.Axes(AxisName).Value"] - новое положение оси станка, в которое она перемещается.
Pos5D: ComplexType	cmd.Ptr["EndPos.Pos5D"] - структура, содержащая координаты, определяющие конечное положение инструмента в пространстве. Определяет не только положение точки настройки инструмента (поля X, Y и Z - декартовы координаты), но и ориентацию инструмента относительно текущей системы координат (поля NX, NY, NZ и NW). В зависимости от конфигурации системы (заданной в схеме станка) ориентация инструмента может задаваться: вектором нормали n = (NX, NY, NZ), пространственными углами поворота вокруг осей координат - углами Эйлера (порядок поворотов также задаётся в настройках) A = NX, B = NY, C = NZ, кватернионом q = (NX, NY, NZ, NW).
X: Double	cmd.Flt["EndPos.Pos5D.X"] - декартова координата X положения точки инструмента.
Y: Double	cmd.Flt["EndPos.Pos5D.Y"] - декартова координата Y положения точки инструмента.
Z: Double	cmd.Flt["EndPos.Pos5D.Z"] - декартова координата Z положения точки инструмента.

TCLDMultiArc: ComplexType	Команда перемещения по дуге окружности, заданной тремя точками в пространстве.
NX: Double	cmd.Flt["EndPos.Pos5D.NX"] - в зависимости от конфигурации системы может быть компонентом NX нормального вектора инструмента, углом поворота вокруг одной из осей координат либо коэффициентом кватерниона.
NY: Double	cmd.Flt["EndPos.Pos5D.NY"] - в зависимости от конфигурации системы может быть компонентом NY нормального вектора инструмента, углом поворота вокруг одной из осей координат либо коэффициентом кватерниона.
NZ: Double	cmd.Flt["EndPos.Pos5D.NZ"] - в зависимости от конфигурации системы может быть компонентом NZ нормального вектора инструмента, углом поворота вокруг одной из осей координат либо коэффициентом кватерниона.
NW: Double	cmd.Flt["EndPos.Pos5D.NW"] - если выбран способ ориентации инструмента в виде кватерниона, это четвёртый коэффициент кватерниона.
MachineStateFlags: Integer	cmd.Ptr["EndPos.MachineStateFlags"].Bit[i], i= (0..31). Целое число, являющееся битовым полем из 32 бит, от 0 до 31. Если существует несколько вариантов расположения осей станка, обеспечивающих заданное положение инструмента в пространстве, каждый бит этого поля определяет выбор одного из возможных решений. Например, у пятиосевого станка с осями A и C для большинства положений инструмента есть два возможных решения - с положительным значением оси A и с отрицательным.

TCLDMultiArc: ComplexType	Команда перемещения по дуге окружности, заданной тремя точками в пространстве.
	Нулевой бит этого поля выбирает одно из этих решений. Для стандартного шестиосевого шарнирного робота многие положения инструмента в пространстве могут обеспечиваться разными сочетаниями положений трёх его ключевых шарниров (основания, локтя и запястья), что даёт в сумме 8 возможных решений. В этом случае нулевой бит этого поля определит положение основания, первый бит - положение локтя, а третий бит - положение запястья робота. Таким образом, значение каждого из этих битов полностью определяется используемой схемой станка. При использовании станков, которым эти флаги не требуются, вывод этого параметра в CLData можно отключить в схеме станка.
MidPos: ComplexType	cmd.Ptr["MidPos"] - структура, поля которой определяют положение инструмента в промежуточной точке дуги.
Axes: Array, Key="AxisID"	cmd.Ptr["MidPos.Axes"] - массив структур типа Axis. Одна команда может включать перемещение по нескольким осям, положение каждой из них хранится в этом массиве.
Axis: ComplexType	cmd.Ptr["MidPos.Axes"].Item[Index] или cmd.Ptr["MidPos.Axes(AxisName)"] - отдельный элемент массива Axes. Содержит информацию о перемещении по одной оси станка. Доступ к элементам массива возможен как по индексу, так и по ключевому полю. Здесь AxisName - значение ключевого поля, которое должно совпадать со значением поля AxisID.

TCLDMultiArc: ComplexType	Команда перемещения по дуге окружности, заданной тремя точками в пространстве.
AxisID: String	cmd.Str["MidPos.Axes(AxisName).AxisID"] - идентификатор оси станка, для которой задано новое положение. Определяется схемой станка.
Value: Double	cmd.Flt["MidPos.Axes(AxisName).Value"] - новое положение оси станка, в которое она перемещается.
Pos5D: ComplexType	cmd.Ptr["MidPos.Pos5D"] - структура, содержащая координаты, определяющие конечное положение инструмента в пространстве. Определяет не только положение точки настройки инструмента (поля X, Y и Z - декартовы координаты), но и ориентацию инструмента относительно текущей системы координат (поля NX, NY, NZ и NW). В зависимости от конфигурации системы (заданной в схеме станка) ориентация инструмента может задаваться: вектором нормали n = (NX, NY, NZ), пространственными углами поворота вокруг осей координат - углами Эйлера (порядок поворотов также задаётся в настройках) A = NX, B = NY, C = NZ, кватернионом q = (NX, NY, NZ, NW).
X: Double	cmd.Flt["MidPos.Pos5D.X"] - декартова координата X положения точки инструмента.
Y: Double	cmd.Flt["MidPos.Pos5D.Y"] - декартова координата Y положения точки инструмента.
Z: Double	cmd.Flt["MidPos.Pos5D.Z"] - декартова координата Z положения точки инструмента.
NX: Double	cmd.Flt["MidPos.Pos5D.NX"] - в зависимости от конфигурации системы может быть компонентом NX нормального вектора

TCLDMultiArc: ComplexType	Команда перемещения по дуге окружности, заданной тремя точками в пространстве.
	инструмента, углом поворота вокруг одной из осей координат либо коэффициентом кватерниона.
NY: Double	cmd.Flt["MidPos.Pos5D.NY"] - в зависимости от конфигурации системы может быть компонентом NY нормального вектора инструмента, углом поворота вокруг одной из осей координат либо коэффициентом кватерниона.
NZ: Double	cmd.Flt["MidPos.Pos5D.NZ"] - в зависимости от конфигурации системы может быть компонентом NZ нормального вектора инструмента, углом поворота вокруг одной из осей координат либо коэффициентом кватерниона.
NW: Double	cmd.Flt["MidPos.Pos5D.NW"] - если выбран способ ориентации инструмента в виде кватерниона, это четвёртый коэффициент кватерниона.
MachineStateFlags: Integer	cmd.Ptr["MidPos.MachineStateFlags"].Bit[i], i= (0..31). Целое число, являющееся битовым полем из 32 бит, от 0 до 31. Если существует несколько вариантов расположения осей станка, обеспечивающих заданное положение инструмента в пространстве, каждый бит этого поля определяет выбор одного из возможных решений. Например, у пятиосевого станка с осями A и C для большинства положений инструмента есть два возможных решения - с положительным значением оси A и с отрицательным. Нулевой бит этого поля выбирает одно из этих решений. Для стандартного шестиосевого шарнирного робота многие положения инструмента в пространстве

TCLDMultiArc: ComplexType	Команда перемещения по дуге окружности, заданной тремя точками в пространстве.
	могут обеспечиваться разными сочетаниями положений трёх его ключевых шарниров (основания, локтя и запястья), что даёт в сумме 8 возможных решений. В этом случае нулевой бит этого поля определит положение основания, первый бит - положение локтя, а третий бит - положение запястья робота. Таким образом, значение каждого из этих битов полностью определяется используемой схемой станка. При использовании станков, которым эти флаги не требуются, вывод этого параметра в CLData можно отключить в схеме станка.
Time: Double	Время перемещения в минутах.

Список координат, передаваемых через CLData, определяется схемой станка CAM-системы.

Пример (из постпроцессора робота Newker):

```
program MultiArc

 !========================== MidPoint
=============================================================
  I = CLD.MPX;  J = CLD.MPY;  K = CLD.MPZ
  I@ = MaxReal; J@ = MaxReal; K@ = MaxReal

!========================== EndPoint
=============================================================
  X = CLD.EPX;  Y = CLD.EPY;  Z = CLD.EPZ
  A = CLD.EPNX; B = CLD.EPNY; C = CLD.EPNZ;
  X@=MaxReal; Y@=MaxReal; Z@=MaxReal
  A@=MaxReal; B@=MaxReal; C@=MaxReal;
!=============================================================================================
==
  call A1_A6(19, CLD.EndAxesCount)
  A1@=maxreal; A2@=maxreal; A3@=maxreal; A4@=maxreal; A5@=maxreal; A6@=maxreal;

  Formblock
  output "{"+outstr$+"}"
```

```
    Speed@=Maxreal
    Formblock

    LineN = LineN + 1; LineN@ = LineN
    output "N"+str(LineN)+" MOVC " + outstr$ + " PL=1.0" + ";Arc"

end
```

Доступ из .NET

Обработчики:

```
public override void OnBeforeMovement(ICLDMotionCommand cmd, CLDArray cld)   //
group handler - before any motion command
public override void OnMultiArc(ICLDMultiArcCommand cmd, CLDArray cld)
public override void OnAfterMovement(ICLDMotionCommand cmd, CLDArray cld)    //
group handler - after any motion command
```

Наследование: `ICLDMultiArcCommand : ICLDMotionCommand : ICLDCommand`.

Читайте параметры из строго типизированного объекта `cmd` (рекомендуется); доступны также «сырой» массив `cld[i]` (те же индексы, что и в sppx) и именованные обёртки `cmd["..."]`. В таблице перечислены члены команды, включая унаследованные от базовых интерфейсов; общие члены `ICLDCommand` (навигация, `CLD`, `CLDFile`, `TechOperation`, ...) описаны в разделе Модель доступа к CLData.

Свойство	Тип	Описание
`cmd.EndP`	`ICLDMultiArcPoint`	Конечная точка дуги.
`cmd.MidP`	`ICLDMultiArcPoint`	Промежуточная точка дуги.
`cmd.Time`	double	Продолжительность перемещения в минутах.
`cmd.Center`	`TInp5DPoint`	Центральная точка и ориентация (углы Эйлера или кватернион) дуги.
`cmd.R`	double	Радиус дуги; знак определяет направление (по часовой стрелке или против часовой стрелки

Свойство	Тип	Описание
		относительно нормали плоскости дуги).
`cmd.Ang`	double	Угол дуги (всегда положительный).
`cmd.EP`	`TInp3DPoint`	Конечная точка (X, Y, Z). *(унаследовано от `ICLDMotionCommand`)*

Точка дуги (`ICLDMultiArcPoint`) имеет поля `P` (декартовы координаты X, Y, Z), `N` (ориентация инструмента: нормальный вектор, углы Эйлера или кватернион в зависимости от схемы станка), `MSFlags` (флаги состояния станка), `AxesCount`, `Axis` (список значений физических осей станка в этой точке), `Axes` (перечислитель) и `HasAxis(string ID)`.

Пример (из постпроцессора робота Fanuc - все команды перемещения используют общий метод `Motion`):

```
public override void OnMultiArc(ICLDMultiArcCommand cmd, CLDArray cld)
{
    Motion(cmd);
}

/// <summary>All movement commands call this method (MultiGOTO, PhysicGoto,
MultiArc, From)</summary>
void Motion(ICLDMotionCommand movement)
{
    state.prevPos = state.Pos;
    if (movement.CmdType==CLDCmdType.MultiArc) {
        ICLDMultiArcCommand cmd = movement as ICLDMultiArcCommand;
        state.Pos = new TInpLocation(cmd.MidP.P, cmd.MidP.N);
        state.J.FillJoints(cmd.MidP.Axes);
        state.Flips.Fill(cmd.Int["MidPos.MachineStateFlags"], state.J);
        int midPointIndex = ls.AddPoint(false, state);
        state.middlePos = state.Pos;

        state.Pos = new TInpLocation(cmd.EndP.P, cmd.EndP.N);
        state.J.FillJoints(cmd.EndP.Axes);
        state.Flips.Fill(cmd.Int["EndPos.MachineStateFlags"], state.J);
        int endPointIndex = ls.AddPoint(false, state);

        CalculateCNT(movement);
        AddMotionLine("C P[" + midPointIndex + "] P[" + endPointIndex + "] " +
```

```
Round(state.velocity/60) +
            " mm/sec " + FineOrCnt(state.cntValue) + " ACC" + state.accValue + "
;", false);
    } else {
        ICLDMultiMotionCommand cmd = movement as ICLDMultiMotionCommand;
        state.Pos = new TInpLocation(cmd.EP, cmd.EN);
        state.J.FillJoints(cmd.Axes);
        state.Flips.Fill(cmd.Int["MachineStateFlags"], state.J);
        if (movement.CmdType==CLDCmdType.From) {
            // Output nothing, just remember
            return;
        } else if (movement.CmdType==CLDCmdType.MultiGoto) {
            // Lines
            CalculateCNT(movement);
            int pointIndex = ls.AddPoint(false, state);
            AddMotionLine("L P[" + pointIndex + "] " + Round(state.velocity/60) + "
mm/sec " + FineOrCnt(state.cntValue) + " ACC" + state.accValue + " ;");
        } else { //PhysicGoto or GoHome
            // Joints
            state.cntValue = smooth.cntValueRapid;
            int pointIndex = ls.AddPoint(true, state);
            if (!state.J.ThereAreE || (state.J.ExtAxesGroup==2)) {
                AddMotionLine("J P[" + pointIndex + "] 10% " +
FineOrCnt(state.cntValue) + " ;");
            } else {
                AddMotionLine("J P[" + pointIndex + "] 10% " +
FineOrCnt(state.cntValue) + " ACC30;");
            }
        }
    }
}
```

Сопутствующие ссылки

- Модель доступа к CLData

MULTIGOTO - Многокоординатное перемещение

Отображение команды

Как команда отображается в списке команд CLData:

```
MULTIGOTO COUNT 2, MACHINE, X 34.725, Y 32, GEOM X 34.725, Y 32, Z 194, NX 0, NY 0,
NZ 1, NW 0, MSF 0, Time 0.003
```

MULTIGOTO — универсальная технологическая команда, которая используется для передачи перемещения по координатам станка. Команда MULTIGOTO используется для передачи постпроцессору переменного числа координат (также называемых «осями»). Этими координатами могут быть линейное перемещение инструмента или стола, поворот стола или инструмента либо перемещение по другим координатам станка. Также MULTIGOTO используется для передачи различных технологических команд, специфичных для конкретного станка, которые нельзя передать более специализированными командами. Команда MULTIGOTO может использоваться для управления обычными осями линейного перемещения X, Y, Z, поворотными осями A, B, C или для управления такими «осями», как кулачки патрона, тормоз шпинделя, перемещение люнета и т. д. Список координат станка определяется кинематической схемой.

Доступ из sppx

Обработчик: `program MultiGOTO`

Массив `cld` читается по индексу или по имени — в sppx по типизированному короткому имени (например, `cld.X`), в .NET по тому же имени в виде строкового ключа (например, `cld["X"]`).

Параметр	Массив CLD	Имя CLD	Описание
X Y Z	CLD[1] CLD[2] CLD[3]	CLD.X CLD.Y CLD.Z	Пространственные координаты вершины инструмента относительно активной системы координат заготовки
NX NY	CLD[4] CLD[5]	CLD.NX CLD.NY	Ориентация оси инструмента относительно

Параметр	Массив CLD	Имя CLD	Описание
NZ NW	CLD[6] CLD[7]	CLD.NZ CLD.NW	активной системы координат заготовки. Способ задания ориентации зависит от настройки схемы станка. Это могут быть углы Эйлера в разных соглашениях, вектор нормали инструмента, кватернион или повёрнутый вектор.
MSF	CLD[8]	CLD.MSF	Флаги состояния станка (флипы). Актуально для роботов и станков, которые могут иметь несколько конфигураций физических осей для одного и того же положения в пространстве. См. описание в следующей таблице.
Time	CLD[9]	CLD.Time	Время перемещения.
AxesCount	CLD[10]	CLD.AxesCount	Число физических координат станка, задействованных в этой команде.
r1	CLD[11]		Индекс оси с именем Axis1Pos в списке координат.
Val1	CLD[12]		Значение координаты станка

Параметр	Массив CLD	Имя CLD	Описание
			Axis1Pos.
r2	CLD[13]		Индекс оси с именем Axis2Pos в списке координат.
Val2	CLD[14]		Значение координаты станка Axis2Pos.
...	...		...
rN	CLD[2*N+9]		Индекс оси с именем AxisNPos в списке координат.
ValN	CLD[2*N+10]		Значение координаты станка AxisNPos.

Доступ по имени параметра (помощники)

Параметры, доступные через оператор `cmd` (одинаково в sppx и .NET):

TCLDMultiGoto: ComplexType	Команда многокоординатного перемещения
Axes: Array, Key="AxisID"	cmd.Ptr["Axes"] - Массив структур типа Axis. Одна команда может включать перемещение по нескольким осям, положение каждой хранится в этом массиве.
Axis: ComplexType	cmd.Ptr["Axes"].Item[Index] или cmd.Ptr["Axes(AxisName)"] - Отдельный элемент массива Axes. Содержит информацию о перемещении по одной оси станка. Доступ к элементам массива возможен как по индексу, так и по ключевому полю. Здесь AxisName — значение ключевого поля, которое должно совпадать со значением поля AxisID.

TCLDMultiGoto: ComplexType	Команда многокоординатного перемещения
AxisID: String	cmd.Str["Axes(AxisName).AxisID"] - Идентификатор оси станка, для которой задано новое положение. Определяется схемой станка.
Value: Double	cmd.Flt["Axes(AxisName).Value"] - Новое положение оси станка, в которое она перемещается.
Pos5D: ComplexType	cmd.Ptr["Pos5D"] - структура, содержащая координаты, определяющие конечное положение инструмента в пространстве. Определяет не только положение настроечной точки инструмента (поля X, Y и Z — декартовы координаты), но и ориентацию инструмента относительно текущей системы координат (поля NX, NY, NZ и NW). В зависимости от конфигурации системы (заданной в схеме станка) ориентация инструмента может задаваться: вектором нормали n = (NX, NY, NZ), пространственными углами поворота вокруг координатных осей — углами Эйлера (форма записи углов Эйлера, то есть порядок поворотов, также задаётся настройками) A = NX, B = NY, C = NZ, кватернионом q = (NX, NY, NZ, NW).
X: Double	cmd.Flt["Pos5D.X"] - координата X положения вершины инструмента.
Y: Double	cmd.Flt["Pos5D.Y"] - координата Y положения вершины инструмента.
Z: Double	cmd.Flt["Pos5D.Z"] - координата Z положения вершины инструмента.
NX: Double	cmd.Flt["Pos5D.NX"] - в зависимости от конфигурации системы может быть компонентой NX вектора нормали

TCLDMultiGoto: ComplexType	Команда многокоординатного перемещения
	инструмента, углом поворота вокруг одной из координатных осей или коэффициентом кватерниона.
NY: Double	cmd.Flt["Pos5D.NY"] - в зависимости от конфигурации системы может быть компонентой NY вектора нормали инструмента, углом поворота вокруг одной из координатных осей или коэффициентом кватерниона.
NZ: Double	cmd.Flt["Pos5D.NZ"] - в зависимости от конфигурации системы может быть компонентой NZ вектора нормали инструмента, углом поворота вокруг одной из координатных осей или коэффициентом кватерниона.
NW: Double	cmd.Flt["Pos5D.NW"] - если выбран способ ориентации инструмента в виде кватерниона, это четвёртый коэффициент кватерниона.
MachineStateFlags: Integer	cmd.Ptr["MachineStateFlags"].Bit[i], i=(0..31). Целое число, являющееся битовым полем из 32 бит, от 0 до 31. Если существует несколько вариантов расположения осей станка, обеспечивающих заданное положение инструмента в пространстве, каждый бит этого поля определяет выбор одного из возможных решений. Например, у пятиосевого станка с осями A и C для большинства положений инструмента есть два возможных решения — с положительным значением оси A и с отрицательным значением оси A. Нулевой бит этого поля выбирает одно из этих решений. Для стандартного шестиосевого шарнирного робота многие положения инструмента в пространстве могут быть

TCLDMultiGoto: ComplexType	Команда многокоординатного перемещения
	обеспечены разными комбинациями трёх его ключевых шарниров (основание, локоть и запястье), что даёт в сумме 8 возможных решений. В этом случае нулевой бит этого поля определяет положение основания, первый бит — положение локтя, а третий бит задаёт положение запястья робота. Таким образом, значение каждого из этих битов полностью определяется используемой схемой станка. При использовании станков, которым не требуются эти флаги, вывод этого параметра в CLData можно отключить в схеме станка.
Time: Double	Время перемещения в минутах.

Список координат, передаваемых через CLData команды MULTIGOTO, определяется схемой станка CAM-системы.

Используйте массив GMA для обработки команды MULTIGOTO. Свойствам массива GMA автоматически присваиваются заранее рассчитанные значения параметров перемещения по координатам. Параметры команды также доступны через оператор cmd.

Ниже приведён простой пример обработчика команды MULTIGOTO с использованием оператора cmd.

```
program MultiGoto
  if cmd.Ptr["Axes(AxisXPos)"]<>0 then begin
    X = cmd.Flt["Axes(AxisXPos).Value"]
  end
  if cmd.Ptr["Axes(AxisYPos)"]<>0 then begin
    Y = cmd.Flt["Axes(AxisYPos).Value"]
  end
  if cmd.Ptr["Axes(AxisZPos)"]<>0 then begin
    Z = cmd.Flt["Axes(AxisZPos).Value"]
  end
  if cmd.Ptr["Axes(AxisCPos)"]<>0 then begin
    C = cmd.Flt["Axes(AxisCPos).Value"]
  end
  OutBlock
end
```

Доступ из .NET

Обработчики (специфичный обработчик плюс скобки группы перемещений):

```
public override void OnMultiGoto(ICLDMultiGotoCommand cmd, CLDArray cld)
public override void OnBeforeMovement(ICLDMotionCommand cmd, CLDArray cld)
public override void OnAfterMovement(ICLDMotionCommand cmd, CLDArray cld)
```

Наследование: `ICLDMultiGotoCommand : ICLDMultiMotionCommand : ICLDMotionCommand : ICLDCommand` (`ICLDMultiGotoCommand` не добавляет собственных членов; всё перечисленное ниже унаследовано от `ICLDMultiMotionCommand` / `ICLDMotionCommand`).

Читайте параметры из строго типизированного объекта `cmd` (рекомендуется); доступны также «сырой» массив `cld[i]` (те же индексы, что и в sppx) и именованные обёртки `cmd[...]`. В таблице перечислены члены команды, включая унаследованные от базовых интерфейсов; общие члены `ICLDCommand` (навигация, `CLD`, `CLDFile`, `TechOperation`, ...) описаны в разделе Модель доступа к CLData.

Член	Тип	Описание
`cmd.EP`	`TInp3DPoint`	Конечная точка (X, Y, Z) перемещения. *(из `ICLDMotionCommand`)*
`cmd.EN`	`TInpRotation`	Конечная нормаль или ориентация перемещения (вектор нормали, углы Эйлера, кватернион, ... в зависимости от схемы станка).
`cmd.MSFlags`	`BitArray`	Флаги состояния станка / флипы робота, которые (вместе с декартовыми координатами) определяют состояние механизма в случае избыточности; значение каждого бита зависит от оборудования.
`cmd.Time`	double	Продолжительность перемещения в минутах.

Член	Тип	Описание
`cmd.AxesCount`	int	Число осей в списке `Axis[]`.
`cmd.Axis`	`ICLDMultiMotionAxesIndexer`	Список осей перемещения `[0..AxesCount-1]`; доступен по индексу, по текстовому `ID` или по имени (`Axis.X/.Y/.Z/.A/.B/.C/.C1/.C2/.A1..A6/.E1..E3`, `null`, если отсутствует).
`cmd.Axes`	`IEnumerable`	Перечислитель осей для `foreach`.
`cmd.HasAxis(string ID)`	bool	Истина, если перемещение содержит ось с указанным текстовым ID.

Элемент оси (`CLDMultiMotionAxis`) содержит `IntID` (числовой идентификатор оси станка), `ID` (текстовый идентификатор оси станка, заданный в схеме станка), `Value` (конечное значение оси; единицы — мм/дюймы/градусы/импульсы — в соответствии с определением оси) и проверки `IsX/IsY/IsZ/IsA/...`.

Пример (из постпроцессора Fanuc 30i mill):

```
public override void OnMultiGoto(ICLDMultiGotoCommand cmd, CLDArray cld)
{
    foreach (CLDMultiMotionAxis ax in cmd.Axes)
    {
        if (ax.IsX) nc.X.v = ax.Value;
        else if (ax.IsY) nc.Y.v = ax.Value;
        else if (ax.IsZ) nc.Z.v = ax.Value;
        else if (ax.IsA) nc.A.v = ax.Value;
        else if (ax.IsB) nc.B.v = ax.Value;
        else if (ax.IsC) nc.C.v = ax.Value;
    }
    nc.Block.Out();
}
```

Сопутствующие ссылки

- Модель доступа к CLData

OPSKIP - Условный пропуск

> Устарела. Эта команда является устаревшей - текущая версия CAM-системы не генерирует её автоматически. Она документирована здесь для постпроцессоров, которые всё ещё обрабатывают устаревший CLData.

Отображение команды

Как команда отображается в списке команд CLData:

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

Команда OPSKIP активирует и деактивирует режим формирования кадров управляющей программы с признаком условного пропуска. Обычно для пометки строк, подлежащих условному пропуску, используется символ `/`. Помеченные кадры обрабатываются системой ЧПУ только в том случае, если опция пропуска кадров выключена, в противном случае они игнорируются.

Доступ из sppx

Обработчик: `program OpSkip`

Параметр	Массив CLD	Имя CLD	Описание
ON(71) OFF(72)	CLD[1]	CLD.onoff	71 (ON) - активация условного пропуска, 72 (OFF) - деактивация условного пропуска.

Пример (из постпроцессора Sinumerik (840D) TurnMill):

```
program OpSkip
   print "Not marketed "
end
```

Сопутствующие ссылки

- Модель доступа к CLData

OPSTOP - Дополнительный останов

Отображение команды

Как команда отображается в списке команд CLData:

```
OPSTOP
```

Команда OPSTOP используется для формирования команд УП необязательного прерывания выполнения управляющей программы (обычно это код M01). Система ЧПУ прерывает выполнение по этой команде только если установлен соответствующий режим выполнения, иначе управляющая программа на этой команде не останавливается.

Доступ из sppx

Обработчик: `program OpStop`

Пример (из постпроцессора Fanuc (30i)_Mill):

```
program OpStop
  if M <> M@ then
    OutBlock
  M = 1      ! M01
  M@ = 0
end
```

Доступ из .NET

Обработчик:

```
public override void OnOpStop(ICLDOpStopCommand cmd, CLDArray cld)
```

Наследование: `ICLDOpStopCommand : ICLDCommand`.

Читайте параметры из строго типизированного объекта `cmd` (рекомендуется); доступны также «сырой» массив `cld[i]` (те же индексы, что и в sppx) и именованные обёртки `cmd["..."]`. В таблице перечислены члены команды, включая унаследованные от базовых интерфейсов; общие члены `ICLDCommand` (навигация, `CLD`, `CLDFile`, `TechOperation`, ...) описаны в разделе Модель доступа к CLData.

Пример - `OnOpStop` (из постпроцессора Fanuc (30i)_Mill_DN):

```
public override void OnOpStop(ICLDOpStopCommand cmd, CLDArray cld)
{
    nc.Block.Out();
    nc.M.Show(1);
    nc.Block.Out();
}
```

Сопутствующие ссылки

- Модель доступа к CLData

ORIGIN - Исходные координаты

Отображение команды

Как команда отображается в списке команд CLData:

```
ORIGIN X x, Y y, Z z, PPFUN f, N n, A a, B b, C c
```

Команда ORIGIN задаёт систему координат, в которой выводится последующий код управляющей программы.

Фактически у команды Origin есть два режима.

В первом режиме команда Origin используется для активации одной из предопределённых систем координат заготовки (G54-G59). Режим определяется параметром `CLD[4]` (`CLD.PPFun`), равным нулю. Номер активируемой системы координат задаётся параметром `CLD[5]` (`CLD.N`).

Во втором режиме команда Origin используется для определения преобразования системы координат заготовки (Sinumeric TRANS, ROT, Heidenhain CYCLE7, CYCLE19, PLANE SPATIAL|AXIAL, Fanuc G52, G68 и т. д.). Режим определяется параметром `CLD[4]` (`CLD.PPFun`), равным `PPFun(1079)`. Преобразование задаётся смещением по осям XYZ (CLD[1], CLD[2], CLD[3]) и поворотом вокруг тех же осей (CLD[6], CLD[7], CLD[8]).

Доступ из sppx

Обработчик: `program Origin`

Массив `cld` читается по индексу или по имени - в sppx по типизированному короткому имени (например, `cld.X`), в .NET по тому же имени как строковому ключу (например, `cld["X"]`).

Параметр	Массив CLD	Имя CLD	Описание
x y z	CLD[1] CLD[2] CLD[3]	CLD.X CLD.Y CLD.Z	Смещения локальной системы вдоль осей X, Y и Z соответственно.
f	CLD[4]	CLD.PPFun	Режим определения системы координат: 0 - команда задаёт одну из стандартных систем; PPFun(1079) - преобразование

Параметр	Массив CLD	Имя CLD	Описание
			локальной системы координат.
n	CLD[5]	CLD.N	Номер выбранной стандартной системы координат: 54 (G54), 55 (G55), 59 (G59) и так далее.
a b c	CLD[6] CLD[7] CLD[8]	CLD.A CLD.B CLD.C	Углы поворота вокруг осей X, Y и Z соответственно. Порядок поворотов вокруг осей (нотация углов Эйлера) может различаться и определяется настройками схемы станка.

Доступ по имени параметра (вспомогательные операторы)

Параметры, доступные через оператор `cmd` (одинаково в sppx и .NET):

TCLDOrigin: ComplexType	Команда определения системы координат
OriginType: Integer	cmd.Int["OriginType"] - Тип команды определения системы координат: 0 (SelectStandardLCS) - команда выбора стандартной системы координат заготовки (G54-G59), 1 (TransformLCS) - команда преобразования локальной системы координат.
CSNumber: Integer	cmd.Int["CSNumber"] - Номер стандартной системы координат заготовки: 54 (G54), 55 (G55), 59 (G59) и так далее.
MCS: ComplexType	cmd.Ptr["MCS"] - Параметры новой системы координат

TCLDOrigin: ComplexType	Команда определения системы координат
OriginPoint: ComplexType	cmd.Ptr["MCS.OriginPoint"] - Смещение исходной точки системы координат
X: Double	cmd.Flt["MCS.OriginPoint.X"] - смещение по X
Y: Double	cmd.Flt["MCS.OriginPoint.Y"] - смещение по Y
Z: Double	cmd.Flt["MCS.OriginPoint.Z"] - смещение по Z
RotAngles: ComplexType	cmd.Ptr["MCS.RotAngles"] - Углы поворота системы координат вдоль осей (в градусах). Порядок поворотов вокруг осей (нотация углов Эйлера) может различаться и определяется настройками схемы станка.
A: Double	cmd.Flt["MCS.RotAngles.A"] - угол поворота вокруг оси X
B: Double	cmd.Flt["MCS.RotAngles.B"] - угол поворота вокруг оси Y
C: Double	cmd.Flt["MCS.RotAngles.C"] - угол поворота вокруг оси Z
WCS: ComplexType	cmd.Ptr["WCS"] - Параметры новой системы координат. Параметры, задающие преобразование новой системы координат относительно подвижной системы координат заготовки. Включает геометрические (пространственные) координаты.
OriginPoint: ComplexType	cmd.Ptr["WCS.OriginPoint"] - Смещение начала системы координат
X: Double	cmd.Flt["WCS.OriginPoint.X"] - смещение по X
Y: Double	cmd.Flt["WCS.OriginPoint.Y"] - смещение по Y
Z: Double	cmd.Flt["WCS.OriginPoint.Z"] - смещение по Z
RotAngles: ComplexType	cmd.Ptr["WCS.RotAngles"] - Углы поворота системы координат вдоль осей (в градусах).

TCLDOrigin: ComplexType	Команда определения системы координат
	Порядок поворотов вокруг осей (нотация углов Эйлера) может различаться и определяется настройками схемы станка.
A: Double	cmd.Flt["WCS.RotAngles.A"] - угол поворота вокруг оси X
B: Double	cmd.Flt["WCS.RotAngles.B"] - угол поворота вокруг оси Y
C: Double	cmd.Flt["WCS.RotAngles.C"] - угол поворота вокруг оси Z
Axes: Array, Key="AxisID"	cmd.Ptr["Axes"] - Массив записей "Axes". Задаёт положения реальных (станочных) осей, соответствующих определяемой системе координат. Команда Origin может содержать положения для нескольких осей.
Axis: ComplexType	cmd.Ptr["Axes"].Item[Index] или cmd.Ptr["Axes(AxisName)"] - Элемент массива Axes. Содержит положение одной оси станка. Доступ к элементу массива возможен по индексу или по ключевому полю. Здесь AxisName - значение ключевого поля, которое должно совпадать со значением поля AxisID.
AxisID: String	cmd.Str["Axes(AxisName).AxisID"] - Идентификатор оси станка, для которой задано положение. Описывается в схеме станка.
Value: Double	cmd.Flt["Axes(AxisName).Value"] - Положение оси.
IsSpatial: Integer	cmd.Int["IsSpatial"] - параметр, определяющий, что имеет приоритет - геометрические координаты (A,B,C) или оси станка. Возможны два значения: 0 - приоритет имеют оси станка, заданные

TCLDOrigin: ComplexType	Команда определения системы координат
	массивом Axes; 1 - приоритет имеют геометрические координаты, заданные MCS и WCS.
PositioningMode: Integer	cmd.Int["PositioningMode"] - параметр, определяющий, генерирует ли эта команда реальные перемещения осей станка или только задаёт локальную систему координат внутри ЧПУ без каких-либо перемещений. Возможны три значения: 0 - STAY - преобразование не создаёт реальных перемещений станка; 1 - TURN - преобразование перемещает только поворотные оси станка; 2 - MOVE - преобразование перемещает все необходимые оси, чтобы сохранить прежнее положение вершины инструмента в новой системе координат.

Некоторые системы ЧПУ требуют определения локальных систем координат через пространственные углы, тогда как другие поддерживают определение локальных систем координат только через реальные углы осей станка. Команда ORIGIN включает всю информацию, необходимую для обоих способов определения. Свойства cmd.Ptr["MCS"] и cmd.Ptr["WCS"] содержат пространственные углы (A, B, C) локальной системы координат, а массив cmd.Ptr["Axes"] содержит соответствующие координаты поворотных осей станка. Главное преимущество свойства cmd.Ptr["Axes"] в том, что оно задаёт локальную систему координат однозначно, тогда как для cmd.Ptr["WCS"] обычно существует два возможных решения. Для систем ЧПУ, требующих пространственные углы для определения локальной системы координат, для разрешения неоднозначных случаев можно использовать углы осей станка (см. пример ниже).

Примечание: пространственные углы задаются относительно локальной системы координат, активной на момент определения команды ORIGIN.

Если система ЧПУ поддерживает оба способа определения локальной системы координат, желаемый формат вывода команды ORIGIN можно задать в CAM-системе в настройках параметров системы ЧПУ. В постпроцессоре желаемый формат вывода доступен в свойстве cmd.Int["IsSpatial"] команды ORIGIN.

Некоторые системы ЧПУ поддерживают различные режимы позиционирования команды ORIGIN. CAM-система предлагает следующие режимы позиционирования локальной системы

координат.

cmd.Int["PositioningMode"]=STAY. Команда origin не перемещает оси станка.

cmd.Int["PositioningMode"]=TURN. Команда origin поворачивает оси станка так, чтобы направление оси инструмента совпало с осью Z локальной системы координат

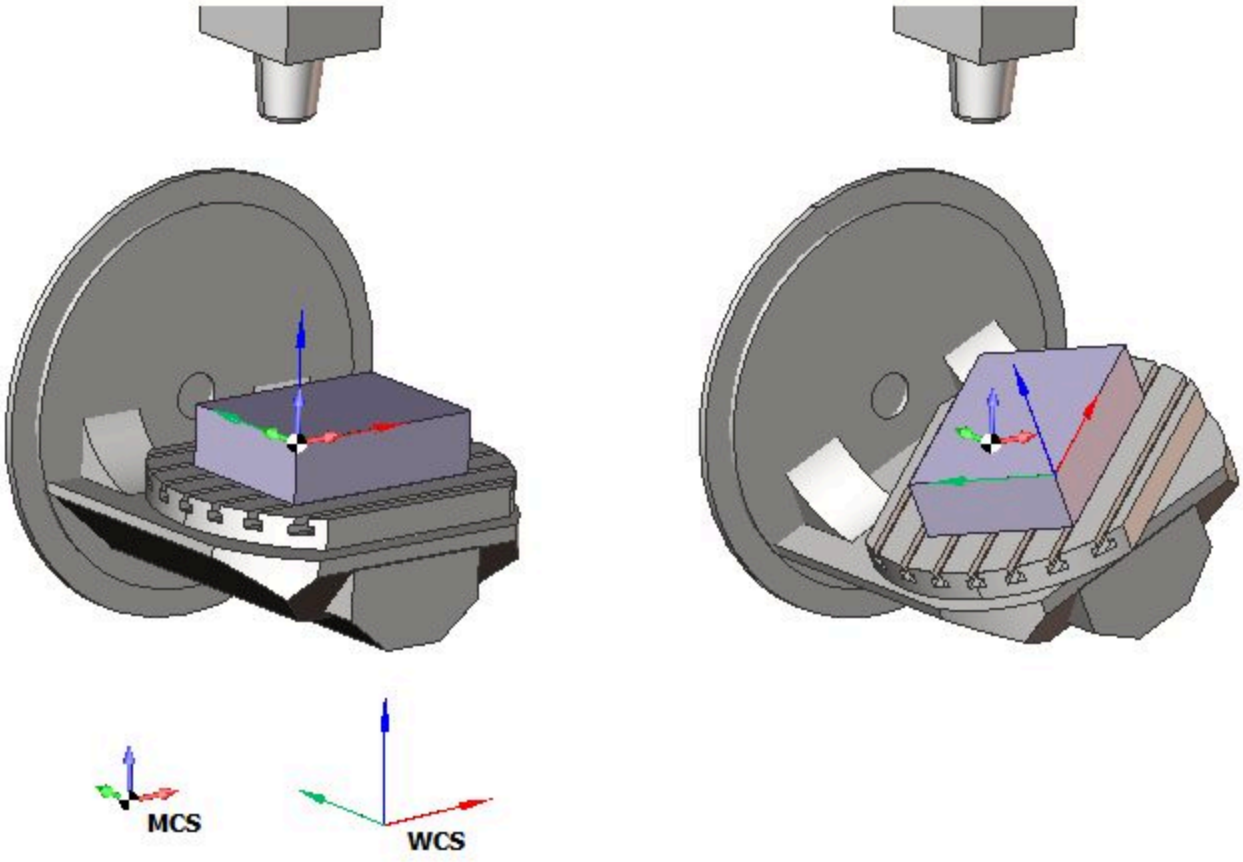
cmd.Int["PositioningMode"]=MOVE. Команда origin поворачивает оси станка так, чтобы совместить ось инструмента с осью Z локальной системы координат, и перемещает линейные оси так, чтобы положение вершины инструмента относительно заготовки осталось прежним

Системы ЧПУ определяют локальную систему координат относительно некоторой базовой системы координат заготовки (G54). В зависимости от используемой системы ЧПУ для определения локальной системы координат следует использовать либо матрицу MCS, либо матрицу WCS.

Современные системы ЧПУ знают о кинематике станка. На таких системах ЧПУ система координат заготовки поворачивается вместе с поворотным столом. Поэтому на таких системах ЧПУ можно определять локальную систему координат относительно поворачивающейся системы координат заготовки. В этом случае удобно использовать матрицу WCS.

Старые системы ЧПУ ничего не знают о кинематике станка и не могут поворачивать систему координат заготовки вместе с поворотным столом. Для таких систем ЧПУ для определения преобразования локальной системы координат следует использовать матрицу MCS.

На приведённых ниже рисунках показано различие между базовыми системами координат для матриц MCS и WCS. Предпочтительнее использовать матрицу WCS, так как она не зависит от установки заготовки, тогда как MCS сильно от неё зависит.



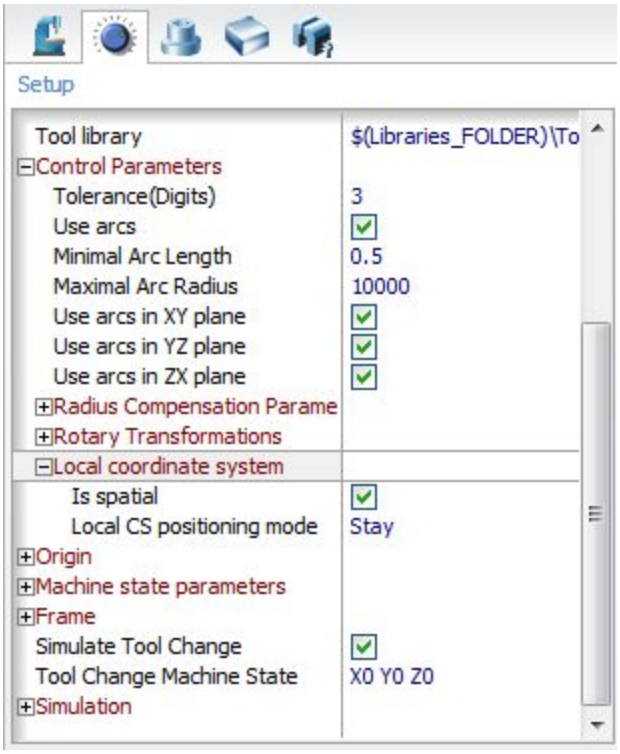
Настройки локальной системы координат CAM-системы.

В CAM-системе настройки команды ORIGIN задаются в файле конфигурации станка.

```
SCType ID="MachineName" type="AbstractMachine"

  ...
  <ControlData>
  ...
  <LocalCS>
  <IsSpatial DefaultValue="True"/>
  <PositioningMode DefaultValue="Stay"/>
  </LocalCS>
  ...
  </ControlData>
  ...
  <SCType>
```

Эти параметры также доступны из графического интерфейса, в разделе "Параметры системы ЧПУ" параметров станка.



Примеры кода для команды ORIGIN.

Пример - смещение начала координат G54-G59 / G52:

```
program Origin
  if cmd.Int["OriginType"]=0 then begin !G54-G59
    Output "G" + cmd.Str["CSNumber"]
  end else begin !G52 X Y Z
    X = cmd.Flt["MCS.OriginPoint.X"]
    Y = cmd.Flt["MCS.OriginPoint.Y"]
    Z = cmd.Flt["MCS.OriginPoint.Z"]
    Output "G52 X" + Str(X) + " Y" + Str(Y) + " Z" + Str(Z)
  end
end
```

Пример - смещение начала координат и рабочая плоскость в стиле Heidenhain CYCL DEF:

```
program Origin
  if CLD[4]=0 then begin ! Выбор стандартной LCS (G54-G59)
    ! Для Heidenhain ничего не делать
  end else if CLD[4]=1079 then begin ! Преобразование LCS
    Output "CYCL DEF 7.0 DATUM SHIFT"
    Output "CYCL DEF 7.1 X" + Str(cmd.Flt["WCS.OriginPoint.X"])
    Output "CYCL DEF 7.2 Y" + Str(cmd.Flt["WCS.OriginPoint.Y"])
    Output "CYCL DEF 7.3 Z" + Str(cmd.Flt["WCS.OriginPoint.Z"])
    if (abs(cmd.Flt["WCS.RotAngles.A"])>0.0001) or
```

```
    (abs(cmd.Flt["WCS.RotAngles.B"])>0.0001) or
    (abs(cmd.Flt["WCS.RotAngles.C"])>0.0001)
    then begin
      Output "CYCL DEF 19.0 WORKING PLANE"
      if cmd.Int["IsSpatial"]>0 then begin ! Пространственные координаты
        Output "CYCL DEF 19.1 A" + str(cmd.Flt["WCS.RotAngles.A"]) +
        " B" + str(cmd.Flt["WCS.RotAngles.B"]) +
        " C" + str(cmd.Flt["WCS.RotAngles.C"])
      end else begin ! Оси станка
        OutStr$ = "CYCL DEF 19.1"
        if cmd.Ptr["Axes(AxisBPos)"]<>0 then begin
          OutStr$ = OutStr$ + " B" + str(cmd.Flt["Axes(AxisBPos).Value"])
        end
        if cmd.Ptr["Axes(AxisCPos)"]<>0 then begin
          OutStr$ = OutStr$ + " C" + str(cmd.Flt["Axes(AxisCPos).Value"])
        end
        Output OutStr$
      end
    end
  end
end
```

Доступ из .NET

Обработчики - помимо базового `OnOrigin`, платформа вызывает `OnWorkpieceCS` для случая G54-G59 и `OnLocalCS` для преобразования локальной системы координат; ORIGIN также является командой перемещения, поэтому срабатывают скобки группы перемещений:

```
public override void OnOrigin(ICLDOriginCommand cmd, CLDArray cld)         //
базовый диспетчер
public override void OnWorkpieceCS(ICLDOriginCommand cmd, CLDArray cld)    //
G54-G59
public override void OnLocalCS(ICLDOriginCommand cmd, CLDArray cld)        //
преобразование локальной СК
public override void OnBeforeMovement(ICLDMotionCommand cmd, CLDArray cld)
public override void OnAfterMovement(ICLDMotionCommand cmd, CLDArray cld)
```

Наследование: `ICLDOriginCommand : ICLDMultiMotionCommand : ICLDMotionCommand : ICLDCommand`.

Читайте параметры из строго типизированного объекта `cmd` (рекомендуется); также доступны «сырой» массив `cld[i]` (те же индексы, что и в sppx) и именованные обёртки `cmd[...]`. В таблице перечислены члены команды, включая унаследованные от базовых интерфейсов;

общие члены `ICLDCommand` (навигация, `CLD`, `CLDFile`, `TechOperation`, ...) описаны в Модели доступа к CLData.

Свойство	Тип	Описание
`cmd.OriginType`	`CLDOriginType`	Выбор стандартной системы координат заготовки (G54-G59) или переключение на конкретную локальную систему координат (G68.2, PLANE SPATIAL, Cycle800).
`cmd.IsWorkpieceCS` `cmd.IsLocalCS`	bool	True, если это команда выбора системы координат заготовки / переключения локальной системы координат.
`cmd.IsOn` `cmd.IsOff`	bool	True, если это команда включения / выключения переключения системы координат.
`cmd.CSNumber`	double	Номер стандартной системы координат заготовки для активации.
`cmd.MCS`	`TInpLocation`	Декартовы координаты новой системы координат относительно базы станка.
`cmd.WCS`	`TInpLocation`	Декартовы координаты новой системы координат относительно заготовки (может поворачиваться вместе с ней).
`cmd.QMCS` `cmd.QWCS`	`TInpQuaternion`	Кватернион ориентации новой системы координат относительно базы станка / заготовки.

Свойство	Тип	Описание
`cmd.Convention`	`TInpComplexRotationConvention`	Соглашение об углах Эйлера, используемое внутри MCS и WCS.
`cmd.IsSpatial`	bool	True, если для определения новой системы координат следует предпочесть декартовы координаты, а не физические оси станка.
`cmd.PositioningMode`	`CLDOriginPositionMode`	Как изменяются оси станка при переключении системы координат: `Stay`, `Turn` или `Move`.

Плюс унаследованные члены `ICLDMultiMotionCommand` (`EN`, `MSFlags`, `Time`, `AxesCount`, `Axis`, `Axes`, `HasAxis`) и `EP` - их описание см. в MULTIGOTO.

Пример - `OnWorkpieceCS` (из постпроцессора Fanuc (30i)_Mill_DN):

```
public override void OnWorkpieceCS(ICLDOriginCommand cmd, CLDArray cld)
{
    nc.GWCS.v = cmd.CSNumber;    // G54..G59
}
```

Пример - `OnLocalCS` (из постпроцессора Fanuc (30i)_Mill_DN):

```
public override void OnLocalCS(ICLDOriginCommand cmd, CLDArray cld)
{
    if (cmd.IsOn) {
        nc.GLCS.Show(68.2);
        nc.X.Show(cmd.WCS.P.X);
        nc.Y.Show(cmd.WCS.P.Y);
        nc.Z.Show(cmd.WCS.P.Z);
        nc.I.Show(cmd.WCS.N.A);
        nc.J.Show(cmd.WCS.N.B);
        nc.K.Show(cmd.WCS.N.C);
        nc.Block.Hide(nc.GInterp);
        nc.Block.Out();
        nc.Block.Reset(nc.X, nc.Y, nc.Z);
        nc.OutWithN("G53.1");
```

```
    } else {
        nc.GLCS.Show(69);
        nc.Block.Out();
    }
}
```

Сопутствующие ссылки

- Модель доступа к CLData

PALETA - Смена палеты

> Устарела. Эта команда является устаревшей - текущая версия CAM-системы не генерирует её автоматически. Она документирована здесь для постпроцессоров, которые всё ещё обрабатывают устаревший CLData.

Отображение команды

Как команда отображается в списке команд CLData:

```
PALETA N n
```

Команда PALETA используется для формирования кадров управляющей программы, управляющих сменой палеты. Номер новой палеты передаётся в качестве параметра команды.

Доступ из sppx

Обработчик: `program Paleta`

Параметр	Массив CLD	Имя CLD	Описание
n	CLD[1]	CLD.N	Номер палеты

Сопутствующие ссылки

- Модель доступа к CLData

PARTNO - Номер детали

Отображение команды

Как команда отображается в списке команд CLData:

```
PARTNO "<CAM system project name>"
```

PARTNO — эта команда является первой в списке технологических команд проекта.
Используйте команду PARTNO для формирования начального блока управляющей программы.
Команда PARTNO помещает имя проекта в переменную CLDATA$.

Доступ из sppx

Обработчик: `program PartNo`

Пример (из постпроцессора Fanuc (30i)_Mill):

```
program PartNo
! Initialization of global variables
  MaxReal = 99999.999
  XT_ = 0;  FROMX_ = 0
  YT_ = 0;  FROMY_ = 0
  ZT_ = 0;  FROMZ_ = 0

  KorEcv = 40; KorEcv@=KorEcv
  KorDl  = 49; KorDl@=KorDl
  Cycle  = 80; Cycle@=Cycle
  Plane  = -17; Plane@=Plane
  GSpindl = 97; GSpindl@ = GSpindl
  CoordSys = 54; CoordSys@ = CoordSys; WCSNumber = 54
  PG = 0; PG@ = PG
  YAddress = "Y"
  IsRotaryInterpolation = 0
  IsMultiaxisInterpolation = 0
  CylR = 0
  Pi = 3.14159265358979323846264338327795
  Deg2Rad = Pi / 180
  Rad2Deg = 180 / Pi
  CirclesThroughRadius = 0
  call check_Euler_angles_types
  IsLocalCS = 0
  G68_2_X = MaxReal
  G68_2_Y = MaxReal
```

```
  G68_2_Z = MaxReal
  G68_2_I = MaxReal
  G68_2_J = MaxReal
  G68_2_K = MaxReal
  HasAxisA = 0
  HasAxisB = 0
  HasAxisC = 0

! Constsnt cycles variables
  CycleOn = 0          ! 0 - cycle Off, 1 - cycle On
  KodCycle = 0         ! constant cycle code
  Za = 0               ! work depth
  Zf = 0               ! safe level
  ZP_ = 0               ! reverse move level
  Zl = 0               ! depth of one drilling step
  Zi = 0               ! transitional value
  Dwell = 0            ! pause in constsnt cycles
  Fr = 0               ! work FEED_
! Set machene functions by default
  X = MaxReal; X@=X
  Y = MaxReal; Y@=Y
  Z = MaxReal; Z@=Z
  A = MaxReal; A@=A
  B = MaxReal; B@=B
  C = MaxReal; C@=C
  S = 0; S@=S
  SpDir = 5
  SpRPM = 0
  DrillPause = MaxReal; DrillPause@ = DrillPause;
  QStep = MaxReal; QStep@ = QStep;
  Zcycle=Maxreal; Zcycle@=Zcycle
  P=0;Q=0
  D_=0; H_=0;             ! values D, H
  RapFR  = 2000
  INTERP_ = 100;  INTERP_@ = 100
  Msp = 5; Msp@ = 5     ! spindle off
  Mc=9; Mc@=Mc;
  Zcycle=Maxreal
  OperationIsLathe = 2
  BlkN=100

! Creating NC heading
  output "%"
  MainProgramID = 5
  call GetProgNumberFromName(NCName$,MainProgramID)
  !Input "Input program number:", MainProgramID
```

```
!  ProgNum = MainProgramID; ProgNum@ = MaxReal
  FormBlock
  s$ = NCName$
  call CheckProgramName(s$)
  output OutStr$ + UPCASE(s$)

  output ""
  Output "( GENERATED BY CAM )"
  Output "( DATE: " + CurDate + " )"
  Output "( TIME: " + CurTime + " )"

  output ""
  call PrintAllTools
  output ""
  Output "G00G21G40G49G69G80G90G17"
  output "<StartGoHome>"

  OutBlock
end
```

Доступ из .NET

Обработчики:

```
public override void OnPartNo(ICLDPartNoCommand cmd, CLDArray cld)
public override void OnStartProject(ICLDProject prj)   // project-level, raised
on PARTNO
```

Наследование: `ICLDPartNoCommand : ICLDCommand`.

Читайте параметры из строго типизированного объекта `cmd` (рекомендуется); доступны также «сырой» массив `cld[i]` (те же индексы, что и в sppx) и именованные обёртки `cmd["..."]`. В таблице перечислены члены команды, включая унаследованные от базовых интерфейсов; общие члены `ICLDCommand` (навигация, `CLD`, `CLDFile`, `TechOperation`, ...) описаны в разделе Модель доступа к CLData.

Член	Тип	Описание
`cmd.ProjectName`	`string`	Возвращает имя файла проекта без пути и расширения.

Пример - `OnStartProject` (из постпроцессора Fanuc (30i)_Mill_DN):

```
public override void OnStartProject(ICLDProject prj)
{
    probingCycle.Init();

    mainNC = new NCFile();
    nc = mainNC;
    nc.OutputFileName = Settings.Params.Str["OutFiles.NCFileName"];
    // Log.Info("Output file name: " + nc.OutputFileName);
    nc.ProgNumber = Settings.Params.Int["OutFiles.NCProgNumber"];

    CirclesThroughRadius = Settings.Params.Bol["BlockFormat.CirclesThroughRadius"];

    if (Settings.Params.Bol["BlockFormat.Numbering"])
        nc.BlockN.Show();
    else
        nc.BlockN.Disable();
    if (Settings.Params.Bol["BlockFormat.Spaces"])
        nc.Block.WordsSeparator = " ";
    else
        nc.Block.WordsSeparator = "";

    nc.WriteLine("%");
    nc.WriteLine("O" + Str(nc.ProgNumber));

    nc.WriteLine();
    nc.WriteLine(String.Format("( {0} )", "Generated by CAM"));
    nc.WriteLine(String.Format("( {0} )", "Date: " +
DateTime.Now.ToShortDateString()));
    nc.WriteLine(String.Format("( {0} )", "Time: " +
DateTime.Now.ToShortTimeString()));
    nc.WriteLine();

    PrintAllTools();
    nc.WriteLine();

    nc.Block.Show(nc.GAbsInc, nc.GFeed ,nc.GMeasure, nc.GWCS, nc.GLCS, nc.GPlane,
        nc.GLCompens, nc.GRCompens, nc.GInterp, nc.GCycle);
    nc.Block.Out();
    nc.OutWithN("G53", nc.Z.ToString(0));
    OutG53ABC();
}
```

Сопутствующие ссылки

- Модель доступа к CLData

PHYSICGOTO - Перемещение в физических осях станка

Отображение команды

Как команда отображается в списке команд CLData:

```
PhysicGOTO COUNT 2, MACHINE, X 0, Y 0, GEOM X 125.281, Y -32, Z 194, NX 0, NY 0, NZ 1, NW 0, MSF 0, Time 0.013
```

PhysicGOTO - это универсальная технологическая команда, которая используется для передачи перемещения по координатам станка. Она похожа на команду MULTIGOTO, разница состоит в следующем. Команда MULTIGOTO содержит координаты осей относительно текущей системы координат заготовки (пространственные координаты относительно G54-G56 или локальной системы координат), а интерполяция базовых осей выполняется с контролем прямолинейности траектории для каждой из промежуточных точек. В ISO-кодах это соответствует командам линейной интерполяции G01. В системе управления роботом это соответствует типу перемещения Linear (L, LIN, MOVEL), Control path (CP) motion. В отличие от неё, команда PhysicGOTO содержит физические значения координат относительно машинных нулей по каждой оси, а интерполяция базовых осей не контролирует промежуточные точки траектории, проверяется только достижимость конечной точки. В ISO-кодах ей соответствует команда перемещения G53 (SUPA в Sinumerik, M91 в HEIDENHAIN). В системе управления роботом ей соответствует тип перемещения Joint move (J, MOVEJ), Point-to-point (PTP) motion.

Доступ из sppx

Обработчик: `program PhysicGOTO`

Параметр	Массив CLD	Описание
X Y Z	CLD[1] CLD[2] CLD[3]	Пространственные координаты кончика инструмента относительно активной системы координат заготовки
NX NY NZ NW	CLD[4] CLD[5] CLD[6] CLD[7]	Ориентация системы координат инструмента относительно активной системы координат заготовки. Способ задания

Параметр	Массив CLD	Описание
		ориентации зависит от настроек схемы станка. Это могут быть углы Эйлера в разных нотациях, вектор нормали инструмента, кватернион или вращающийся вектор.
MSF	CLD[8]	Флаги состояния станка (flips). Актуально для роботов и станков, которые могут обеспечивать несколько конфигураций физических осей для одной и той же пространственной позиции. См. описание в следующей таблице.
Time	CLD[9]	Время перемещения.
AxesCount	CLD[10]	Число физических координат станка, задействованных в этой команде.
r1	CLD[11]	Номер оси с именем Axis1Pos в списке координат.
Val1	CLD[12]	Значение координаты станка Axis1Pos.
r2	CLD[13]	Номер оси с именем Axis2Pos в списке координат.
Val2	CLD[14]	Значение координаты станка Axis2Pos.
…	…	…
rN	CLD[2*N+9]	Номер оси с именем AxisNPos в списке координат.

Параметр	Массив CLD	Описание
ValN	CLD[2*N+10]	Значение координаты станка AxisNPos.

Параметры, доступные через оператор cmd

TCLDPhysicGoto: ComplexType	Команда многокоординатного перемещения по физическим осям станка
Axes: Array, Key="AxisID"	cmd.Ptr["Axes"] - Массив структур типа Axis. Одна команда может включать перемещение по нескольким осям, положение каждой из которых хранится в этом массиве.
Axis: ComplexType	cmd.Ptr["Axes"].Item[Index] или cmd.Ptr["Axes(AxisName)"] - Отдельный элемент массива Axes. Содержит информацию о перемещении по одной оси станка. Доступ к элементам массива возможен либо по индексу, либо по ключевому полю. Здесь AxisName - значение ключевого поля, которое должно совпадать со значением поля AxisID.
AxisID: String	cmd.Str["Axes(AxisName).AxisID"] - Идентификатор оси станка, для которой задаётся новое положение. Определяется схемой станка.
Value: Double	cmd.Flt["Axes(AxisName).Value"] - Новое положение оси станка, в которое она перемещается.
Pos5D: ComplexType	cmd.Ptr["Pos5D"] - структура, содержащая координаты, определяющие конечное положение инструмента в пространстве. Определяет не только положение настроечной точки инструмента (поля X, Y и Z - декартовы координаты), но и ориентацию инструмента относительно текущей системы координат (поля NX, NY, NZ и NW). В

TCLDPhysicGoto: ComplexType	Команда многокоординатного перемещения по физическим осям станка
	зависимости от настроек системы (задаются в схеме станка) ориентация инструмента может задаваться: вектором нормали n = (NX, NY, NZ), пространственными углами поворота вокруг координатных осей - углами Эйлера (нотация углов Эйлера, то есть последовательность поворотов, также задаётся в настройках) A = NX, B = NY, C = NZ, кватернионом q = (NX, NY, NZ, NW).
X: Double	cmd.Flt["Pos5D.X"] - декартова координата X положения кончика инструмента.
Y: Double	cmd.Flt["Pos5D.Y"] - декартова координата Y положения кончика инструмента.
Z: Double	cmd.Flt["Pos5D.Z"] - декартова координата Z положения кончика инструмента.
NX: Double	cmd.Flt["Pos5D.NX"] - в зависимости от настроек системы может быть компонентом NX вектора нормали инструмента, углом поворота вокруг одной из координатных осей или коэффициентом кватерниона.
NY: Double	cmd.Flt["Pos5D.NY"] - в зависимости от настроек системы может быть компонентом NY вектора нормали инструмента, углом поворота вокруг одной из координатных осей или коэффициентом кватерниона.
NZ: Double	cmd.Flt["Pos5D.NZ"] - в зависимости от настроек системы может быть компонентом NZ вектора нормали инструмента, углом поворота вокруг одной из координатных осей или коэффициентом кватерниона.
NW: Double	cmd.Flt["Pos5D.NW"] - если выбран способ задания ориентации инструмента в виде

TCLDPhysicGoto: ComplexType	Команда многокоординатного перемещения по физическим осям станка
	кватерниона, представляет собой четвёртый коэффициент кватерниона.
MachineStateFlags: Integer	cmd.Ptr["MachineStateFlags"].Bit[i], i=(0..31). Целое число, представляющее собой битовое поле из 32 бит от 0 до 31. Если существует несколько вариантов расположения осей станка, обеспечивающих заданное положение инструмента в пространстве, каждый бит этого поля определяет выбор одного из возможных решений. Например, пятиосевой станок с осями A и C для большинства положений инструмента имеет два возможных решения - с положительным значением оси A и с отрицательным значением оси A. Нулевой бит этого поля будет определять, какое из этих решений выбрать. Для стандартного шестиосевого шарнирного робота многие положения инструмента в пространстве могут обеспечиваться разными сочетаниями положений трёх его ключевых сочленений (базы, локтя и кисти), что даёт в общей сложности 8 возможных решений. В этом случае нулевой бит этого поля будет определять положение базы, первый бит - положение локтя, а третий бит будет задавать положение кисти робота. Таким образом, смысл каждого бита этого поля полностью определяется используемой схемой станка. При использовании станков, для которых эти флаги не нужны, в схеме станка можно отключить вывод этого параметра в CLData.
Time: Double	Время перемещения в минутах.

Список координат, передаваемых через CLData команды PHYSICGOTO, определяется схемой станка CAM-системы.

Для работы с командой PHYSICGOTO используйте массив GMA. Свойствам массива GMA автоматически присваиваются заранее рассчитанные значения параметров перемещения по координатам. Параметры команды также доступны через оператор cmd.

Ниже приведён простой пример обработчика команды PHYSICGOTO с использованием оператора cmd.

```
program PhysicGoto
  if cmd.Ptr["Axes(AxisXPos)"]<>0 then begin
    X = cmd.Flt["Axes(AxisXPos).Value"]
  end
  if cmd.Ptr["Axes(AxisYPos)"]<>0 then begin
    Y = cmd.Flt["Axes(AxisYPos).Value"]
  end
  if cmd.Ptr["Axes(AxisZPos)"]<>0 then begin
    Z = cmd.Flt["Axes(AxisZPos).Value"]
  end
  if cmd.Ptr["Axes(AxisAPos)"]<>0 then begin
    A = cmd.Flt["Axes(AxisAPos).Value"]
  end
  if cmd.Ptr["Axes(AxisCPos)"]<>0 then begin
    C = cmd.Flt["Axes(AxisCPos).Value"]
  end
  if (X<>X@) or (Y<>Y@) or (Z<>Z@) or (A<>A@) or (C<>C@) then begin
    Interp_ = 53; Interp_@ = 0; ! G53
    OutBlock
  end
end
```

Доступ из .NET

Обработчики:

```
public override void OnBeforeMovement(ICLDMotionCommand cmd, CLDArray cld)   //
групповой обработчик - перед любой командой движения
public override void OnPhysicGoto(ICLDPhysicGotoCommand cmd, CLDArray cld)
public override void OnAfterMovement(ICLDMotionCommand cmd, CLDArray cld)   //
групповой обработчик - после любой команды движения
```

Наследование: `ICLDPhysicGotoCommand : ICLDMultiMotionCommand : ICLDMotionCommand : ICLDCommand`.

Читайте параметры из строго типизированного объекта `cmd` (рекомендуется); также доступны «сырой» массив `cld[i]` (те же индексы, что и в sppx) и именованные помощники `cmd["..."]`. В

таблице перечислены члены команды, включая унаследованные от базовых интерфейсов; общие члены `ICLDCommand` (навигация, `CLD`, `CLDFile`, `TechOperation`, ...) описаны в разделе Модель доступа к CLData.

Член	Тип	Описание
`cmd.EN`	`TInpRotation`	Конечная нормаль или ориентация перемещения в декартовой системе координат. В зависимости от настроек текущей схемы станка это может быть вектор нормали, углы Эйлера, кватернион и т. д.
`cmd.MSFlags`	`BitArray`	Флаги состояния станка или флипы робота, которые помогают определить (совместно с декартовыми координатами) конкретное состояние механизма в случае избыточности. То, какой бит отвечает за какой флаг, зависит от типа и конфигурации оборудования.
`cmd.Time`	`double`	Продолжительность перемещения в минутах.
`cmd.AxesCount`	`int`	Количество осей, которые содержит перемещение внутри списка Axis[].
`cmd.Axis`	`ICLDMultiMotionAxesIndexer`	Список осей перемещения [0..(AxesCount-1)].
`cmd.Axes`	`IEnumerable`	Объект-перечислитель осей перемещения, позволяющий использовать удобный синтаксис foreach.

Член	Тип	Описание
`cmd.HasAxis(...)`	`bool`	Возвращает «true», если перемещение содержит ось с указанным текстовым идентификатором. Иначе возвращает «false».
`cmd.EP`	`TInp3DPoint`	Конечная точка (X, Y, Z) перемещения

Пример - `OnPhysicGoto` (из постпроцессора Fanuc (30i)_Mill_DN):

```
public override void OnPhysicGoto(ICLDPhysicGotoCommand cmd, CLDArray cld)
{
    foreach(CLDMultiMotionAxis ax in cmd.Axes) {
        if (ax.IsX) {
            nc.X.Show(ax.Value);
        } else if (ax.IsY) {
            nc.Y.Show(ax.Value);
        } else if (ax.IsZ) {
            nc.Z.Show(ax.Value);
        } else if (ax.IsA) {
            nc.A.Show(ax.Value);
        } else if (ax.IsB)  {
            nc.B.Show(ax.Value);
        } else if (ax.IsC) {
            nc.C.Show(ax.Value);
        }
    }
    if (nc.X.Changed || nc.Y.Changed || nc.Z.Changed || nc.A.Changed || nc.B.Changed
|| nc.C.Changed) {
        nc.GHome.Show(53);
        nc.Block.Out();
    }
}
```

Сопутствующие ссылки

- Модель доступа к CLData

PLANE - Рабочая плоскость

Отображение команды

Как команда отображается в списке команд CLData:

```
PLANE XY(33) | YZ(37) | XZ(41) | YX(133) | ZY(137) | ZX(141)
```

Команда PLANE используется для задания активной плоскости обработки. Обычно эта команда транслируется в G-коды G17, G18, G19. Активная плоскость используется при круговой интерполяции (определяет плоскость дуг), для реализации коррекции на радиус инструмента, для задания циклов обработки отверстий и т. д.

Доступ из sppx

Обработчик: `program Plane`

Параметр	Массив CLD	Имя CLD	Описание
XY(33) YZ(37) XZ(41) YX(133) ZY(137) ZX(141)	CLD[1]	CLD.Plane	Код плоскости: 33 – XY, 37 – YZ, 41 – XZ, 133 – YX, 137 – ZY, 141 – ZX.

Для создания ISO-постпроцессора используйте шаблон ISO.G, которому автоматически присваивается код плоскости:

Значение элемента CLD	Значение ISO
CLD[1] = 33	ISO.G = 17
CLD[1] = 37	ISO.G = 19
CLD[1] = 41	ISO.G = 18

Пример (из постпроцессора Fanuc (30i)_Mill):

```
program Plane
  case cld[1] of
    33: Plane = 17;      ! XY
```

```
    37: Plane = 19;      ! YZ
    41: Plane = 18;      ! XZ
    133: Plane = -17;      ! XY
    137: Plane = -19;      ! YZ
    141: Plane = -18;      ! XZ
  end
  if (Plane<>Plane@) and (abs(Plane)=abs(Plane@)) then
    Plane@ = Plane
!  if Plane<>Plane@ then
!    OutBlock
  case abs(Plane) of
    17: begin
      regZPlane = Rgs["Z"].Num
      regZCycle = Rgs["ZCYCLE"].Num
    end
    18: begin
      regZPlane = Rgs["Y"].Num
      regZCycle = Rgs["YCYCLE"].Num
    end
    19: begin
      regZPlane = Rgs["X"].Num
      regZCycle = Rgs["XCYCLE"].Num
    end
    else begin
      regZPlane = Rgs["Z"].Num
      regZCycle = Rgs["ZCYCLE"].Num
    end
  end
end
```

Доступ из .NET

Обработчик:

```
public override void OnPlane(ICLDPlaneCommand cmd, CLDArray cld)
```

Наследование: `ICLDPlaneCommand : ICLDCommand`.

Читайте параметры из строго типизированного объекта `cmd` (рекомендуется); доступны также «сырой» массив `cld[i]` (те же индексы, что и в sppx) и именованные обёртки `cmd["..."]`. В таблице перечислены члены команды, включая унаследованные от базовых интерфейсов; общие члены `ICLDCommand` (навигация, `CLD`, `CLDFile`, `TechOperation`, ...) описаны в разделе Модель доступа к CLData.

Член	Тип	Описание
`cmd.Plane`	`CLDPlaneType`	Содержит точный тип плоскости. Возможны 6 стандартных значений: XY, YZ, ZX, InvXY, InvYZ, InvZX. Возможные значения: `XY` (17) - Стандартная плоскость XY, или G17 `YZ` (19) - Стандартная плоскость YZ, или G19 `ZX` (18) - Стандартная плоскость ZX, или G18 `InvXY` (-17) - Плоскость, противоположная плоскости XY (-17) `InvYZ` (-19) - Плоскость, противоположная плоскости YZ (-19) `InvZX` (-18) - Плоскость, противоположная плоскости ZX (-18)
`cmd.PlaneGCode`	`int`	Содержит номер стандартного ISO G-кода для типа плоскости. Возможны 6 значений: 17, 18, 19, -17, -18, -19.
`cmd.PlaneSign`	`int`	Возвращает "+1" для плоскостей XY, YZ, ZX. Возвращает "-1" для плоскостей InvXY, InvYZ и InvZX.

Пример - `OnPlane` (из постпроцессора Fanuc (30i)_Mill_DN):

```
public override void OnPlane(ICLDPlaneCommand cmd, CLDArray cld)
{
    nc.GPlane.v = cmd.PlaneGCode;
    planeSign = cmd.PlaneSign;
```

```
    switch (cmd.Plane) {
        case CLDPlaneType.XY:
        case CLDPlaneType.InvXY:
            planeZIndex = 3;
            nc.PlaneZReg = nc.Z;
            break;
        case CLDPlaneType.ZX:
        case CLDPlaneType.InvZX:
            planeZIndex = 2;
            nc.PlaneZReg = nc.Y;
            break;
        case CLDPlaneType.YZ:
        case CLDPlaneType.InvYZ:
            planeZIndex = 1;
            nc.PlaneZReg = nc.X;
            break;
    }
}
```

Сопутствующие ссылки

- Модель доступа к CLData

POWER - Регулировка мощности

Отображение команды

Как команда отображается в списке команд CLData:

```
POWER N n, Value
```

Команда CLData POWER задаёт процент номинальной мощности (заданной в параметрах операции CAM) для конкретного активного исполнительного устройства (лазер, форсунка, дуговой сварочный аппарат и т. д.), который следует использовать для последующих перемещений. Например, CAM-система может вычислять значение, передаваемое этой командой, как процент от номинальной толщины заготовки или процент от толщины обрабатываемого слоя (ближайшей точки предыдущего прохода).

Доступ из sppx

Обработчик: `program Power`

Параметр	Массив CLD	Имя CLD	Описание
N	CLD[1]	CLD.N	Уникальный идентификатор устройства в CAM, мощность которого нужно изменить
Value	CLD[2]	CLD.Value	Значение мощности в процентах от некоторой константы, заданной либо в параметрах операции, либо подразумеваемой неявно.

Пример:

```
program Power
  ! cld[1] - идентификатор устройства, cld[2] - мощность в процентах от
номинального значения
  NominalCurrent = 120                                    ! номинальный ток (A)
```

```
исполнительного устройства, обычно берётся из параметров операции
  Output "A" + Str(NominalCurrent * cld[2] / 100)       ! фактический ток для
последующих перемещений
end
```

Доступ из .NET

Обработчик:

```
public override void OnPower(ICLDPowerCommand cmd, CLDArray cld)
```

Наследование: `ICLDPowerCommand : ICLDCommand`.

Читайте параметры из строго типизированного объекта `cmd` (рекомендуется); доступны также «сырой» массив `cld[i]` (те же индексы, что и в sppx) и именованные обёртки `cmd["..."]`. В таблице перечислены члены команды, включая унаследованные от базовых интерфейсов; общие члены `ICLDCommand` (навигация, `CLD`, `CLDFile`, `TechOperation`, ...) описаны в разделе Модель доступа к CLData.

Член	Тип	Описание
`cmd.ID`	`int`	Числовой идентификатор устройства, для которого задаётся мощность.
`cmd.Value`	`double`	Значение мощности в процентах от некоторой константы, заданной либо в параметрах операции, либо подразумеваемой неявно.

Пример:

```
public override void OnPower(ICLDPowerCommand cmd, CLDArray cld)
{
    // cmd.Value - процент от номинальной мощности активного
исполнительного устройства
    double nominalCurrent = 120;                          // номинальный ток (A),
обычно берётся из параметров операции
    nc.Amperage.Show(nominalCurrent * cmd.Value / 100);   // фактический ток для
последующих перемещений
```

```
    nc.Block.Out();
}
```

Сопутствующие ссылки

- Модель доступа к CLData

PPFUN - Постпроцессорная функция

Отображение команды

Как команда отображается в списке команд CLData:

```
PPFUN
PPFUN(500)|STARTSUB(50)|ENDSUB(51)|CALLSUB(52)|REPSTART(53)|REPEND(54)|JUMP(55)|
      TECHINFO(58)|ENDTECHINFO(59)|WEDMConditions(56)|WEDMPassItem(57) {, a} {,
b} ...
```

Постпроцессорная функция PPFUN — это многоцелевая команда генератора постпроцессоров. Команда PPFUN может использоваться для представления различных видов данных и параметров обработки. Фактический тип команды PPFUN определяется первым параметром массива CLData — `CLD[1]` (`CLD.SubCode`). Остальные параметры массива CLD (`CLD[2]` – `CLD[257]`) могут иметь различный смысл в зависимости от типа команды PPFUN.

Доступ из sppx

Обработчик: `program PPFun`

Массив `cld` читается по индексу или по имени — в sppx по типизированному короткому имени (например, `cld.X`), в .NET по тому же имени в виде строкового ключа (например, `cld["X"]`).

Параметр	Массив CLD	Имя CLD	Описание
PPFUN(500) STARTSUB(50) ENDSUB(51) CALLSUB(52) REPSTART(53) REPEND(54) JUMP(55) TECHINFO(58) WEDMConditions(56) WEDMPassItem(57)	CLD[1]	CLD.SubCode	Тип постпроцессорной функции
a, b, c, d...	CLD[2] – CLD[257]		Список необязательных параметров. Структура списка параметров

Параметр	Массив CLD	Имя CLD	Описание
			определяется конкретной постпроцессорной функцией

Список типов постпроцессорных функций включает:

- PPFUN_PPFUN(500) - команда контроля стойкости инструмента
- PPFUN_STARTSUB(50) - команда начала подпрограммы УП
- PPFUN_ENDSUB(51) - команда конца подпрограммы УП
- PPFUN_CALLSUB(52) - команда вызова подпрограммы УП
- PPFUN_TECHINFO(58) - команда задаёт технологические параметры операции
- PPFUN_ENDTECHINFO(59) - конец технологической информации операции
- PPFUN_WEDMConditions(56) - команда задаёт условия электроэрозионной обработки
- Свойства прохода проволочной электроэрозионной обработки PPFun WEDMPassItem(57)

Пример (из постпроцессора Fanuc (30i)_Mill):

```
program PPFun
  case CLD.SubCode of
    50: begin ! StartSub
      OutBlock;
      Output ""
      Output "%"
      ProgNum = MainProgramID + CLD[2]; ProgNum@ = MaxReal
      if CLData$<>"" then
        Output OutStr$ + " (" + UPCASE(CLData$) + ")"
      OutBlock
      Interp_ = MaxReal; Interp_@ = Interp_
      Feed_ = MaxReal; Feed_@ = Feed_
    end
    51: begin ! EndSub
      OutBlock
      MSub = 99; MSub@ = MaxReal
      OutBlock
      Output "%"
    end
    52: begin ! CallSub
      OutBlock
      MSub = 98; MSub@ = MaxReal
      P = MainProgramID + CLD[2]; P@ = MaxReal
      OutBlock
    end
```

```
    58: begin ! TechInfo
      case cld[60] of
        1: OperationIsLathe = 0 !- фрезерование,
        2: OperationIsLathe = 1 !- точение,
        else OperationIsLathe = 3 ! прочее
      end
      CSOnCount = 0
      NeedKorDl = 0
      WasLoadTool = 0
      KorDl = 0; KorDl@ = KorDl
      H = cld[31]; H@ = H
      Zcycle = 0; Zcycle@ = Zcycle
      ZClear = 0; ZClear@ = ZClear
      QStep = 0; QStep@ = QStep
      !MC = 9; MC@ = Mc;
      Interp_ = MaxReal; Interp_@ = Interp_
      FEED_ = MaxReal; FEED_@ = FEED_
    end;
    59: begin ! EndTechInfo
      if Cmd.Int["PPFun(EndTechInfo).Enabled"]>0 then begin
        if CLDFile.CurrentFile <> CLDFile.FileCount - 1 then begin
          Output "M1"
        end
        Output " "
      end
      call CheckSwitchOffLocalCS
    end
  end
end
```

Доступ из .NET

Помимо базового `OnPPFun`, каркас вызывает обработчики, специфичные для подкодов:

```
public override void OnPPFun(ICLDPPFunCommand cmd, CLDArray cld)
// базовый
public override void OnStartTechOperation(ICLDTechOperation op, ICLDPPFunCommand
cmd, CLDArray cld) // TECHINFO(58)
public override void OnFinishTechOperation(ICLDTechOperation op, ICLDPPFunCommand
cmd, CLDArray cld)// ENDTECHINFO(59)
public override void OnStartNCSub(ICLDSub sub, ICLDPPFunCommand cmd, CLDArray cld)
// STARTSUB(50)
public override void OnFinishNCSub(ICLDSub sub, ICLDPPFunCommand cmd, CLDArray cld)
// ENDSUB(51)
```

```
public override void OnCallNCSub(ICLDSub sub, ICLDPPFunCommand cmd, CLDArray cld)
// CALLSUB(52)
```

Наследование: `ICLDPPFunCommand : ICLDCommand`.

Читайте параметры из строго типизированного объекта `cmd` (рекомендуется); также доступны «сырой» массив `cld[i]` (те же индексы, что и в sppx) и именованные помощники `cmd["..."]`. В таблице перечислены члены команды, включая унаследованные от её базовых интерфейсов; общие члены `ICLDCommand` (навигация, `CLD`, `CLDFile`, `TechOperation`, ...) описаны в разделе Модель доступа к CLData.

Свойство	Тип	Описание
`cmd.SubCode`	int	Числовой код подтипа PPFUN. Стандартные подтипы перечислены выше, но код может иметь любое произвольное значение, заданное разработчиком постпроцессора.
`cmd.TechInfo`	`IPPFunTechInfo`	Для `TECHINFO(58)`/`ENDTECHINFO(59)` — дополнительная информация об операции.

`IPPFunTechInfo` предоставляет `Enabled` (истина, если операция использует инструмент, отличный от предыдущего), `SetupStage` (`Index`, `Name`), `Tool` (`Number`, `ConnectorIndex`, `RevolverID`), `Workpiece` (`ConnectorID`) и `Part` (`Index`, `Number`, `Name`).

Пример — `OnPPFun` (из постпроцессора AgieCharmilles_WireEDM_DN):

```
public override void OnPPFun(ICLDPPFunCommand cmd, CLDArray cld)
{
    switch (cmd.SubCode)
    {
        case 58: // TechInfo
            nc.WriteLine("(Rapid level       = " + Str((double)cld[9]) + ")");
            nc.WriteLine("(Upper guide level = " + Str((double)cld[47]) + ")");
            nc.WriteLine("(Upper work level  = " + Str((double)cld[10]) + ")");
            nc.WriteLine("(Lower work level  = " + Str((double)cld[11]) + ")");
            nc.WriteLine("(Lower guide level = " + Str((double)cld[48]) + ")");
            nc.WriteLine("(Wire diameter     = " + Str((double)cld[27]) + ")");
```

```
            break;

        case 56: // WEDMConditions
            nc.H.Show(cld[4]); // код коррекции
            nc.HValue.Show(cld[5]); // значение коррекции
            nc.Block.Out();
            break;

        case 50: // STARTSUB
            nc.BlockN.Show(nc.ProgN + cld[2]);
            nc.Block.Out();
            break;

        case 51: // ENDSUB
            if (wireInserted)
            {
                BreakWire();
            }
            nc.MSub.Show(99);
            nc.Block.Out();
            nc.WriteLine();
            break;

        case 52: // CALLSUB
            nc.MSub.Show(98);
            nc.SubN.Show(nc.ProgN + cld[2]);
            nc.Block.Out();
            // холостой вызов подпрограммы
            // TODO: NCSub.Output(cld[2], 0)
            break;
    }
}
```

Сопутствующие ссылки

- Модель доступа к CLData

PPFUN CALLSUB(52) - Команда вызова подпрограммы УП

Команда вызова подпрограммы УП `PPFUN CALLSUB(52)` используется, когда управляющая программа содержит несколько разных подпрограмм УП и в некоторый момент выполнения программы требуется передать управление другой подпрограмме. Для этого постпроцессор команды `PPFUN CALLSUB(52)` должен вызвать один из операторов вывода подпрограммы УП. При вызове этих операторов обрабатываются некоторые или все команды подпрограммы УП. В противном случае технологические команды подпрограмм УП игнорируются.

Параметр `CLD[2]` содержит уникальный идентификатор подпрограммы УП.

Доступ из sppx

В sppx у PPFUN нет отдельного обработчика для каждого подкода. Все подкоды PPFUN обрабатываются в едином обработчике `program PPFun` с диспетчеризацией по `cld[1]` (`CLD.SubCode`); обработайте `CALLSUB`, добавив в него соответствующую ветку. Полный обработчик см. в разделе Постпроцессорная функция (PPFUN).

Пример (из постпроцессора Fanuc (30i)_Mill):

```
program PPFun
  case cld[1] of
      52: begin ! CallSub
        OutBlock
        MSub = 98; MSub@ = MaxReal
        P = MainProgramID + CLD[2]; P@ = MaxReal
        OutBlock
      end
  end
end
```

Доступ из .NET

Помимо базового `OnPPFun`, этот подкод вызывает специальный обработчик `OnCallNCSub`:

```
public override void OnCallNCSub(ICLDSub sub, ICLDPPFunCommand cmd, CLDArray cld)
```

Общая модель .NET (члены, индексация 1-based и 0-based) описана в разделе Постпроцессорная функция (PPFUN).

Пример — `OnCallNCSub` (из постпроцессора Heidenhain_Mill_iTNC530_DN):

```
public override void OnCallNCSub(ICLDSub cldSub, ICLDPPFunCommand cmd, CLDArray cld)
{
    // 25 ;Top plane
    // 26 CALL LBL101
    cldSub.Tag = 100 + cldSub.SubCode;
    nc.OutText($"CALL LBL{cldSub.Tag}");

    // Сбросить состояние основных регистров, чтобы вывести их при
первом присваивании.
    nc.Block.Reset(nc.X, nc.Y, nc.Z, nc.Feed, nc.MoveType, nc.RCompens, nc.S,
nc.MCoolant, nc.MSpindle);

    // Холостой прогон трансляции cldSub (с отключённой записью в nc-файл), чтобы
заполнить регистры актуальным состоянием.
    NCFiles.DisableOutput();
    cldSub.Translate(false);
    NCFiles.EnableOutput();
}
```

Сопутствующие ссылки

- Постпроцессорная функция (PPFUN)
- Модель доступа к CLData

PPFUN ENDSUB(51) - Команда конца подпрограммы УП

Команда конца подпрограммы УП `PPFUN ENDSUB(51)` вместе с командой начала подпрограммы УП `PPFUN STARTSUB(50)` используется для выделения технологических команд подпрограммы. Все технологические команды между PPFUN STARTSUB(50) и `PPFUN ENDSUB(51)` считаются командами подпрограммы УП. Команды основной программы обрабатываются последовательно, тогда как команды подпрограммы УП пропускаются. Команды подпрограммы УП транслируются только при вызове операторов вывода подпрограммы УП из одной из команд основной программы.

Параметр `CLD[2]` содержит уникальный идентификатор подпрограммы УП.

Доступ из sppx

В sppx у PPFUN нет отдельного обработчика для каждого подкода. Все подкоды PPFUN обрабатываются в едином обработчике `program PPFun` с диспетчеризацией по `cld[1]` (`CLD.SubCode`); обработайте `ENDSUB`, добавив в него соответствующую ветку. Полный обработчик см. в разделе Постпроцессорная функция (PPFUN).

Пример (из постпроцессора Fanuc (30i)_Mill):

```
program PPFun
  case cld[1] of
      51: begin ! EndSub
        OutBlock
        MSub = 99; MSub@ = MaxReal
        OutBlock
        Output "%"
      end
  end
end
```

Доступ из .NET

Помимо базового `OnPPFun`, этот подкод вызывает специальный обработчик `OnFinishNCSub`:

```
public override void OnFinishNCSub(ICLDSub sub, ICLDPPFunCommand cmd, CLDArray cld)
```

Общая модель .NET (члены, индексация 1-based и 0-based) описана в разделе Постпроцессорная функция (PPFUN).

Пример — `OnFinishNCSub` (из постпроцессора Heidenhain_Mill_iTNC530_DN):

```
public override void OnFinishNCSub(ICLDSub cldSub, ICLDPPFunCommand cmd,
CLDArray cld)
{
    // 454 LBL0
    nc.Block.Out();
    nc.OutText("LBL0");
}
```

Сопутствующие ссылки

- Постпроцессорная функция (PPFUN)
- Модель доступа к CLData
- `PPFUN STARTSUB(50)`

PPFUN ENDTECHINFO(59) - Конец технологической информации операции

Эта команда обычно является последней в списке технологических команд операции и служит для обнаружения конца операции. Набор параметров команды полностью совпадает с параметрами команды PPFUN TECHINFO(58). Она предоставляет технологическую информацию об операции CAM-системы.

Доступ из sppx

В sppx у PPFUN нет отдельного обработчика для каждого подкода. Все подкоды PPFUN обрабатываются в едином обработчике `program PPFun` с диспетчеризацией по `cld[1]` (`CLD.SubCode`); обработайте `ENDTECHINFO`, добавив в него соответствующую ветку. Полный обработчик см. в разделе Постпроцессорная функция (PPFUN).

Пример (из постпроцессора Fanuc (30i)_Mill):

```
program PPFun
  case cld[1] of
      59: begin ! EndTechInfo
        if Cmd.Int["PPFun(EndTechInfo).Enabled"]>0 then begin
          if CLDFile.CurrentFile <> CLDFile.FileCount - 1 then begin
            Output "M1"
          end
          Output " "
        end
        call CheckSwitchOffLocalCS
      end
  end
end
```

Доступ из .NET

Помимо базового `OnPPFun`, этот подкод вызывает специальный обработчик `OnFinishTechOperation`:

```
public override void OnFinishTechOperation(ICLDTechOperation op, ICLDPPFunCommand
cmd, CLDArray cld)
```

Общая модель .NET (члены, индексация 1-based и 0-based) описана в разделе Постпроцессорная функция (PPFUN).

Пример — `OnFinishTechOperation` (из постпроцессора Sinumerik (840D)_Mill_DN):

```
public override void OnFinishTechOperation(ICLDTechOperation op, ICLDPPFunCommand
cmd, CLDArray cld)
{
    if (cmd.TechInfo.Enabled) {
        if (cmd.CLDFile.Index != CLDProject.CLDFiles.FileCount - 1){
            nc.M.Show(1);
            nc.Block.Out();
        }
        nc.WriteLine();
    }
}
```

Сопутствующие ссылки

- Постпроцессорная функция (PPFUN)
- Модель доступа к CLData
- `PPFUN TECHINFO(58)`

PPFUN PPFUN(500) - Команда контроля стойкости инструмента

Команда `PPFUN(500)` генерируется, когда требуется сменить инструмент из-за его износа. В этом случае значение параметра `CLD[2]` — стойкость инструмента, выраженная в минутах.

Доступ из sppx

В sppx у PPFUN нет отдельного обработчика для каждого подкода. Все подкоды PPFUN обрабатываются в едином обработчике `program PPFun` с диспетчеризацией по `cld[1]` (`CLD.SubCode`); обработайте `PPFUN`, добавив в него соответствующую ветку. Полный обработчик см. в разделе Постпроцессорная функция (PPFUN).

Пример:

```
program PPFun
  case cld[1] of
    500: begin        ! PPFUN
      ! cld[2] содержит стойкость инструмента в минутах
    end
  end
end
```

Доступ из .NET

Для этого подкода нет специального обработчика — он доставляется через базовый `OnPPFun` (диспетчеризация по `cmd.SubCode`). См. раздел Постпроцессорная функция (PPFUN).

Сопутствующие ссылки

- Постпроцессорная функция (PPFUN)
- Модель доступа к CLData

PPFUN STARTSUB(50) - Команда начала подпрограммы УП

Команда начала подпрограммы УП `PPFUN STARTSUB(50)` вместе с командой конца подпрограммы УП `PPFUN ENDSUB(51)` используется для выделения технологических команд подпрограммы. Все технологические команды между `PPFUN STARTSUB(50)` и PPFUN ENDSUB(51) считаются командами подпрограммы УП. Команды основной программы обрабатываются последовательно, тогда как команды подпрограммы УП пропускаются. Команды подпрограммы УП транслируются только при вызове операторов вывода подпрограммы УП из одной из команд основной программы.

Параметр `CLD[2]` содержит уникальный идентификатор подпрограммы УП.

Доступ из sppx

В sppx у PPFUN нет отдельного обработчика для каждого подкода. Все подкоды PPFUN обрабатываются в едином обработчике `program PPFun` с диспетчеризацией по `cld[1]` (`CLD.SubCode`); обработайте `STARTSUB`, добавив в него соответствующую ветку. Полный обработчик см. в разделе Постпроцессорная функция (PPFUN).

Пример (из постпроцессора Fanuc (30i)_Mill):

```
program PPFun
  case cld[1] of
      50: begin ! StartSub
        OutBlock;
        Output ""
        Output "%"
        ProgNum = MainProgramID + CLD[2]; ProgNum@ = MaxReal
        if CLData$<>"" then
          Output OutStr$ + " (" + UPCASE(CLData$) + ")"
        OutBlock
        Interp_ = MaxReal; Interp_@ = Interp_
        Feed_ = MaxReal; Feed_@ = Feed_
      end
  end
end
```

Доступ из .NET

Помимо базового `OnPPFun`, этот подкод вызывает специальный обработчик `OnStartNCSub`:

```
public override void OnStartNCSub(ICLDSub sub, ICLDPPFunCommand cmd, CLDArray cld)
```

Общая модель .NET (члены, индексация 1-based и 0-based) описана в разделе Постпроцессорная функция (PPFUN).

Пример — `OnStartNCSub` (из постпроцессора Heidenhain_Mill_iTNC530_DN):

```
public override void OnStartNCSub(ICLDSub cldSub, ICLDPPFunCommand cmd,
CLDArray cld)
{
    //433 LBL101
    cldSub.Tag = 100 + cldSub.SubCode;
    nc.WriteLine();
    nc.OutText($"LBL{cldSub.Tag}");
}
```

Сопутствующие ссылки

- Постпроцессорная функция (PPFUN)
- Модель доступа к CLData
- `PPFUN_ENDSUB(51)`

PPFUN TECHINFO(58) - Команда технологических параметров

Команда технологических параметров `PPFUN TECHINFO(58)` передаёт дополнительную технологическую информацию об операции CAM-системы. Команда `PPFUN TECHINFO(58)` генерируется в начале списка технологических команд операции.

Параметры

Массив CLD	Параметр	Описание переменной	Группа
1	SubCode	58 (TechInfo)	
2	2	Версия CLData	
3	3	Минимальное значение координаты X	Габарит траектории
4	4	Минимальное значение координаты Y	
5	5	Минимальное значение координаты Z	
6	6	Максимальное значение координаты X	
7	7	Максимальное значение координаты Y	
8	8	Максимальное значение координаты Z	
9	9	Уровень безопасной плоскости (уровень ускоренного	Уровни

Массив CLD	Параметр	Описание переменной	Группа
		перемещения для электроэрозионных операций)	
10	10	Верхний уровень обработки	
11	11	Нижний уровень обработки	
12	12	Припуск	Заготовка
13	13	Минимальное значение координаты X	
14	14	Минимальное значение координаты Y	
15	15	Минимальное значение координаты Z	
16	16	Максимальное значение координаты X	
17	17	Максимальное значение координаты Y	
18	18	Максимальное значение координаты Z	
19	19	Допуск CLData (число десятичных знаков)	Допуск

Массив CLD	Параметр	Описание переменной	Группа
20	20	Единицы измерения: 0 – мм; 1 – дюймы.	
21	21	Минимальная длина дуги	
22	22	Отклонение от заготовки	
23	23	Отклонение в заготовку	
24	24	Припуск операции	
25	25	Тип инструмента: 0 – цилиндрическая 1 – сферическая 2 – тороидальная 3 – двухрадиусная 4 – ограниченная двухрадиусная 5 – коническая фреза 6 – ограниченная коническая фреза 7 – гравёр 8 – сверло 9 – составной инструмент 10 – фреза с отрицательным радиусом 11 – резец 12 – метчик 13 – центровочное сверло 14 – зенковка 15 – цековка 16 – развёртка 17 – резьбовое сверло 100 – наружный резец 101 – расточной резец 102	Инструмент

Массив CLD	Параметр	Описание переменной	Группа
		– наружный резьбовой резец 103 – внутренний резьбовой резец 104 – наружный канавочный резец 105 – внутренний канавочный резец 106 – торцовый канавочный резец	
26	26	Номер инструмента	
27	27	Диаметр инструмента (диаметр проволоки для электроэрозионных операций)	
28	28	Длина рабочей части	
29	29	Угол рабочей части (угол конуса в радианах)	
30	30	Радиус скругления	
31	31	Номер коррекции на длину. Устанавливается в 0, если коррекция на длину выключена	
32	32	Номер коррекции на радиус. Устанавливается в 0, если коррекция на радиус выключена	

Массив CLD	Параметр	Описание переменной	Группа
33	33	Значение коррекции на длину	
34	34	Значение коррекции на радиус	
35	35	Точка привязки осевого инструмента: 0 – Центр, 1 – Пользовательская, 2 – Торец. Точка привязки токарного инструмента: 0 – Слева, 1 – Справа, 2 – Сверху, 3 – Снизу, 4 – Сверху слева, 5 – Сверху справа, 6 – Снизу слева, 7 – Снизу справа, 8 – Центр, 9 – Пользовательская.	
36	36	Частота вращения шпинделя (об/мин)	Скорость
37	37	Ускоренная подача (мм/мин или дюйм/мин)	
38	38	Рабочая подача	
39	39	Подача врезания	
40	40	Подача подхода	
41	41	Подача отвода	
42	42	Следующая подача	

Массив CLD	Параметр	Описание переменной	Группа
43	43	Подача возврата	
44	44	Длина траектории	Свойства траектории
45	45	Время обработки операции в минутах	
46	46	Общее время техпроцесса в минутах	
47	47	Верхний уровень направляющей электроэрозионного станка	Уровни
48	48	Нижний уровень направляющей электроэрозионного станка	
49	49	Рабочее время операции в минутах	Свойства траектории
51	51	Ось инструмента: 1 – X, 2 – Y, 3 – Z. (Точение – 2, Токарное сверление – 1, Фрезерование – 3)	
52	52	Координата X	Точка смены инструмента
53	53	Координата Y	
54	54	Координата Z	
55	55	Координата X	Опорная точка

Массив CLD	Параметр	Описание переменной	Группа
56	56	Координата Y	
57	57	Координата Z	
58	58	Номер активного сопла: 0 – СОЖ выключена, 1 – полив, 2 – туман, 3 – через инструмент.	
60	60	Технологическая группа операции: 0 – не задана, 1 – фрезерование, 2 – точение, 3 – вспомогательная, 4 – электроэрозионная.	
61	61	0 – команда PPFUN не активирована 1 – команда PPFUN активирована	
62	62	Кинематика робота: 0 – положение 3-го сустава не зависит от положения 2-го сустава 1 – 2-й и 3-й суставы связаны параллелограммным механизмом	Кинематика робота
63	63	Номер детали, явно заданный пользователем в параметрах детали CAM-системы	Деталь

Массив CLD	Параметр	Описание переменной	Группа
64	64	Код вектора направления смещения вершины токарного инструмента (0-8): 0: X=0, Y=0; 1: X=+1, Y=+1; 2: X=-1, Y=+1; 3: X=-1, Y=-1; 4: X=+1, Y=-1; 5: X=+1, Y=0; 6: X=0, Y=+1; 7: X=-1, Y=0; 8: X=0, Y=-1;	Код вершины инструмента
65	65	Номер системы координат заготовки (G54, G55, G54.2 и т. д.) операции	Номер системы координат
80	80	Шаг вдоль оси инструмента для многослойной обработки (только для операции 2D-контур)	Слои
81	81	Припуск чистового слоя вдоль оси инструмента (только для операции 2D-контур)	Слои

Параметры, доступные через оператор cmd

Тип	Описание	Описание вложенного поля
TCLDPPFun: ComplexType	Команда, определяющая технологические параметры операции	

Тип	Описание	Описание вложенного поля
PPFun: Array, Key="RecordID"	cmd.Ptr["PPFun"] - массив структур разных типов для хранения параметров команды PPFun	
TPPFunTechInfoParameters: ComplexType	cmd.Ptr["PPFun"].Item[1] или cmd.Ptr["PPFun(TechInfo)"] - отдельный элемент массива PPFun. Содержит информацию о технологических параметрах операции. Доступ к элементам массива возможен как по индексу, так и по ключевому полю RecordID. Для команды определения технологических параметров операции всегда имеет значение "TechInfo".	
RecordID: String	cmd.Str["PPFun(TechInfo).RecordID"] - идентификатор типа постпроцессорной функции. В данном случае всегда имеет значение "TechInfo".	
RecordCode: Integer	cmd.Int["PPFun(TechInfo).RecordCode"] - код, идентифицирующий тип постпроцессорной функции. В данном случае всегда имеет значение "58".	
Operation: Array	cmd.Ptr["PPFun(TechInfo).Operation"] - почти полный список параметров технологической операции. Дочерними элементами являются	

Тип	Описание	Описание вложенного поля
	разнообразные параметры операции: тип операции, её имя, группа, свойства инструмента, припуски, допуски расчёта, методы подхода и отхода, уровни и шаг обработки и т. д. Конкретный набор параметров зависит от типа операции. Тип операции можно получить как cmd.Ptr["PPFun(TechInfo).Operation(1)"].Name.	
SetupStage: ComplexType	cmd.Ptr["PPFun(TechInfo).SetupStage"] - информация об установе, в котором находится операция.	
	Index: Integer	Индекс установа в списке установов проекта CAM-системы
	Name: String	Имя группы установа в проекте CAM-системы
Tool: ComplexType	cmd.Ptr["PPFun(TechInfo).Tool"] - параметры, относящиеся к инструменту, используемому в технологической операции.	
	ID: Integer	cmd.Int["PPFun(TechInfo).Tool.ID"] - идентификатор (номер) инструмента
	HolderID: Integer	cmd.Int["PPFun(TechInfo).Tool.HolderID"] - идентификатор (номер) державки инструмента

Тип	Описание	Описание вложенного поля
	RevolverID: String	cmd.Str["PPFun(TechInfo).Tool.RevolverID"] - строковый идентификатор револьверной головки.
Workpiece: ComplexType	cmd.Ptr["PPFun(TechInfo).Workpiece"] - параметры, относящиеся к заготовке, используемой в технологической операции.	
	HolderID: String	cmd.Str["PPFun(TechInfo).Workpiece.HolderID"] - строковый идентификатор державки заготовки.
Part: ComplexType	cmd.Ptr["PPFun(TechInfo).Part"] - информация о детали, обрабатываемой этой операцией	
	Index: Integer	Индекс детали в списке деталей проекта CAM-системы
	Number: Integer	Номер детали, явно заданный пользователем в параметрах детали CAM-системы
	Name: String	Имя детали (имя группы), заданное в CAM-системе

Ниже приведён пример доступа к свойствам операции через оператор cmd.

```
program PPFun
  if cld[1]=58 then begin ! TechInfo
    call AnalysePPFun
  end
end
sub AnalysePPFun
```

```
  OpType: String
  OpGroup: Integer
  ! Строковое имя типа операции: TSTRoughingWaterlineOp, TST2DContouringOp,
TSTDrillOp, LatheRoughOp...
  OpType = cmd.Ptr["PPFun(TechInfo).Operation(1)"].Name
  ! OperationGroup:
  ! 1 - фрезерование
  ! 2 - точение
  ! 3 - вспомогательная
  ! 4 - проволочная электроэрозионная
  OpGroup = cmd.Int["PPFun(TechInfo).Operation(1).OperationGroup"]
  Output "; Operation type = " + OpType
  Output "; Operation comment = " + cmd.Str["PPFun(TechInfo).Operation(1).Comment"]
  if OpGroup=1 then begin ! фрезерование
    Output "; Tool overhang = " +
Str(cmd.Flt["PPFun(TechInfo).Operation(1).ToolSection.Tools(1).AxialOverhang"])
    if OpType="TSTRoughingWaterlineOp" then begin
      Output "; Axial stock = " +
Str(cmd.Flt["PPFun(TechInfo).Operation(1).Stocks.Axial"])
      Output "; Radial stock = " +
Str(cmd.Flt["PPFun(TechInfo).Operation(1).Stocks.Radial"])
    end
  end else if OpGroup=2 then begin ! точение
    Output "; Tool X overhang = " +
Str(cmd.Flt["PPFun(TechInfo).Operation(1).ToolSection.Tools(1).Overhang.Value.X"])
    Output "; Tool Y overhang = " +
Str(cmd.Flt["PPFun(TechInfo).Operation(1).ToolSection.Tools(1).Overhang.Value.Y"])
    Output "; Tool Z overhang = " +
Str(cmd.Flt["PPFun(TechInfo).Operation(1).ToolSection.Tools(1).Overhang.Value.Z"])
    Output "; Axial stock = " +
Str(cmd.Flt["PPFun(TechInfo).Operation(1).LatheStock.AxialValue"])
    Output "; Radial stock = " +
Str(cmd.Flt["PPFun(TechInfo).Operation(1).LatheStock.RadialValue"])
  end
subend
```

Доступ из .NET

Помимо базового `OnPPFun`, этот подкод вызывает специальный обработчик `OnStartTechOperation`:

```
public override void OnStartTechOperation(ICLDTechOperation op, ICLDPPFunCommand
cmd, CLDArray cld)
```

Общая модель .NET (члены, индексация 1-based и 0-based) описана в разделе Постпроцессорная функция (PPFUN).

Чтение XML-свойств операции по имени

Для `TECHINFO`/`ENDTECHINFO`, помимо фиксированных параметров выше, команда обеспечивает доступ к XML-свойствам операции по имени через помощники `cmd`:

> Предупреждение. Это прямое чтение внутренних структур данных CAM-системы. Их расположение и именование не гарантированно сохраняются между версиями. Используйте это только как крайнее средство, когда нужное значение не предоставляется через публичный API CLData выше. По возможности предпочитайте документированные параметры и типизированные члены.

Индекс элемента массива различается между средами — 1-based в sppx, 0-based в .NET:

- sppx: `cmd.Ptr["PPFun(TechInfo).Operation(1)"]`
- .NET: `cmd.Ptr["PPFun(TechInfo).Operation(0)"]`

Пример (из постпроцессора Heidenhain (iTNC530)_Mill — sppx, 1-based индекс элемента):

```
! глубоко вложенное свойство операции, доступное через команду TechInfo операции
MinDepth =
cmd.Flt["PPFun(TechInfo).Operation(1).ChipBreaking.DepthDegression.MinStepPercent"]
```

Чтобы узнать точное имя свойства: откройте инспектор свойств операции, найдите свойство, нажмите кнопку ... справа в его строке и выберите «Скопировать имя свойства в буфер обмена»; либо изучите XML операции. Правила именования и полная модель доступа описаны в разделе Использование XML-свойств из кода.

Пример — `OnStartTechOperation` (из постпроцессора Heidenhain_Mill_iTNC530_DN):

```
public override void OnStartTechOperation(ICLDTechOperation op, ICLDPPFunCommand
cmd, CLDArray cld)
{
    nc.WriteLine();
    nc.OutText("; " + Transliterate(op.Comment));

    // 17 TOOL CALL 1 Z S159 DL+0 DR+0
    // 18 TOOL DEF 2
    // 19 L Z+0 FMAX M91
    if (op.Tool.Command!=null && op.Tool.Command.ToolChanged) { // если у операции
есть команда загрузки инструмента
```

```
        double s = 0;
        if (op.SpindleCommand!=null)
            s = Abs(op.SpindleCommand.RPMValue);
        nc.OutText($"TOOL CALL {op.Tool.Number} {GetToolPlaneName(op.Tool.Command)}
{nc.S.ToString(s)} DL+0 DR+0");

        int nextTool = FindNextOpToolNumber(op);
        if (nextTool>0)
            nc.OutText($"TOOL DEF {nextTool}");

        if (firstToolNumber<0)
            firstToolNumber = op.Tool.Number;
        nc.OutText("L Z+0 FMAX M91");
    }

}
```

Сопутствующие ссылки

- Постпроцессорная функция (PPFUN)
- Модель доступа к CLData
- PPFUN ENDTECHINFO(59)
- Использование XML-свойств из кода

PPFUN WEDMConditions(56) - Команда задания условий электроэрозионной обработки

Команда задания условий электроэрозионной обработки `PPFUN WEDMConditions(56)` генерируется электроэрозионными операциями CAM-системы. Эта команда используется для передачи постпроцессору таблицы условий обработки электроэрозионной операции. Каждая строка таблицы условий обработки представляет условия обработки одного прохода. Таким образом, число строк в таблице равно числу используемых проходов. Каждая строка таблицы передаётся отдельной командой `PPFUN WEDMConditions(56)`. Поэтому список технологических команд каждой электроэрозионной операции открывается списком команд `WEDMConditions(56)`, число которых равно числу строк в таблице условий электроэрозионной операции.

Параметры

Параметры	Массив CLD	Имя CLD	Описание
Подкод команды	CLD[1]	CLD.SubCode	WEDMCONDITIONS(56) – строка таблицы условий обработки электроэрозионной операции
Порядковый номер прохода	CLD[2]		Порядковые номера проходов обычно назначаются последовательно, начиная с 1. Однако нумерация может начинаться не с 1, в зависимости от настроек операции в CAM-системе

Параметры	Массив CLD	Имя CLD	Описание
Код условий прохода	CLD[3]		Код условий задаёт условия обработки для данного прохода. Смена кода условий по ходу программы выполняется командой FEDRAT.
Код коррекции	CLD[4]		Код коррекции определяет номер корректора, который следует использовать для задания значения коррекции для данного прохода. Переключение номера корректора выполняется командой CUTCOM.
Значение коррекции	CLD[5]		Значение коррекции для данного номера корректора
Значение подачи	CLD[6]		Значение подачи проволоки в мм/мин (дюйм/мин)
Дополнительные параметры	CLD[100] – CLD[256]		Список дополнительных параметров для каждого прохода. Число дополнительных параметров индивидуально для каждого постпроцессора.

Доступ из sppx

В sppx у PPFUN нет отдельного обработчика для каждого подкода. Все подкоды PPFUN обрабатываются в едином обработчике `program PPFun` с диспетчеризацией по `cld[1]` (`CLD.SubCode`); обработайте `WEDMConditions`, добавив в него соответствующую ветку. Полный обработчик см. в разделе Постпроцессорная функция (PPFUN).

Пример:

```
program PPFun
  case cld[1] of
    56: begin        ! WEDMConditions
      ! строка таблицы условий электроэрозионной обработки
    end
  end
end
```

Доступ из .NET

Для этого подкода нет специального обработчика — он доставляется через базовый `OnPPFun` (диспетчеризация по `cmd.SubCode`). См. раздел Постпроцессорная функция (PPFUN).

Сопутствующие ссылки

- Постпроцессорная функция (PPFUN)
- Модель доступа к CLData

PPFUN WEDMPassItem(57) - Свойства прохода проволочной электроэрозионной обработки

Команда свойств прохода проволочной электроэрозионной обработки WEDMPassItem(57) генерируется электроэрозионными операциями CAM-системы. Она используется для передачи постпроцессору информации о том, какому контуру и проходу траектории принадлежат вложенные команды CLData. Она появляется в начале перемещений по контуру и перед концом перемещений по контуру для каждого отдельного прохода.

Параметры

Параметры	Массив CLD	Имя CLD	Описание
Подкод команды	CLD[1]	CLD.SubCode	WEDMPassItem(57) – тип команды PPFun — свойства прохода проволочной электроэрозионной обработки.
On/Off	CLD[2]		Флаг начала-конца контура: On(71) - начало контура, Off(72) - конец контура.
Тип контура	CLD[3]		Тип контура — проход общего типа (0) или проход выборки без сердцевины (1).
Индекс элемента	CLD[4]		Индекс этого контура в списке элементов задания операции.
Индекс прохода	CLD[5]		Индекс прохода смещения
Тип прохода	CLD[6]		Тип прохода: 0 - черновой, 1 -

Параметры	Массив CLD	Имя CLD	Описание
			черновой с перемычкой, 2 - перемычка, 3 - чистовой совместный, 4 - чистовой раздельный.

Доступ из sppx

В sppx у PPFUN нет отдельного обработчика для каждого подкода. Все подкоды PPFUN обрабатываются в едином обработчике `program PPFun` с диспетчеризацией по `cld[1]` (`CLD.SubCode`); обработайте `WEDMPassItem`, добавив в него соответствующую ветку. Полный обработчик см. в разделе Постпроцессорная функция (PPFUN).

Пример:

```
program PPFun
  case cld[1] of
    57: begin        ! WEDMPassItem
      ! свойства прохода проволочной электроэрозионной обработки
    end
  end
end
```

Доступ из .NET

Для этого подкода нет специального обработчика — он доставляется через базовый `OnPPFun` (диспетчеризация по `cmd.SubCode`). См. раздел Постпроцессорная функция (PPFUN).

Сопутствующие ссылки

- Постпроцессорная функция (PPFUN)
- Модель доступа к CLData

PPRINT - Печать постпроцессора

Отображение команды

Как команда отображается в списке команд CLData:

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

Команда PPRINT передаёт текстовую информацию из CAM-системы постпроцессору. Текст технологической команды помещается в предопределённую переменную CLDATA$, после чего этот текст может быть выведен в окно оператора как комментарий, поясняющий работу постпроцессора.

Теги траектории

Помимо произвольного текста, PPRINT также используется для тегирования ключевых точек траектории. Такой тег - это PPRINT, текст которого начинается с `#`, за которым следует тип тега и, необязательно, значение:

```
PPRINT "#KeyPoint: StartCutting"
PPRINT "#Approach:[Rule 1]=C; Z10; X; Z"
PPRINT "#Return:[Rule 1]=Z10; X; Z; C"
```

CAM-система вставляет эти теги для аннотирования траектории - например, начало резания и правило, по которому построено перемещение подвода или отвода. Постпроцессор может распознать тег по ведущему символу `#`, отреагировать на него при необходимости или просто пропустить, чтобы метка не попала в управляющую программу. См. раздел Структура листинга CLData о том, где эти теги встречаются внутри реальной операции.

Доступ из sppx

Обработчик: `program PPrint`

Пример (из постпроцессора Fanuc (30i)_Mill):

```
program PPrint
  print cldata$
end
```

Доступ из .NET

Обработчик:

```
public override void OnPrint(ICLDPrintCommand cmd, CLDArray cld)
```

Наследование: `ICLDPrintCommand : ICLDCommand`.

Читайте параметры из строго типизированного объекта `cmd` (рекомендуется); также доступны «сырой» массив `cld[i]` (те же индексы, что и в sppx) и именованные помощники `cmd["..."]`. В таблице перечислены члены команды, включая унаследованные от базовых интерфейсов; общие члены `ICLDCommand` (навигация, `CLD`, `CLDFile`, `TechOperation`, ...) описаны в разделе Модель доступа к CLData.

Член	Тип	Описание
`cmd.Text`	`string`	Произвольная текстовая информация, переданная из CAM.

Пример:

```
public override void OnPrint(ICLDPrintCommand cmd, CLDArray cld)
{
    nc.OutWithN("(" + cmd.Text + ")");   // передать текст CAM в управляющую программу (здесь - как комментарий)
}
```

Сопутствующие ссылки

- Модель доступа к CLData

RAPID - Ускоренная подача

Отображение команды

Как команда отображается в списке команд CLData:

```
RAPID N n
```

Команда RAPID активирует режим ускоренного перемещения (G-код G00). Все последующие перемещения выполняются в ускоренном режиме до получения команды изменения подачи FEDRAT.

Доступ из sppx

Обработчик: `program Rapid`

Параметр	Массив CLD	Имя CLD	Описание
n	CLD[1]	CLD.N	Значение ускоренной подачи

Пример (из постпроцессора Fanuc (30i)_Mill):

```
program Rapid
  if IsRotaryInterpolation>0 then begin
    INTERP_ = 1
    Feed_ = 5000
    GFeed = 94
  end else begin
    if INTERP_ > 0 then INTERP_ = 0    ! G0
  end
  RapFr=cld[1]
end
```

Доступ из .NET

Обработчики:

```
public override void OnMoveVelocity(ICLDMoveVelocityCommand cmd, CLDArray cld)   //
общий обработчик скорости (FEDRAT и RAPID)
public override void OnRapid(ICLDRapidCommand cmd, CLDArray cld)   //
ускоренная подача
```

Наследование: `ICLDRapidCommand : ICLDMoveVelocityCommand : ICLDCommand`.

Читайте параметры из строго типизированного объекта `cmd` (рекомендуется); также доступны «сырой» массив `cld[i]` (те же индексы, что и в sppx) и именованные помощники `cmd["..."]`. В таблице перечислены члены команды, включая унаследованные от базовых интерфейсов; общие члены `ICLDCommand` (навигация, `CLD`, `CLDFile`, `TechOperation`, ...) описаны в разделе Модель доступа к CLData.

Член	Тип	Описание
`cmd.IsRapid`	`bool`	Содержит "true", если команда является командой RAPID, иначе - "false".
`cmd.FeedKind`	`CLDFeedKind`	Вид подачи (рабочая, ускоренная, подвод, возврат, чистовая и т. д.). Возможные значения: `Working` (0) - Вид подачи для базовых рабочих перемещений. `Rapid` (1) - Вид подачи для ускоренных перемещений. `First` (2) - Вид подачи для рабочих перемещений на первом (черновом) проходе. `Engage` (4) - Вид подачи для перемещений, входящих в первую точку обработки (резания). `Retract` (8) - Вид подачи для перемещений, покидающих последнюю точку обработки (резания). `Plunge` (16) - Вид подачи для перемещений, вертикально врезающихся в зону обработки. `Finish` (32) - Вид подачи для рабочих перемещений на последнем (чистовом)

Член	Тип	Описание
		проходе. `Next` (64) - Вид подачи для коротких связующих перемещений. `Return` (128) - Вид подачи для перемещений, уводящих инструмент от места обработки. `Approach` (256) - Вид подачи для перемещений, приближающих инструмент к месту обработки. `Rapid5D` (512) - Значение вида подачи, зарезервированное для внутреннего использования. `TransitionOnSafe` (1024) - Вид подачи для перемещений, выполняемых в безопасной зоне. `ApproachFromSafe` (2048) - Вид подачи для перемещений из безопасной зоны к месту обработки. `ReturnToSafe` (4096) - Вид подачи для перемещений от места обработки в безопасную зону. `LongNext` (8192) - Вид подачи для длинных связующих перемещений. `ThreadPitch` (16384) - Вид подачи для перемещений при нарезании резьбы. `Overload` (1073741824) - Значение вида подачи, зарезервированное для внутреннего использования.

Член	Тип	Описание
`cmd.FeedCode`	`int`	Числовое представление флага "FeedKind", определяющего вид подачи (рабочая, ускоренная, подвод, возврат, чистовая и т. д.). Для проволочной электроэрозионной обработки содержит числовой код строки в таблице условий обработки.
`cmd.FeedUnits`	`CLDFeedUnits`	Единицы измерения подачи (мм/мин, мм/об и т. д.) Возможные значения: `MPM` (0) - Текущие линейные единицы (мм или дюйм) в минуту. `MPR` (1) - Текущие линейные единицы (мм или дюйм) на оборот. `ConditionCode` (2) - Числовой код строки в таблице условий обработки (для проволочной электроэрозионной обработки).
`cmd.FeedValue`	`double`	Значение подачи.

Пример - `OnMoveVelocity` (из постпроцессора Fanuc (30i)_Mill_DN):

```
public override void OnMoveVelocity(ICLDMoveVelocityCommand cmd, CLDArray cld)
{
    if (cmd.IsRapid) {
        nc.GInterp.v = 0;
    } else {
        if (nc.GInterp == 0)
            nc.GInterp.v = 1;
        nc.F.v = cmd.FeedValue;
        if (cmd.FeedUnits==CLDFeedUnits.MPR)
```

```
            nc.GFeed.v = 95;
        else
            nc.GFeed.v = 94;
        if (IsFirstFeed)
        {
            nc.F.Show();
            nc.GFeed.Show();
            IsFirstFeed = false;
        }
    }
}
```

Сопутствующие ссылки

- Модель доступа к CLData

ROTABL - Поворот стола

> Устарела. Эта команда устарела - текущая версия CAM-системы не формирует её автоматически. Она документирована здесь для постпроцессоров, которые всё ещё обрабатывают устаревшие CLData.

Отображение команды

Как команда отображается в списке команд CLData:

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

Команда ROTABL передаёт позиционирование поворотной оси станка. Эта команда устарела и поддерживается для обратной совместимости. Для обработки поворотных осей используйте команду MULTIGOTO.

Доступ из sppx

Обработчик: `program Rotabl`

или

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

Параметр	Массив CLD	Имя CLD	Описание
ABS(0) INCR(66)	CLD[1]	CLD.Incr	Тип поворота: 0 (ABS) абсолютный, 66 (INCR) приращение
ab	CLD[2]	CLD.AB	Значение угла поворота стола или значение приращения
CCLW(59) CLW(60)	CLD[3]	CLD.CCLW	Направление вращения: CCLW - против часовой стрелки; CLW - по часовой стрелке.

Параметр	Массив CLD	Имя CLD	Описание
XY(33) YZ(37) XZ(41)	CLD[4]	CLD.Plane	Плоскость

Используйте ISO.A, ISO.B, ISO.C для создания подобных ISO-постпроцессоров с шаблонами. Система автоматически заполняет эти переменные значениями элементов массива CLD:

Значение параметров CLD	Значение ISO
CLD[4] = 33 CLD[1] = 0 CLD[3] = 59	ISO.C = CLD[2]
CLD[4] = 33 CLD[1] = 0 CLD[3] = 60	ISO.C = -CLD[2]
CLD[4] = 33 CLD[1] = 66	ISO.C = CLD[2]
CLD[4] = 41 CLD[1] = 0 CLD[3] = 59	ISO.B = CLD[2]
CLD[4] = 41 CLD[1] = 0 CLD[3] = 60	ISO.B = -CLD[2]
CLD[4] = 41 CLD[1] = 66	ISO.B = CLD[2]
CLD[4] = 37 CLD[1] = 0 CLD[3] = 59	ISO.A = CLD[2]
CLD[4] = 37 CLD[1] = 0 CLD[3] = 60	ISO.A = -CLD[2]
CLD[4] = 37 CLD[1] = 66	ISO.A = CLD[2]

Пример (из постпроцессора HAAS (SL30)_TurnMill):

```
program RoTabl
!  using RotAng,RotAngStep,RotPlan initialize in PARTNO
!  if cld[3] = 59 then sing = 1 else sign = -1
!  if cld[1] = 0 then RotAng = cld[2]*sign
!  if cld[1] = 66 then RotAngStep = cld[2]*sign
!  if cld[4] = 33 then RotPlan = 17        ! плоскость поворота стола C, XY
!  else if cld[4] = 37 then RotPlan = 18   ! плоскость поворота стола B, XZ
!  else if cld[4] = 41 then RotPlan = 19   ! плоскость поворота стола A, YZ
end
```

Сопутствующие ссылки

- Модель доступа к CLData

SAFPOS - Точка смены инструмента

> Устарела. Эта команда является устаревшей - текущая версия CAM-системы не генерирует её автоматически. Она документирована здесь для постпроцессоров, которые всё ещё обрабатывают устаревший CLData.

Отображение команды

Как команда отображается в списке команд CLData:

```
SAFPOS X x, Y y, Z z, N n
```

Команда SAFPOS определяет координаты точки, в которую будет позиционирован центр инструмента для смены инструмента (команда LOADTL). По умолчанию смена инструмента выполняется в текущей точке.

Доступ из sppx

Обработчик: `program SafPos`

Параметр	Массив CLD	Имя CLD	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 на станке с ЧПУ

Сопутствующие ссылки

- Модель доступа к CLData

SELWORKPIECE - Выбор активной державки заготовки

Отображение команды

Как команда отображается в списке команд CLData:

```
SELWORKPIECE WorkpieceConnectorID
```

Используйте команду SELWORKPIECE для указания активной державки заготовки. Державка заготовки - это узел станка, используемый для закрепления заготовки. Для многошпиндельных токарных станков эта команда указывает активный шпиндель. Для фрезерных станков державкой заготовки может быть, например, палета. Все команды обработки будут относиться к заготовке, закреплённой в активной державке.

Доступ из sppx

Обработчик: `program SelWorkpiece`

Единственный параметр команды SELWORKPIECE - строковый идентификатор державки заготовки. Этот идентификатор определяется схемой станка CAM-системы. Значение параметра сохраняется в предопределённой переменной CLDATA$. Ниже приведён пример программы, обрабатывающей эту команду.

```
program SelWorkpiece
  if CLData$="MainSpindle" then
    Output "SETMS(1)" ! Активировать главный шпиндель
  else
    Output "SETMS(2)" ! Активировать контршпиндель
end
```

Доступ из .NET

Обработчик:

```
public override void OnSelWorkpiece(ICLDSelWorkpieceCommand cmd, CLDArray cld)
```

Наследование: `ICLDSelWorkpieceCommand : ICLDCommand`.

Читайте параметры из строго типизированного объекта `cmd` (рекомендуется); доступны также «сырой» массив `cld[i]` (те же индексы, что и в sppx) и именованные обёртки `cmd["..."]`. В

таблице перечислены члены команды, включая унаследованные от базовых интерфейсов; общие члены `ICLDCommand` (навигация, `CLD`, `CLDFile`, `TechOperation`, ...) описаны в разделе Модель доступа к CLData.

Член	Тип	Описание
`cmd.ConnectorID`	`string`	Текстовый идентификатор державки заготовки (как он определён в схеме станка).

Пример:

```
public override void OnSelWorkpiece(ICLDSelWorkpieceCommand cmd, CLDArray cld)
{
    activeWorkpieceConnector = cmd.ConnectorID;   // запомнить, к какой державке заготовки относятся следующие команды
}
```

Сопутствующие ссылки

- Модель доступа к CLData

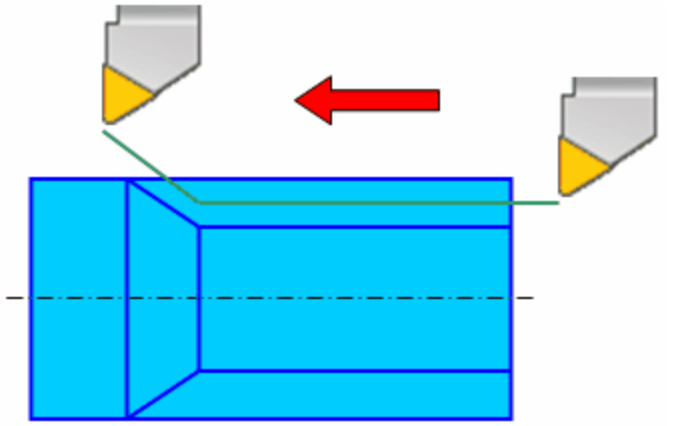
SINGLETHREAD - Однопроходная резьба

Отображение команды

Как команда отображается в списке команд CLData:

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

Команда SINGLETHREAD используется при непрерывном нарезании цилиндрической или конической резьбы с постоянным шагом токарным резцом. Обычно этому соответствуют G-коды G32 или G33 в управляющей программе.



При активации режима нарезания резьбы включается режим синхронизации перемещения инструмента и вращения шпинделя. Все последующие перемещения инструмента выполняются в этом режиме до получения команды изменения подачи (FEDRAT) или команды ускоренного перемещения (RAPID). Если инструмент перемещается вдоль оси шпинделя, формируется цилиндрическая резьба. Если инструмент перемещается вдоль оси шпинделя и одновременно перпендикулярно ей, обрабатывается коническая резьба. Возможна обработка торцевой резьбы перемещением инструмента только перпендикулярно оси шпинделя.

Параметры команды SINGLETHREAD определяют ориентацию резьбы, шаг резьбы и начальный угол шпинделя.

Шаг резьбы можно задать двумя способами: значением шага или количеством витков резьбы на единицу длины.

Начальный угол шпинделя задаётся в градусах и определяет угол, откладываемый от нулевого сигнала датчика шпинделя перед началом нарезания резьбы. Для нарезания многозаходной резьбы задавайте разные значения угла для разных проходов.

Доступ из sppx

Обработчик: `program SingleThread`

Параметр	Массив CLD	Имя CLD	Описание
OD(0) ID(1) FACE(2)	CLD[1]	CLD.Orientation	Ориентация резьбы: OD(0) - наружная, ID(1) - внутренняя, FACE(2) - торцевая.
STEP(0) COUNT(1)	CLD[2]	CLD.IsCount	Способ задания шага резьбы: STEP(0) - значением, COUNT(1) - количеством витков резьбы на единицу длины.
s	CLD[3]	CLD.Value	Значение шага резьбы.
a	CLD[4]	CLD.StartAngle	Начальный угол шпинделя (для многозаходной резьбы).

Пример (из постпроцессора Fanuc (30i)_Mill):

```
program SingleThread
  Feed_ = cld[3]; Feed_@ = MaxReal
  QStartAngle = cld[4]; QStartAngle@ = MaxReal
  Interp_ = 33; Interp_@ = MaxReal
end
```

Доступ из .NET

Обработчик:

```
public override void OnSinglePassThread(ICLDSinglePassThreadCommand cmd,
CLDArray cld)
```

Наследование: `ICLDSinglePassThreadCommand : ICLDCommand`.

Читайте параметры из строго типизированного объекта `cmd` (рекомендуется); также доступны «сырой» массив `cld[i]` (те же индексы, что и в sppx) и именованные помощники `cmd["..."]`. В таблице перечислены члены команды, включая унаследованные от базовых интерфейсов;

общие члены `ICLDCommand` (навигация, `CLD`, `CLDFile`, `TechOperation`, ...) описаны в разделе Модель доступа к CLData.

Член	Тип	Описание
`cmd.Orientation`	`CLDThreadOrient`	Ориентация резьбы (OD, ID, Face)
`cmd.IsODThread`	`bool`	Возвращает "True", если это команда нарезания наружной резьбы.
`cmd.IsIDThread`	`bool`	Возвращает "True", если это команда нарезания внутренней резьбы.
`cmd.IsFaceThread`	`bool`	Возвращает "True", если это команда нарезания торцевой резьбы.
`cmd.Units`	`CLDThreadUnits`	Единицы измерения шага резьбы (мм или TPI).
`cmd.StepIsDistance`	`bool`	Возвращает "True", если шаг резьбы задан в линейных единицах (обычно в мм).
`cmd.StepIsTPU`	`bool`	Возвращает "True", если шаг резьбы задан как число ниток на единицу длины (обычно в нитках на дюйм).
`cmd.Step`	`double`	Значение шага резьбы.
`cmd.StartAngle`	`double`	Начальный угол резьбы в градусах. Часто используется для многозаходной резьбы.

Пример - `OnSinglePassThread` (из постпроцессора Goodway SW):

```
public override void OnSinglePassThread(ICLDSinglePassThreadCommand cmd,
CLDArray cld)
{
```

```
        nc.GInterp.v = 32;
        if (cmd.StepIsTPU)
            nc.F.v = 1.0/cmd.Step;
        else
            nc.F.v = cmd.Step;
        nc.QThreadAngle.v = cmd.StartAngle;
        nc.QThreadAngle.v0 = 0;
}
```

Сопутствующие ссылки

- Модель доступа к CLData

SPINDL - Шпиндель

Отображение команды

Как команда отображается в списке команд CLData:

```
SPINDL ON(71)|OFF(72)|ORIENT(246), NO n, K k, MODE RPM(0)|RANGE(1)|CSS(2) {,
SPEED c}
```

Команда SPINDL передаёт параметры управления шпинделем. Команда SPINDL используется для задания частоты вращения шпинделя или диапазона частот, скорости резания и т. д.

Доступ из sppx

Обработчик: `program Spindl`

Массив `cld` читается по индексу или по имени - в sppx по типизированному короткому имени (например, `cld.X`), в .NET по тому же имени как строковому ключу (например, `cld["X"]`).

Параметр	Массив CLD	Имя CLD	Описание
ON(71) OFF(72) ORIENT(246)	CLD[1]	CLD.OnOff	Тип команды: ON(71) - включение вращения шпинделя, OFF(72) - выключение вращения шпинделя, ORIENT(246) - ориентированный останов шпинделя.
n	CLD[2]	CLD.NO	Частота вращения, угол поворота шпинделя или максимальная частота вращения шпинделя в режиме CSS.
k	CLD[3]	CLD.K	Диапазон частот вращения.

Параметр	Массив CLD	Имя CLD	Описание
RPM(0) RANGE(1) CSS(2)	CLD[4]	CLD.Mode	Режим задания частоты вращения: RPM(0) - обороты в минуту, RANGE(1) - диапазон частот вращения шпинделя, CSS(2) - режим постоянной скорости резания.
c	CLD[5]	CLD.CSS	Значение постоянной скорости резания для режима CSS.

Для создания ISO-постпроцессора с помощью шаблонов используйте значения ISO.M, которые соответствуют параметрам SPINDL.

Параметр CLD	Значение ISO
CLD[1] = 71	ISO.M = 3
CLD[1] = 72	ISO.M = 5

Пример (из постпроцессора Fanuc (30i)_Mill):

```
program Spindl
  OutBlock
  case cld[1] of
    71: begin ! Включение
      if cld[4]=2 then begin ! CSS
        GSpindl = 96
        S = abs(cld[5])
        OutBlock
        if cld[5] > 0 then SpDir = 3
                      else SpDir = 4    ! M3/M4  по/против часовой стрелки
        SpRPM = abs(cld[2])
      end else begin ! RPM
        if cld[2] > 0 then Msp = 3
                      else Msp = 4    ! M3/M4  по/против часовой стрелки
        if ((SpDir=3) or (SpDir=4)) and (SpDir<>Msp) then ! Реверс шпинделя
          Output "M5"
        GSpindl = 97
```

```
        S = abs(cld[2]); S@=SpRPM
        Msp@ = SpDir
        SpDir = Msp
        SpRPM = abs(cld[2])
        OutBlock
      end
    end
    72: begin ! Выключение
      Msp = 5
      S=0; S@=S
      SpRPM = 0
      SpDir = 5    ! M3/M4/M5
      OutBlock  ! Msp = 5;; OutBlock; Msp=1; OutBlock
    end
    246: begin  ! M19 ORIENT
      OutBlock
      M = 19; M@ = MaxReal
      OutBlock
    end
  end
end
```

Доступ по имени параметра (помощники)

Параметры, доступные через оператор `cmd` (одинаково в sppx и .NET):

Имя	Помощник	Описание
`Action`	`cmd.Int["Action"]`	Тип команды (0 - включение, 1 - выключение, 2 - ориентация).
`RotationDirection`	`cmd.Int["RotationDirection"]`	Направление вращения.
`SpeedMode`	`cmd.Int["SpeedMode"]`	0 - RPM, 1 - RANGE, 2 - CSS.
`Value`	`cmd.Flt["Value"]`	Частота вращения / угол ориентации / максимальные обороты в минуту.
`CSSValue`	`cmd.Flt["CSSValue"]`	Значение постоянной скорости резания.

Имя	Помощник	Описание
`RangeValue`	`cmd.Int["RangeValue"]`	Диапазон коробки передач.
`SpindleID`	`cmd.Str["SpindleID"]`	Идентификатор шпинделя.

Доступ из .NET

Обработчик:

```
public override void OnSpindle(ICLDSpindleCommand cmd, CLDArray cld)
```

Наследование: `ICLDSpindleCommand : ICLDCommand`.

Читайте параметры из строго типизированного объекта `cmd` (рекомендуется); также доступны «сырой» массив `cld[i]` (те же индексы, что и в sppx) и именованные помощники `cmd[...]`. В таблице перечислены члены команды, включая унаследованные от базовых интерфейсов; общие члены `ICLDCommand` (навигация, `CLD`, `CLDFile`, `TechOperation`, ...) описаны в разделе Модель доступа к CLData.

Свойство	Тип	Описание
`cmd.SpindleID`	string	Текстовый идентификатор шпинделя, для которого эта команда задаёт параметры.
`cmd.Action`	`CLDSpindleAction`	Действие для шпинделя (`On`, `Off` или `Orient`).

`cmd.IsOn`
`cmd.IsOff`
`cmd.IsOrient`
	bool	True, если это команда включения / выключения / углового ориентирования.

`cmd.SpeedMode`	`CLDSpindleSpeedMode`	Режим вращения шпинделя (`RPM` или `CSS`).
`cmd.IsRPM` `cmd.IsCSS`	bool	True, если режим - RPM (об/мин, G97) / CSS (постоянная скорость резания, G96).
`cmd.RotationDir`	`CLDRotationDir`	Направление вращения шпинделя (по часовой стрелке или против).

Свойство	Тип	Описание
`cmd.IsClockwiseDir`	bool	True, если направление вращения - по часовой стрелке (CW).
`cmd.RPMValue`	double	Обороты в минуту (максимальные, если активен режим CSS).
`cmd.CSSValue`	double	Значение линейной (поверхностной) скорости для шпинделя.
`cmd.GearRange`	int	Диапазон коробки передач шпинделя.

Пример (из постпроцессора Sinumerik 840D mill):

```
public override void OnSpindle(ICLDSpindleCommand cmd, CLDArray cld)
{
    if (cmd.IsOn)
    {
        switch (cmd.SpeedMode)
        {
            case CLDSpindleSpeedMode.RPM:
                nc.S.Show(cld[2]);
                nc.Msp.Show(cmd.IsClockwiseDir ? 3 : 4);
                break;
            case CLDSpindleSpeedMode.CSS:
                throw new Exception("CSS mode not realized");
        }
    }
    else if (cmd.IsOff)
        nc.Msp.Show(5);
}
```

Сопутствующие ссылки

- Модель доступа к CLData

STOP - Останов

Отображение команды

Как команда отображается в списке команд CLData:

```
STOP
```

Используйте команду STOP для формирования кадров прерывания выполнения управляющей программы. Обычно это код M00 управляющей программы. При достижении кадра останова выполнение управляющей программы прерывается.

Доступ из sppx

Обработчик: `program Stop`

Пример (из постпроцессора Fanuc (30i)_Mill):

```
program Stop
  if M <> M@ then OutBlock           ! вывод в кадр УП
  M = 0                                ! M0
  M@ = 1
end
```

Доступ из .NET

Обработчик:

```
public override void OnStop(ICLDStopCommand cmd, CLDArray cld)
```

Наследование: `ICLDStopCommand : ICLDCommand`.

Читайте параметры из строго типизированного объекта `cmd` (рекомендуется); доступны также «сырой» массив `cld[i]` (те же индексы, что и в sppx) и именованные обёртки `cmd["..."]`. В таблице перечислены члены команды, включая унаследованные от базовых интерфейсов; общие члены `ICLDCommand` (навигация, `CLD`, `CLDFile`, `TechOperation`, ...) описаны в разделе Модель доступа к CLData.

Пример - `OnStop` (из постпроцессора Fanuc (30i)_Mill_DN):

```
public override void OnStop(ICLDStopCommand cmd, CLDArray cld)
{
```

```
    nc.Block.Out();
    nc.M.Show(0);
    nc.Block.Out();
}
```

Сопутствующие ссылки

- Модель доступа к CLData

STRUCTURE - Структура траектории

Отображение команды

Как команда отображается в списке команд CLData:

```
STRUCTURE ON(71)|OFF(72), TYPE(Level), Comment(Level: -65)
```

Команда STRUCTURE позволяет представить древовидную структуру команд CLData в виде линейного списка команд.

Доступ из sppx

Обработчик: `program Structure`

В CAM-системе траектория каждой операции представлена в виде дерева, разделённого на группы и подгруппы, внутри которых размещаются обычные команды перемещения инструмента, включения шпинделя или подачи и т. д. Внутри генератора постпроцессоров траектория представлена линейным списком команд, не разделённым на группы. Команда STRUCTURE предназначена для сохранения исходной информации о том, в какой группе изначально находилась команда CLData. Каждая группа преобразуется в две команды: `STRUCTURE ON(71)` и `STRUCTURE OFF(72)`. Первая команда определяет начало группы, а вторая - её конец. Все элементы, размещённые внутри группы, будут находиться между этими двумя командами. Группы также могут быть вложены друг в друга: внутри блока `STRUCTURE ON(71)`... `STRUCTURE OFF(72)` может быть вложено несколько других блоков `STRUCTURE ON(71)`... `STRUCTURE OFF(72)`.

Параметр	Массив CLD	Описание
ON(71) OFF(72)	CLD[1]	Признак того, что данная команда является началом или концом группы: ON(71) - команда определяет начало группы, OFF(72) - команда определяет конец группы.

Параметры, доступные через оператор cmd

TCLDStructure: ComplexType	Команда, определяющая структуру
IsClose: Integer	cmd.Int["IsClose"] - Признак того, что данная команда является началом или концом

TCLDStructure: ComplexType	Команда, определяющая структуру
	группы: 0 - команда определяет начало группы, 1 - команда определяет конец группы.
NodeType: String	cmd.Str["NodeType"] - Строка, определяющая тип группы. Например, Approach - группа перемещений подвода, Level - слой обработки и т. д. Эта строка не переводится на локальные языки.
Comment: String	cmd.Str["Comment"] - Имя группы, отображаемое в дереве траектории CAM-системы. Может быть переведено на локальные языки и содержать произвольные символы.

Пример (из постпроцессора робота KUKA):

```
program Structure
  str1, str2: string
  str1 = Cmd.Str["NodeType"]
!  str2 = Cmd.Str["Comment"]

  if CLD.OnOff=71 and drilling = 10 then begin
    cdis = ""
  end

  if CLD.OnOff=72  and drilling = 10 then begin
    cdis = " C_DIS"
  end

!---------------------------------------------------------------------------

  if CLD.OnOff=71 and drilling=1 and pos("BodySection", str1)>0 then !только для
обработки отверстий
    drilling = 10

  if CLD.OnOff=72 and drilling=10 and pos("BodySection", str1)>0 then !только для
обработки отверстий
    drilling = 0

end
```

Доступ из .NET

Обработчик:

```
public override void OnStructure(ICLDStructureCommand cmd, CLDArray cld)
```

Наследование: `ICLDStructureCommand : ICLDCommand`.

Читайте параметры из строго типизированного объекта `cmd` (рекомендуется); также доступны «сырой» массив `cld[i]` (те же индексы, что и в sppx) и именованные помощники `cmd["..."]`. В таблице перечислены члены команды, включая унаследованные от базовых интерфейсов; общие члены `ICLDCommand` (навигация, `CLD`, `CLDFile`, `TechOperation`, ...) описаны в разделе Модель доступа к CLData.

Член	Тип	Описание
`cmd.IsOpen`	`bool`	Содержит "true", если команда является началом новой группы. Иначе содержит "false".
`cmd.IsClose`	`bool`	Содержит "true", если команда является концом группы. Иначе содержит "false".
`cmd.NodeType`	`string`	Текстовый атрибут, определяющий тип группы. Существует несколько стандартных типов групп, таких как "Approach", "String", "Level" и т. д. Но может содержать и любое произвольное значение.
`cmd.Comment`	`string`	Произвольный текстовый комментарий или имя группы.

Пример:

```
public override void OnStructure(ICLDStructureCommand cmd, CLDArray cld)
{
    if (cmd.IsOpen)
        nc.OutWithN("(>>> " + cmd.NodeType + ": " + cmd.Comment + ")");    //
начало группы
    else if (cmd.IsClose)
        nc.OutWithN("(<<< " + cmd.NodeType + ")");                         //
конец группы
}
```

Сопутствующие ссылки

- Модель доступа к CLData

SYNCAXES - Синхронизация движения осей

Назначение этой команды - включить или отключить синхронизацию одновременного движения двух осей станка (например, осей вращения левого и правого шпинделей на токарном станке с двумя шпинделями).

Команда: SYNCAXES ON(71) | OFF(72), FirstAxisID(Axis1), SecondAxisID(Axis2)

Доступ из sppx

Обработчик: `program SyncAxes`

Параметр	Массив CLD	Описание
IsOn	CLD[1]	Новое состояние синхронизации: 71 - синхронизация осей активна. 72 - синхронизация движения осей отключена.

Параметры, доступные через оператор cmd

Параметр	Тип	Описание
Action	Integer	Переключатель, задающий состояние синхронизации (включена/отключена): 1 - установить состояние синхронизации как включённое. 0 - установить состояние синхронизации как отключённое.
Axis1	String	Строковый идентификатор первой из двух синхронизируемых осей.
Axis2	String	Строковый идентификатор второй из двух синхронизируемых осей.

Простой пример программы-обработчика для этой команды:

```
program SyncAxes
  Outblock
  a1: String;
  a1 = cmd.Str["Axis1"];
  a2: String;
  a2 = cmd.Str["Axis2"];
  if (a1="AxisCPos") or (a2="AxisCPos") or (a1="AxisC1Pos") or (a2="AxisC1Pos")
then begin
    if CLD.IsOn=71 then begin
      GSyncSpeeds = 288; GSyncSpeeds@ = Maxreal;
      Formblock;
      Output Outstr$ + " (Sync spindles rotation)";
      Phase = C2-C; Phase@=MaxReal;
      GSyncPhase = 289; GSyncPhase@=0;
      FormBlock;
      output OutStr$ + " (Sync phase)";
    end else begin
      GsyncSpeeds = 290; GsyncSpeeds@ = Maxreal;
      Formblock;
      Output Outstr$ + " (Cancel spindles sync)";
    end;
  end;
end
```

Доступ из .NET

Обработчик:

```
public override void OnSyncAxes(ICLDSyncAxesCommand cmd, CLDArray cld)
```

Наследование: `ICLDSyncAxesCommand : ICLDCommand`.

Читайте параметры из строго типизированного объекта `cmd` (рекомендуется); также доступны «сырой» массив `cld[i]` (те же индексы, что и в sppx) и именованные помощники `cmd["..."]`. В таблице перечислены члены команды, включая унаследованные от базовых интерфейсов; общие члены `ICLDCommand` (навигация, `CLD`, `CLDFile`, `TechOperation`, ...) описаны в разделе Модель доступа к CLData.

Член	Тип	Описание
`cmd.IsOn`	`bool`	Содержит "true", если команда включает режим

Член	Тип	Описание
		синхронного движения, иначе - "false".
`cmd.IsOff`	`bool`	Содержит "true", если команда отключает режим синхронного движения, иначе - "false".
`cmd.FirstAxisID`	`string`	Текстовый идентификатор первой из осей станка, между которыми выполняется синхронизация.
`cmd.SecondAxisID`	`string`	Текстовый идентификатор второй из осей станка, между которыми выполняется синхронизация.

Пример - `OnSyncAxes` (из постпроцессора Goodway SW):

```
public override void OnSyncAxes(ICLDSyncAxesCommand cmd, CLDArray cld)
{
    if (SameText(CurrentOperation.TypeName, "TSTTakeoverMTM")) return;

    base.OnSyncAxes(cmd, cld);
    if (cmd.FirstAxisID.Contains("AxisC1") || cmd.FirstAxisID.Contains("AxisC2")){
        SyncSpindles(cmd.IsOn,false);
    }else
    if (cmd.FirstAxisID.Contains("AxisZ1") || cmd.FirstAxisID.Contains("AxisZ2")){
        SyncZ(cmd.IsOn, false);
    }else{
        Log.Error("Axes can not be syncronized:
("+cmd.FirstAxisID+","+cmd.SecondAxisID+")");
    }
}
```

Сопутствующие ссылки

- Модель доступа к CLData

TAKEOVER - Перехват заготовки

Отображение команды

Как команда отображается в списке команд CLData:

```
TAKEOVER WorkpieceConnectorID
```

Используйте команду TAKEOVER для перехвата заготовки из активной державки в державку, указанную в параметре команды. Активная державка заготовки задаётся командой SELWORKPIECE. Державка заготовки - это узел станка, используемый для закрепления заготовки. Используйте эту команду для многошпиндельных токарных или токарно-фрезерных станков для передачи заготовки из одного шпинделя в другой.

Доступ из sppx

Обработчик: `program Takeover`

Единственный параметр команды TAKEOVER - строковый идентификатор державки заготовки. Этот идентификатор определяется схемой станка CAM-системы. Значение параметра сохраняется в предопределённой переменной CLDATA$.

Пример:

```
program Takeover
  ! CLDATA$ - идентификатор целевой державки заготовки
  Output "( TAKEOVER TO " + CLDATA$ + " )"
end
```

Доступ из .NET

Обработчик:

```
public override void OnTakeover(ICLDTakeoverCommand cmd, CLDArray cld)
```

Наследование: `ICLDTakeoverCommand : ICLDCommand`.

Читайте параметры из строго типизированного объекта `cmd` (рекомендуется); доступны также «сырой» массив `cld[i]` (те же индексы, что и в sppx) и именованные обёртки `cmd["..."]`. В таблице перечислены члены команды, включая унаследованные от базовых интерфейсов; общие члены `ICLDCommand` (навигация, `CLD`, `CLDFile`, `TechOperation`, ...) описаны в разделе Модель доступа к CLData.

Член	Тип	Описание
`cmd.TargetConnectorID`	`string`	Текстовый идентификатор державки заготовки. Этот идентификатор определяется схемой станка.

Пример:

```
public override void OnTakeover(ICLDTakeoverCommand cmd, CLDArray cld)
{
    nc.OutWithN("( TAKEOVER TO " + cmd.TargetConnectorID + " )");   // передать
заготовку в целевую державку заготовки
}
```

Сопутствующие ссылки

- Модель доступа к CLData

TLCONTACT - Контакт инструмента

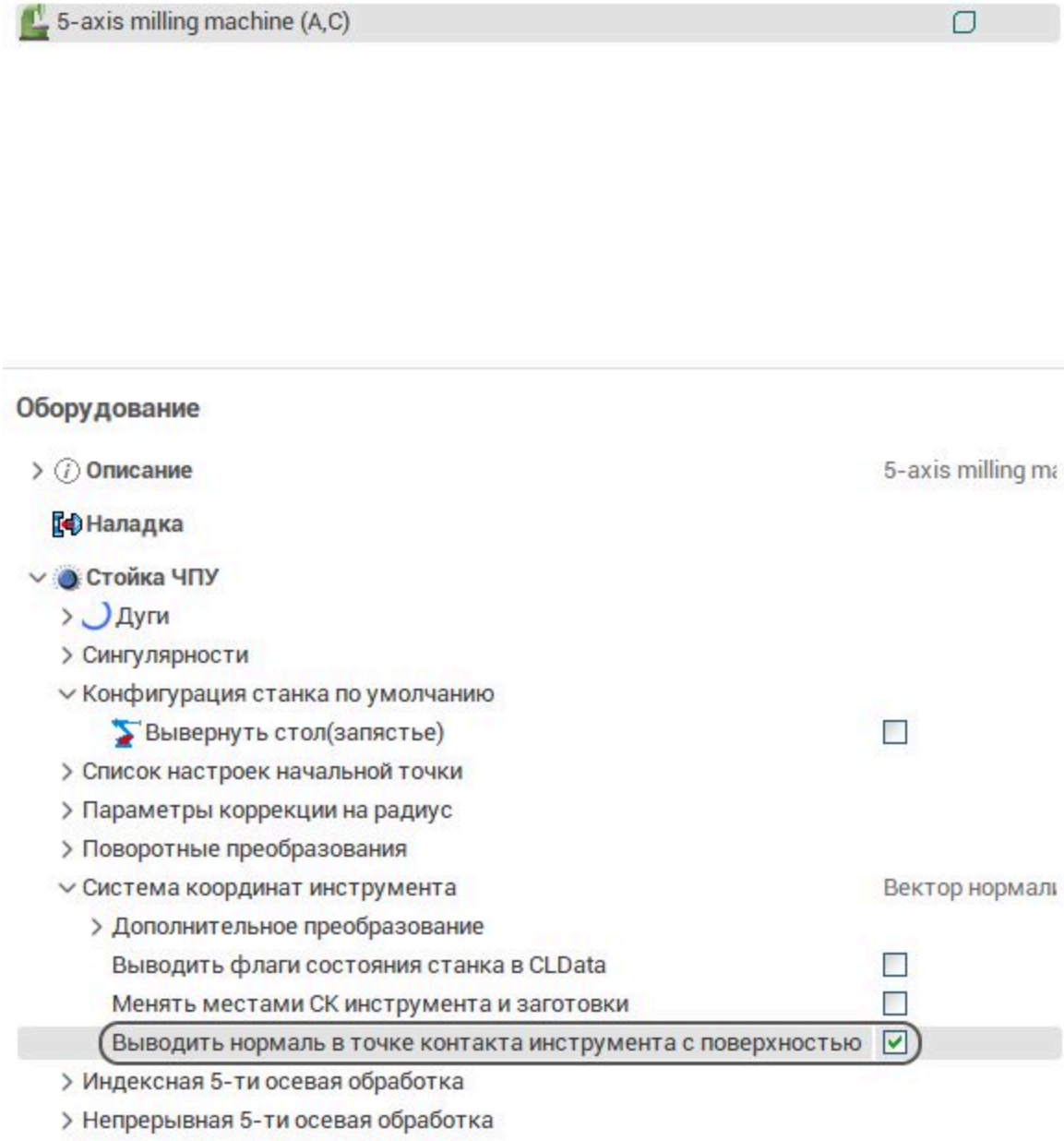
Команда TLContact сообщает нормаль к обрабатываемой поверхности в точке контакта инструмента с этой поверхностью. Эти данные используются для расчёта 5D-коррекции. По сути, они определяют, в каком направлении (по какому вектору) нужно сместить инструмент в пространстве на величину коррекции (разницу между радиусом теоретического и фактического инструмента).

Доступ из sppx

Обработчик: `program TLContact`

Параметр	Массив CLD	Имя CLD	Описание
OnOff	CLD[1]	CLD.OnOff	Определяет, контактирует ли инструмент с обрабатываемой поверхностью в последующих перемещениях. ON(1) - информация о контакте есть, OFF(0) - информации о контакте нет.
NX	CLD[2]	CLD.NX	X-компонента вектора нормали поверхности
NY	CLD[3]	CLD.NY	Y-компонента вектора нормали поверхности
NZ	CLD[4]	CLD.NZ	Z-компонента вектора нормали поверхности

Эта команда формируется в CLData, только если в настройках схемы станка включена соответствующая опция "Задавать вектора нормали поверхности контакта инструмента".



Пример:

```
! deltaR - разница между радиусом инструмента, заданным в CAM, и реальным радиусом
инструмента на станке
! (присваивается один раз, там, где становится известен реальный радиус инструмента)

program TLContact
  ! cld[1] - флаг контакта ON(1)|OFF(0); cld[2..4] - нормаль поверхности контакта
NX, NY, NZ
  if cld[1] = 1 then begin
    NcX = cld[2]  NcY = cld[3]  NcZ = cld[4]            ! запоминаем нормаль
контакта в глобальных переменных
    HasContact = 1
    Output "( CONTACT N = " + Str(NcX) + " " + Str(NcY) + " " + Str(NcZ) + " )"
  end else
```

```
    HasContact = 0
end

program AbsMov
  ! смещаем острие инструмента вдоль запомненной нормали контакта на deltaR
  if HasContact = 1 then begin
    X = cld[1] + NcX * deltaR
    Y = cld[2] + NcY * deltaR
    Z = cld[3] + NcZ * deltaR
  end else begin
    X = cld[1]  Y = cld[2]  Z = cld[3]
  end
  OutBlock
end
```

Доступ из .NET

Обработчик:

```
public override void OnTLContact(ICLDTLContactCommand cmd, CLDArray cld)
```

Наследование: `ICLDTLContactCommand : ICLDCommand`.

Читайте параметры из строго типизированного объекта `cmd` (рекомендуется); также доступны «сырой» массив `cld[i]` (те же индексы, что и в sppx) и именованные помощники `cmd["..."]`. В таблице перечислены члены команды, включая унаследованные от базовых интерфейсов; общие члены `ICLDCommand` (навигация, `CLD`, `CLDFile`, `TechOperation`, ...) описаны в разделе Модель доступа к CLData.

Член	Тип	Описание
`cmd.IsOn`	`bool`	Определяет, контактирует ли инструмент с обрабатываемой поверхностью в последующих перемещениях. "True" - информация о контакте есть, "False" - информации о контакте нет.
`cmd.IsOff`	`bool`	Определяет, контактирует ли инструмент с

Член	Тип	Описание
		обрабатываемой поверхностью в последующих перемещениях. "True" - информации о контакте нет, "False" - информация о контакте есть.
`cmd.ContactNormal`	`TInp3DPoint`	Вектор нормали контакта обрабатываемой поверхности (X, Y, Z).

Пример:

```
// deltaR - разница между радиусом инструмента, заданным в CAM, и реальным радиусом инструмента на станке
TInp3DPoint contactNormal;
bool hasContact;

public override void OnTLContact(ICLDTLContactCommand cmd, CLDArray cld)
{
    if (cmd.IsOn) {
        contactNormal = cmd.ContactNormal;      // запоминаем нормаль контакта поверхности
        hasContact = true;
        nc.OutWithN($"( CONTACT N = {contactNormal.X} {contactNormal.Y} {contactNormal.Z} )");
    } else {
        hasContact = false;
    }
}

public override void OnGoto(ICLDGotoCommand cmd, CLDArray cld)
{
    // смещаем острие инструмента вдоль запомненной нормали контакта на deltaR (коррекция по радиусу)
    if (hasContact) {
        nc.X.Show(cmd.EP.X + contactNormal.X * deltaR);
        nc.Y.Show(cmd.EP.Y + contactNormal.Y * deltaR);
        nc.Z.Show(cmd.EP.Z + contactNormal.Z * deltaR);
    } else {
        nc.X.Show(cmd.EP.X);
        nc.Y.Show(cmd.EP.Y);
```

```
            nc.Z.Show(cmd.EP.Z);
        }
        nc.Block.Out();
    }
```

Сопутствующие ссылки

- Модель доступа к CLData

WAIT - Ожидание точки синхронизации

Отображение команды

Как команда отображается в списке команд CLData:

```
WAIT ID(<SynchPointID>), Index(<SynchPointIndex>)
```

Команда WAIT используется для синхронизации нескольких параллельных потоков управления (нескольких управляющих программ, каждая из которых может управлять отдельным рабочим элементом станка). Для этой команды в управляющую программу следует добавлять кадры (метки), реализующие приостановку программы до достижения аналогичной метки в другой (параллельной) программе.

Доступ из sppx

Обработчик: `program Wait`

Параметры, доступные через оператор cmd

TCLDSyncPoint: ComplexType	Команда ожидания точки синхронизации
SyncPointID: String	cmd.Str["SyncPointID"] - уникальный идентификатор точки синхронизации, заданный пользователем.
Index: Integer	cmd.Int["Index"] - порядковый номер точки синхронизации, автоматически формируемый системой.

Если в проекте применяется операция «Токарного перехвата», которая должна быть синхронизирована с операцией «Токарной отрезки», то в командах меток ожидания параметр `SyncPointID` имеет свои строго определённые значения:

- Takeover1: метка ожидания с таким идентификатором выводится непосредственно после команд PPFUN TECHINFO (58) обеих операций. Оба канала ожидают взаимной синхронизации.
- Takeover2: с достижением метки ожидания с таким идентификатором операция «Токарного перехвата» перемещается к главному шпинделю и захватывает деталь в нём. Операция «Токарной отрезки» при этом находится в состоянии ожидания до достижения метки ожидания с идентификатором Takeover3.

- Takeover3: с достижением метки ожидания с таким идентификатором операция «Токарной отрезки» выполняет отрезку готовой детали из главного шпинделя. Операция «Токарного перехвата» в состоянии захвата отрезаемой детали ждёт окончания отрезки, то есть метки ожидания с идентификатором Takeover4.
- Takeover4: с достижением метки ожидания с таким идентификатором операция «Токарного перехвата» начинает отвод противошпинделя с захваченной отрезанной деталью. Операция «Токарной отрезки» ожидает, пока не будет выполнен отвод противошпинделя на расстояние подачи возврата, что обозначается меткой ожидания с идентификатором Takeover5.
- Takeover5: по достижении этой метки ожидания операция «Токарной отрезки» выполняет отвод инструмента (при необходимости), операция «Токарного перехвата» отводит противошпиндель в исходное положение.

SyncPointID	Операция «Токарной отрезки»	Операция «Токарного перехвата»
Takeover1	Ожидание взаимной синхронизации	Ожидание взаимной синхронизации
Takeover2	Находится в состоянии ожидания до достижения метки Takeover3.	Перемещается к главному шпинделю и захватывает деталь в нём.
Takeover3	Выполняет отрезку готовой детали из главного шпинделя.	В состоянии захвата отрезаемой детали ждёт окончания отрезки (метки Takeover4).
Takeover4	Ожидает, пока не будет выполнен отвод противошпинделя на расстояние подачи возврата (метка Takeover5).	Начинает отвод противошпинделя с захваченной отрезанной деталью.
Takeover5	Выполняет отвод инструмента (при необходимости).	Отводит противошпиндель в исходное положение.

Пример (из постпроцессора PUMA MX series):

```
program Wait
    Call OutBl
```

```
    if NextCode= CodeOfCmd("PPFUN") then begin
      output "M01";
      NeedM1 =0;
    end
    i : integer;
    i = Cmd.Int["Index"];
    call AddWait(i);
end
```

Доступ из .NET

Обработчик:

```
public override void OnSyncWait(ICLDSyncWaitCommand cmd, CLDArray cld)
```

Наследование: `ICLDSyncWaitCommand : ICLDCommand`.

Читайте параметры из строго типизированного объекта `cmd` (рекомендуется); доступны также «сырой» массив `cld[i]` (те же индексы, что и в sppx) и именованные обёртки `cmd["..."]`. В таблице перечислены члены команды, включая унаследованные от базовых интерфейсов; общие члены `ICLDCommand` (навигация, `CLD`, `CLDFile`, `TechOperation`, ...) описаны в разделе Модель доступа к CLData.

Член	Тип	Описание
`cmd.PointIndex`	`int`	Порядковый номер точки синхронизации, автоматически формируемый системой.
`cmd.PointID`	`string`	Уникальный текстовый идентификатор точки синхронизации, заданный пользователем.

Пример - `OnSyncWait` (из постпроцессора HanwhaSwissType):

```
public override void OnSyncWait(ICLDSyncWaitCommand cmd, CLDArray cld)
{
    int PntN = 0;
    if (NCFiles.OutputDisabled)
        return;
```

```
if (cmd.PointID.Contains("SubAxial2"))
    return; //Средняя метка в ожидании операции главного шпинделя.
if (SameText(cmd.PointID, "Takeover1")) return;

if (SameText(cmd.PointID, "Takeover2")) {
    PntN = 2;
    SynchPoints.OutPrev(cmd.PointIndex,1);
    if (nc==nc1)
    {
        SyncSpindles(true);
    }
} else if (SameText(cmd.PointID, "Takeover3")) {
    PntN = 3;
    if (activeChannel>0)
      outClamp(false,true);
    SynchPoints.OutPrev(cmd.PointIndex,1);
    nc.WriteLine(GetSyncZCode(true));
} else if (SameText(cmd.PointID, "Takeover4")) {
    PntN = 4;
    SynchPoints.OutPrev(cmd.PointIndex,2);
    nc.WriteLine(GetSyncZCode(false));
    if (nc==nc1) {
        SyncSpindles(false);
    } else {
        IncrementalMove=true;
    }
    SynchPoints.OutPrev(cmd.PointIndex,1);
    if (nc==nc1) {
        nc.WriteLine("S0");
    } else {
        nc.WriteLine("S40");
    }
} else if (SameText(cmd.PointID, "Takeover5")) {
    PntN = 5;
    IncrementalMove=false;
}

SynchPoints.Add(cmd.PointIndex);

if (PntN==4)
    OutDetectBrokenTool();

if ((nc==nc2) && (PntN==2)) {
    outAirBlower(0);
    outClamp(false, false);
}
```

```
    if ((nc==nc2) && (PntN==5)) {
        nc.WriteLine("M95");
        nc.WriteLine("/M91");
        nc.WriteLine("M96");
    }

}
```

Сопутствующие ссылки

- Модель доступа к CLData

Разработка постпроцессоров с помощью ИИ

Этот раздел описывает, как разрабатывать и сопровождать постпроцессоры с помощью искусственного интеллекта (ИИ): как запустить инструменты, о чём просить ассистента и как проверять результат. Раздел относится к обеим подсистемам постпроцессирования — SPPX-постпроцессорам и .NET-постпроцессорам, — потому что обе читают одни и те же входные данные, CLData.

Инструменты находятся в состоянии бета-версии: названия команд и вид панелей ещё могут измениться.

Как выглядит работа

Вы пишете в чате своими словами:

> На этом станке не включается охлаждение после смены инструмента. Разберись, почему, и исправь.

И дальше наблюдаете. Ассистент открывает тестовый проект, находит команду, из которой должен был получиться этот кадр, находит обрабатывающий её обработчик, читает код, вносит небольшое изменение, компилирует постпроцессор, запускает его на том же проекте и показывает полученную управляющую программу (УП) и то, чем она отличается от предыдущей.

Вы читаете результат и отвечаете так же — «верно, но M08 должен идти до запуска шпинделя», — и цикл повторяется.

Вот и весь порядок работы. Ничего не нужно запоминать, никаких команд набирать не требуется, и код за вас пишет ассистент — хотя вы в любой момент можете посмотреть каждое изменение и поправить его руками. Чего передать нельзя, так это решения: корректна ли программа для вашего станка и безопасна ли она на нём, определяете вы. Всё, что в этом разделе сказано о проверке результата, существует именно для этого.

Во что это обходится

Инструменты и знания входят в продукт: серверы, расширения, эта документация и навыки. Модель, которая вам отвечает, — нет: ИИ-ассистент представляет собой услугу того, кто предоставляет модель, и оплачивается по объёму обработанного текста.

В CAM Agent можно начать сразу: с ним идёт ознакомительный лимит, которого достаточно, чтобы почувствовать, на что это способно. Его объём и условия со временем могут меняться. Когда лимит израсходован, подключите собственную учётную запись любого общедоступного

поставщика ИИ-услуг; дальше работа идёт на его условиях и по его расценкам, а CAM-система по-прежнему предоставляет инструменты и знания.

При работе через чат-расширение в Visual Studio Code ваша учётная запись у поставщика нужна с первого же запроса. Ознакомительного лимита в этом случае нет: модель тарифицирует тот поставщик, через которого вы подключились, по условиям, о которых вы договорились с ним.

Если работа должна оставаться внутри вашей сети, модель может жить там же. Современные средства позволяют развернуть её на собственном оборудовании, и любой клиент, которого можно направить на локальную модель, будет с ней работать. Всё остальное в этом разделе не меняется: серверы, расширения и сам постпроцессор и так локальны, поэтому данным проекта покидать сеть не требуется. Отказаться придётся от онлайновой половины сервера знаний: поиск по публичной документации идёт через онлайн-сервис и в изолированной сети недоступен, хотя поиск по навыкам продолжает работать после обновления их кэша. Документация в вашей рабочей папке работает в любом случае, как и ваши собственные заметки, — ещё одна причина держать локальную копию даже тогда, когда всё остальное подключено.

Два способа работы

	CAM Agent	Visual Studio Code
Где вы работаете	Чат агента внутри CAM-системы	VS Code с AI chat extension
Что вы видите	Задачу, план и результат; при желании — среду разработки постпроцессоров	Код, различия, CLData и сформированную УП
Для чего удобнее	Получить результат, не открывая код	Разрабатывать, отлаживать и вести постпроцессор в системе контроля версий
SPPX	Полный цикл: прочитать, изменить, скомпилировать, запустить	Полный цикл, с кодом перед глазами
.NET	Работа по задачам: агент работает с проектом и генератором	Естественный выбор: языковой сервис C#, сборка, отладка

Способы не исключают друг друга. Попросить агента в CAM-системе исследовать данные, а затем довести код в VS Code — нормальный рабочий сценарий.

Порядок чтения

1. Как устроена разработка с помощью ИИ — из каких частей всё состоит и что делает каждая.
2. Настройка инструментов — скачать набор и передать настройку ассистенту.
3. Организация рабочего пространства — как устроить папку, в которой у ассистента есть память.
4. Разработка через CAM Agent или Разработка в Visual Studio Code — выберите точку входа.
5. Что дают расширения — панели и команды, которыми вы будете пользоваться, и где они находятся.
6. Рабочий цикл SPPX или Рабочий цикл .NET — цикл разработки для вашего типа постпроцессора.
7. Проверка и подтверждение результата — что проверить, прежде чем применять УП.
8. Устранение проблем и Расширенная настройка — если что-то не работает и полная процедура настройки.

Skills для ассистента

Skill (по-русски — навык) — это короткая инструкция обычными словами: что сделать, в каком порядке, что проверить и когда работа считается законченной. Никакого кода в ней нет — это текстовый файл, который ассистент читает и которому следует, примерно как толковое объяснение новому сотруднику. Написать или поправить такой навык может любой, кто способен объяснить порядок работы словами.

Готовые навыки охватывают подготовку рабочего пространства, исследование входных данных, рабочие циклы SPPX и .NET и проверку сформированной УП.

Они публикуются с этой документацией, в папке `skills` этого раздела — каждый отдельной папкой с файлом `SKILL.md` внутри. Набор пополняется вместе с инструментами: берите его целиком, а ассистент выберет подходящий задаче. Их загрузка входит в настройку — см. расширенную настройку.

Рядом с ними можно держать свои: это обычный Markdown, и практики, которые накапливает ваша команда, оформляются так же — см. организацию рабочего пространства.

Что остаётся за вами

Считайте любой ответ черновиком. Ассистент видит только то, что вы ему дали: он не знает недокументированного поведения станка, а правдоподобно выглядящая УП может быть неверной. Просмотрите изменения, скомпилируйте, запустите постпроцессор, прочитайте

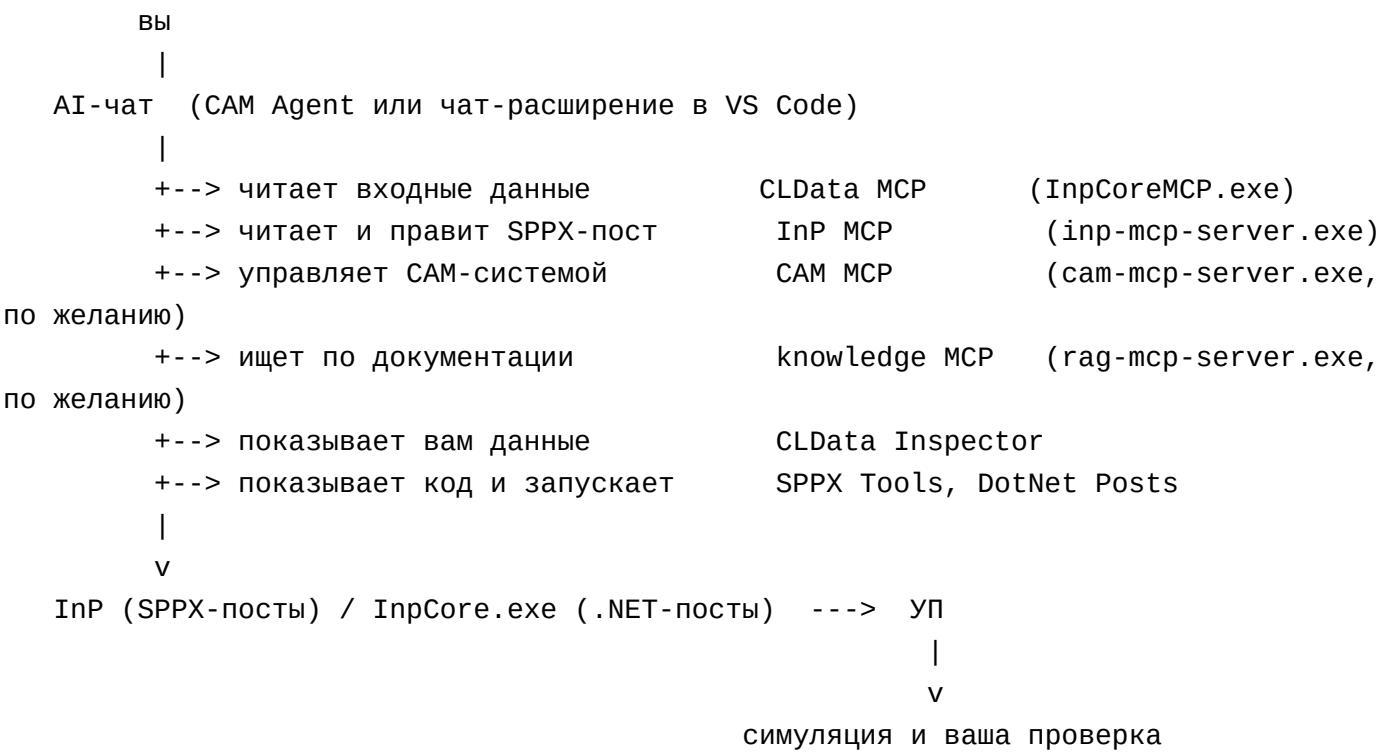
сформированную УП и проверьте её на симуляции, прежде чем она попадёт на станок. Не передавайте в чат пароли, архивы заказчиков и данные о настройках станка.

Как устроена разработка с помощью ИИ

Вы разговариваете с чатом. За ним работают три вида частей: серверы, которые дают ассистенту читать данные и управлять постпроцессором, расширения, которые показывают вам найденное и полученное, и среда исполнения постпроцессора, превращающая CLData в УП.

К серверам обращаются по MCP — Model Context Protocol, общепринятому способу дать чат-клиенту доступ к инструментам. Поэтому одни и те же серверы работают и в CAM Agent, и в любом чат-расширении, к которому вы их подключите.

Слово «сервер» здесь не должно смущать. Это обычные программы на вашем компьютере, которые чат-клиент запускает сам, когда они ему нужны, чтобы инструменты могли с ним обмениваться данными. Они не публикуют файлы вашего проекта и не принимают входящие сетевые подключения. Исключение — сервер знаний: он обращается к службе документации и хранит локальный кэш.



Из чего всё состоит

Часть	Что это	Что она вам даёт
CAM Agent	Расширение CAM-системы	Чат внутри CAM-системы с доступом к проекту и постпроцессору

Часть	Что это	Что она вам даёт
Skills (навыки)	Текстовые файлы	Написанные словами инструкции, которым следует ассистент, поэтому цикл каждый раз одинаков
CLData MCP	Программа на вашем компьютере, которую запускает чат-клиент	Позволяет ассистенту читать входные данные: файлы, секции, команды, параметры, станок. Ничего не изменяет
InP MCP	То же	Позволяет ассистенту работать с SPPX-постпроцессором: структура, код, регистры, компиляция, запуск
CAM MCP	То же, по желанию	Позволяет ассистенту заглянуть в исходный проект — станок, операции, инструменты — и подготовить входные данные: пересчитать траектории, выгрузить CLData, создать тестовый проект
Knowledge MCP	То же, по желанию; единственный, кто обращается в сеть	Ищет по снимку публичной документации и skills — по тому, что умеет продукт. То, что решила ваша команда, живёт в ваших собственных заметках
CLData Inspector	Расширение Visual Studio Code	Показывает вам команды и параметры, о которых говорит ассистент
SPPX Tools	Расширение Visual Studio Code	Поддержка редактирования `*.sppx` и панель Generate NC

Часть	Что это	Что она вам даёт
DotNet Posts	Расширение Visual Studio Code	То же для постпроцессоров на C#: навигация, сборка, запуск
InP	Приложение CAM-системы	Исполняет SPPX-постпроцессоры; он же — среда, за которой можно наблюдать
`InpCore.exe`	Программа CAM-системы, работает без окна	Исполняет .NET-постпроцессоры

Не путайте три названия: `InP.exe` — среда разработки и исполнитель SPPX-постпроцессоров; `InpCore.exe` — пакетный исполнитель .NET-постпроцессоров; InP MCP — сервер, через который ассистент управляет экземпляром InP. Если экземпляр недоступен, InP MCP сообщает об этом, а не запускает молча другой постпроцессор.

Оконный и безоконный режим

SPPX-постпроцессор исполняет InP, и ассистент может запустить его двумя способами.

Оконный открывает среду, за которой можно наблюдать: изменённый код, сообщения компиляции, сформированную УП. Используйте его, когда хотите следить за ходом работы и когда ассистент застрял, — вы можете продолжить сами в том же экземпляре.

Безоконный работает без окна. Он быстрее и допускает несколько параллельных прогонов, но на экране ничего нет, поэтому в вашем распоряжении только отчёт и сформированный файл.

Если ассистент сообщает, что не может запустить экземпляр, откройте InP из CAM-системы сами и дайте ему подключиться.

CLData одинаков для обеих подсистем

Обе подсистемы постпроцессирования читают одни и те же команды и параметры; различается только синтаксис обращения к ним. Поэтому ассистенту следует всегда читать фактический проект, а не рассуждать по названию команды: документация объясняет смысл команды, а проект показывает, что в ней действительно есть.

Что ассистент делает хорошо, а что не может

Он хорошо находит команду, породившую конкретный кадр УП, объясняет незнакомый обработчик, готовит небольшое изменение в стиле окружающего кода, перечисляет случаи, которые обработчик не учитывает, и сравнивает две УП кадр за кадром.

Он не может знать поведение станка, которого нет ни в данных, ни в документации, гарантировать компилируемость сформированного кода, судить о безопасности перемещений и отвечать за постпроцессор, выпущенный в производство. Успешная компиляция и успешный запуск доказывают, что сработали программы, а не что программа верна.

Настройка инструментов

Настройкой занимается сам ассистент. Ваша часть невелика: установить чат, с которым будете разговаривать, подготовить архив инструментов и ссылку на инструкцию и передать их ему. Всё дальнейшее — установку расширений, подключение MCP-серверов, указание им папки установки CAM, загрузку навыков — ассистент делает сам и докладывает о результате.

1. Установите ассистента

В CAM-системе. CAM Agent устанавливается из её собственного Extension Store: откройте магазин, установите CAM Agent, перезапустите CAM-систему, если она об этом попросит. Больше для начала работы ничего не нужно — см. разработку через CAM Agent.

В Visual Studio Code. Чат — такое же расширение, как и остальные: откройте панель Extensions (`Ctrl+Shift+X`), найдите нужное по названию в магазине расширений и установите. Какое выбрать и чем клиент должен обладать для этой работы — в разработке в Visual Studio Code; клиент, живущий в терминале, устанавливается своим способом. Ещё понадобится учётная запись у поставщика ИИ-услуг — см. во что это обходится.

Если вы работаете только через CAM Agent, остальная часть этой страницы всё равно относится к знаниям и навыкам, которыми он пользуется; набор инструментов ниже нужен для работы в VS Code.

2. Подготовьте инструменты и инструкцию

Адрес загрузки инструментов указан на странице «Внешние ссылки и примеры».

- Набор инструментов — архив, имя которого начинается на `posts-vscode-extensions-`. Скачайте и распакуйте его, но пока ничего не устанавливайте.
- Инструкция настройки — откройте расширенную настройку и передайте ассистенту ссылку на эту статью.

Если ваш клиент уже подключён к сервису поиска по документации, достаточно названия статьи: ассистент найдёт её через этот сервис.

3. Передайте это ассистенту

Откройте чат, разрешите ему выполнять команды в рабочей папке и поставьте задачу. Поправьте пути и вставьте:

```
Настрой инструменты для разработки постпроцессоров по инструкции
https://docs.sprut.ru/technical/Postprocessors/20/ru/src/ai-assisted-
development/advanced-setup.html
```

```
Распакованный архив с расширениями: <bundle>
CAM-система установлена в: <cam>

Установи расширения, подключи серверы CLData и InP к этому клиенту,
укажи расширениям папку установки CAM и загрузи опубликованные навыки.
Затем скажи, что именно изменил и какие инструменты теперь видишь.
```

Два действия ассистент за вас сделать не сможет: перезагрузить окно редактора и перезапустить чат-клиент после изменения конфигурации MCP. О том, что это нужно, он сообщит.

4. Проверьте, что всё получилось

Две просьбы в чат — этого достаточно. Обе безопасны: ассистент только читает, ничего не изменяет и не запускает на станке.

Проверка чтения данных. Возьмите любой проект CAM-системы — файл `*.stcp`, лучше небольшой тестовый — и напишите своими словами примерно так:

> Прочти CLData из проекта `D:\Test\Пример.stcp` и скажи, какой там станок, в каких единицах и сколько файлов с командами.

Хороший ответ называет станок, единицы измерения и перечисляет файлы с количеством команд в каждом — то есть сведения именно из вашего файла. Сверьте их с проектом: если станок не тот, вы дали ассистенту не тот путь.

Проверка чтения постпроцессора. Возьмите файл `*.sppx` и напишите:

> Прочитай постпроцессор `D:\Posts\Пример.sppx` и перечисли обработчики, подпрограммы и регистры.

Хороший ответ — список имён, взятых из этого файла. Для SPPX ассистенту нужна работающая среда InP: он либо запустит её сам, либо попросит вас открыть постпроцессор в ней.

Как отличить неудачу. Ассистент без инструментов не молчит, а рассуждает: перечисляет «типичные» обработчики вроде `OnRapid` и `OnLine`, объясняет, как обычно устроен CLData, или просит прислать содержимое файла. Всё это значит, что нужный инструмент не подключился, — вернитесь к шагу 3 или к расширенной настройке.

5. Обновления

Когда доступна новая версия, в панелях постпроцессора появляется действие обновления — это единственное, что вам нужно заметить. Согласитесь на него и перезагрузите окно.

Если что-то не работает

В устранении проблем собраны типичные симптомы. В расширенной настройке вся процедура подробно — та самая страница, которую вы передали ассистенту, так что любой шаг можно выполнить руками и проверить, что он сделал.

Организация рабочего пространства

Ассистент начинает каждую сессию без памяти о предыдущей. Он не помнит, о чём вы договорились вчера, какой станок доставил хлопот в прошлом месяце и почему конкретный кадр форматируется именно так. Он знает то, что записано в вашей рабочей папке; то, что осталось в переписке, для него не существует.

Поэтому рабочая папка — это не порядок ради порядка, а память ассистента. Несколько часов, потраченных на неё однажды, экономят одни и те же объяснения в каждой последующей сессии.

Что в ней держать

Одна папка, открытая в VS Code как рабочая:

```
work\
├── AGENTS.md      что ассистент должен знать каждый раз
├── posts\         постпроцессоры, с которыми вы работаете
├── projects\      тестовые проекты CAM и эталонные УП
├── docs\          копия этой документации
└── notes\         ваши практики и разобранные случаи
    └── skills\    ваши собственные навыки — инструкции для ассистента
```

Имена не важны, важно разделение: исходники отдельно от тестовых данных, ваши знания отдельно от официальной документации.

AGENTS.md — короткая постоянная памятка

Этот файл читается в начале каждой сессии, поэтому держите его коротким — не больше страницы, — а подробности выносите в `notes\`. Что заслуживает места в нём:

- станки и системы ЧПУ, с которыми вы работаете, и какой постпроцессор какой из них обслуживает;
- соглашения, действующие всегда: нумерация кадров, формат чисел, комментарии, имена файлов;
- где что лежит: тестовые проекты, эталонные УП, ваши заметки;
- правила работы — читать фактический CLData до правки кода, запускать постпроцессор и сравнивать с эталоном до любого доклада о готовности, не менять производственный постпроцессор без разрешения;
- ограничения, которые вы уже однажды сформулировали, — записанные так, как вы их сказали. Ассистент читает их как инструкцию; иначе через месяц он снова предложит идею, которую вы обоснованно отвергли.

Попросите ассистента составить черновик и поправьте его — это быстрее, чем начинать с пустого файла.

notes — где накапливается ваш опыт

После каждого разобранного случая — короткий файл: что было не так, как нашли, что помогло и почему именно так. Именно «почему» экономит время потом: без него ассистент с чистой совестью предложит вариант, который вы отвергли не просто так.

Полезными эти записи делают две привычки:

- одна тема — один файл, а не один растущий: вам нужно находить, а маленький файл дёшево прочитать;
- описывайте текущее состояние. Исторические оговорки вроде «раньше было иначе» портятся со временем; если папка под контролем версий, история и так есть.

Когда порядок работы начинает повторяться — одна и та же проверка, одна и та же последовательность для семейства станков, — запишите его как свой навык в `notes\skills\` и укажите ассистенту. Публикуемые skills — обычные Markdown-файлы; ваши выглядят так же.

А если такой skill оказался полезен не только у вас, им можно поделиться. Отправка идёт через CAM Agent: откройте его чат и попросите отправить этот skill в общую базу знаний. Он попадёт не сразу в публикуемый набор, а в очередь на рассмотрение, и после того как ревьюер его примет, он станет доступен всем, кто пользуется этими инструментами. Отправляйте только то, что готовы опубликовать: skill, в котором названы заказчики, номера деталей или особенности именно ваших станков, место которому — `notes\`, и больше нигде.

Задача завершена не тогда, когда постпроцессор заработал, а когда то, что вы выяснили, лежит в `notes\`.

Документация рядом с исходниками

Здесь рядом живут знания двух видов, и их стоит различать.

Публичная документация — это снимок: ассистент ищет по нему через сервер знаний, а копия в рабочей папке служит тому же без сети и на известной вам ревизии — см. расширенную настройку. Она отвечает на вопрос, что умеет продукт, и меняется только с выходом новой версии.

Актуальные руководства доступны через центр документации СПРУТКАМ. Для работы ассистента используйте сервер знаний или передайте ему ссылку на нужную статью. Для работы без сети сохраните нужные материалы локально и отметьте их версию и дату.

Ваша собственная документация — та самая `notes\` выше — живая половина, и единственная, которая знает ваши станки, ваши проекты и решения, принятые вашей командой. Она пополняется каждый раз, когда вы что-то разобрали. Публичный снимок её содержать не может, и ассистент тоже: эту часть не напишет никто, кроме вас.

Если описанная в документации деталь противоречит установленному продукту, прав продукт — а заметка об этом идёт в `notes\`: ровно для такого живая половина и нужна.

Два способа хранить историю

Способ 1. Папки и копии — историю ведёте вы сами

Вы храните файлы в обычных папках и отвечаете за их прошлое вручную: перед существенным изменением копируете постпроцессор, рядом с тестовыми проектами держите эталонные УП, а историей служат папки с датами. Это законный способ работы, так работают многие, и ничто здесь не заставляет вас от него отказываться.

Плата за него — история держится на вашей памяти. Забыли сделать копию — предыдущего состояния больше нет, а вопросы «что именно изменилось с пятницы» и «когда это сломалось» остаются без точного ответа.

Способ 2. Git — историю ведёт инструмент

Та же папка, но за прошлое отвечает инструмент, сделанный именно для этого и используемый разработчиками по всему миру.

Он же снимает главное опасение при работе с ассистентом: ничего необратимого тот сделать не может. Каждое состояние сохранено, поэтому неудачная правка — да и целая сессия, ушедшая не туда, — стоит вам минуты на откат, а не работающего постпроцессора. С этим можно позволить ему пробовать то, что иначе пробовать было бы жалко.

Что это даёт в этой работе:

- точный ответ, *что изменилось* между двумя состояниями, вплоть до кадра;
- точный ответ, *когда* поведение сломалось и *что ещё* поменялось в том же шаге;
- *причину*, по которой строка существует, — сообщение того изменения, которое её добавило;
- безопасный способ попробовать рискованное и отказаться без следов;
- возможность передать коллеге постпроцессор вместе с его историей.

Учить git для этого не нужно. Просите словами, а делает ассистент:

- «зафиксируй, исправление смещения работает» — он фиксирует, с сообщением о том, зачем;

- «сравни с тем, что было в пятницу» — он показывает, что изменилось;
- «найди, когда пропал кадр включения охлаждения» — он ищет по истории;
- «почему здесь эта строка?» — он находит изменение, которое её добавило, и читает его сообщение;
- «верни как было до последней правки» — он восстанавливает из истории, а не набирает заново.

Если git не установлен, попросите ассистента его установить — официальная загрузка на https://git-scm.com/downloads. Попросите заодно подготовить рабочую папку: одна папка под контролем версий, со сформированными УП и временными файлами, исключёнными из неё.

Когда открытой папкой должна быть папка постпроцессора

Для .NET-постпроцессора VS Code должен быть открыт на папке самого постпроцессора — именно так работают языковой сервис C#, сборка и отладчик. Постоянная памятка и ваши заметки при этом оказываются вне открытой папки. Достаточно двух приёмов:

- Держите всё в одном репозитории. Пусть репозиторием будет папка `work\` целиком, а папки постпроцессоров лежат внутри неё. Тогда git работает из любой вложенной папки — он находит репозиторий, которому она принадлежит, — и история остаётся одна, какую бы папку вы ни открыли.

- Укажите памятку один раз в настройках клиента. Чат-расширения читают постоянную памятку в той папке, которую им открыли, и выше не поднимаются, — но её можно назвать явно. В Kilo Code для этого есть глобальная настройка `instructions` в файле `kilo.jsonc` (обычно `~/.config/kilo/kilo.jsonc`): в ней перечисляются пути к дополнительным файлам инструкций, и они действуют во всех проектах. Пропишите там полный путь к вашему `work\AGENTS.md` — и он будет прочитан, какую бы папку вы ни открыли.

  Если ваш клиент так не умеет, положите в папку постпроцессора короткий `AGENTS.md` из двух строк: правила — в `..\..\AGENTS.md`, заметки — в `..\..\notes\`.

С навыками проще: установленные на пользователя, а не на проект, они доступны при любой открытой папке, поэтому их достаточно положить один раз.

Для SPPX ничего этого не требуется: `work\` открывается как есть.

Чего в ней быть не должно

Паролей, токенов, файлов лицензий, архивов заказчиков, не относящихся к постпроцессору, и посторонних репозиториев. Всё, что лежит в рабочей папке, ассистент вправе прочитать, а

если вы ведёте историю — это остаётся в ней.

Разработка через CAM Agent

CAM Agent — это чат внутри CAM-системы. Это самый короткий путь от задачи до проверенного результата: вы описываете, какой должна быть УП, а агент исследует проект, изменяет постпроцессор, компилирует его, запускает и докладывает о результате. Открывать код не обязательно — хотя для SPPX вы при желании можете наблюдать каждый шаг в среде разработки постпроцессоров.

Установка

CAM Agent — обычное расширение CAM-системы: откройте Extension Store в CAM-системе, установите CAM Agent и перезапустите CAM-систему, если магазин об этом попросит. Про сам чат написано в его собственной документации, доступной из карточки расширения; здесь описана только работа с постпроцессорами.

С агентом идёт ознакомительный лимит, поэтому для первых сессий от вас больше ничего не требуется; его объём и условия со временем могут меняться. Дальше подключите в настройках агента собственную учётную запись поставщика ИИ-услуг, и работа продолжится на его условиях — см. во что это обходится.

Рабочий цикл

1. Откройте проект с коротким характерным участком траектории. Небольшой тестовый проект ускоряет каждый следующий шаг и делает сравнение читаемым.
2. Откройте постпроцессор, который нужно изменить, или укажите агенту нужный файл.
3. Сформулируйте задачу в терминах УП: какой кадр неверен, каким он должен быть, для какого станка. «Добавить M08 после смены инструмента» — рабочая задача; «сделать охлаждение правильно» — нет.
4. Сначала попросите план и данные, на которых он основан. Агент должен назвать найденную команду CLData и обрабатывающий её обработчик, прежде чем предлагать правку.
5. Разрешите изменение. Агент правит обработчик, компилирует постпроцессор, запускает его на том же проекте и возвращает сформированную УП.
6. Прочитайте отчёт: изменённый код, сообщения компиляции и кадры УП, отличающиеся от предыдущего прогона.
7. Проверьте результат так, как описано в Проверке и подтверждении результата. Затем примите его или уточните задачу.

Шаги 4 и 5 — это то, где время либо экономится, либо теряется. Агент, прочитавший фактические данные, делает небольшую объяснимую правку; агент, которому не дали проект, догадывается по названию команды.

Что вы видите в процессе

Для SPPX агент управляет средой разработки постпроцессоров. Запущенная в оконном режиме, она находится перед вами: изменённый обработчик, сообщения компиляции, сформированная УП. Этот режим стоит выбирать, когда вы хотите следить за ходом работы, и он же выручает, когда агент застрял, — вы можете продолжить сами в том же экземпляре.

В безоконном режиме InP работает без окна. Это быстрее и позволяет вести несколько прогонов параллельно, но на экране ничего нет, поэтому в вашем распоряжении только отчёт и сформированный файл.

Для .NET-постпроцессоров агент работает с проектом и пакетным генератором, поэтому наблюдать за средой нечего: вы видите результат сборки, отчёт о запуске и файл УП.

Знания

CAM Agent отвечает по проиндексированной копии документации по продукту и официальных skills, с офлайн-кэшем, и хранит ваши собственные skills рядом с ними. Репозиторий документации в ваш проект он не клонирует; если нужно, чтобы агент читал конкретную локальную копию документации, скажите это прямо и укажите путь.

Здесь же ваш собственный skill отправляется в общий доступ: попросите CAM Agent его отправить, и он попадёт в очередь на рассмотрение для публикуемого набора — см. организацию рабочего пространства.

Ограничения

Агент не заменяет инженера по постпроцессированию. Два правила стоит назвать прямо: не позволяйте ему менять производственный постпроцессор без вашей проверки и не позволяйте отправлять УП на станок. Запуск, завершившийся с кодом 0, означает, что программы сработали, а не что УП безопасна.

Разработка в Visual Studio Code

В VS Code код остаётся под вашим контролем: вы видите каждое изменение, при желании сами запускаете генерацию и ведёте постпроцессор в системе контроля версий как обычно. Ассистентом служит чат-расширение на ваш выбор, а результаты показывают три расширения из набора.

Сначала всё установите — см. настройку инструментов.

Какое чат-расширение выбрать

Какой бы вы ни выбрали, он работает на вашей учётной записи у поставщика ИИ-услуг и тарифицируется этим поставщиком с первого запроса — см. во что это обходится.

Подойдёт любой ассистент, который отвечает трём условиям, и выбор за вами:

- он подключается к MCP-серверам — именно так он читает CLData и управляет постпроцессором;
- ему можно разрешить выполнять команды в рабочей папке — так он собирает, устанавливает и открывает нужное;
- у него есть место для постоянных инструкций, чтобы правила вашего проекта действовали в каждом запросе.

Рекомендуемая отправная точка — Kilo Code. Для этой работы важны три его свойства:

- Skills — штатный механизм. Skill — это папка с файлом `SKILL.md`, и Kilo Code подхватывает её из `.kilo/skills/` в проекте или из `~/.kilo/skills/` для всех проектов — то есть ровно в том виде, в котором skills публикуются с этой документацией, так что они кладутся без переделки. Новые skills обнаруживаются при старте сессии или по команде `/reload` в открытой.
- Он не привязан к одному провайдеру моделей. Запросы идут через шлюз, который через одну точку доступа даёт модели всех основных провайдеров, и можно использовать свои ключи, так что выбор провайдера остаётся за вами.
- У него открытый исходный код (лицензия MIT), а его сессии можно наблюдать и направлять удалённо, в том числе с телефона.

Другие, работающие с этим набором инструментов:

Чат-расширение	Постоянные инструкции	Примечания
Kilo Code	`AGENTS.md` в корне проекта; инструкции проекта в `kilo.jsonc`	Публикуемые skills загружает как есть

Чат-расширение	Постоянные инструкции	Примечания
Cline, Roo Code	Файл правил клиента	Общее происхождение, поэтому панели выглядят знакомо
Claude Code	`CLAUDE.md`; публикуемые skills загружает как есть	Команды выполняет после вашего подтверждения
GitHub Copilot Chat	`.github/copilot-instructions.md`	MCP и выполнение команд требуют режима агента
Codex CLI	`AGENTS.md`	Работает из терминала, рядом с VS Code

Какой бы вы ни выбрали, предпочитайте клиента, который загружает skills напрямую, — тогда вы получаете и порядок работы, а не только инструменты. С клиентом без механизма skills укажите нужный `SKILL.md` в постоянных инструкциях или вставьте его в чат: это обычный Markdown, он работает и так.

Где каждый клиент хранит конфигурацию MCP, перечислено в расширенной настройке.

Рабочий цикл

1. Работайте с копией или ветвью постпроцессора, никогда с производственным файлом.
2. Откройте только те папки, которые нужны: постпроцессор и, если используете, копию документации.
3. Откройте небольшой тестовый проект. Попросите ассистента исследовать его — CLData Inspector покажет вам те же команды, которые он прочитал.
4. Поставьте одну ограниченную задачу с ожидаемым результатом в УП. Попросите план и данные, на которых он основан, прежде чем будет записан хоть один файл.
5. Просматривайте изменения сразу, как они появляются. Отклоняйте переформатирование и переписывание, которых вы не просили.
6. Дайте скомпилировать и запустить или запустите Generate NC сами. Сравните сформированную УП с предыдущим прогоном и с эталонной программой.
7. Проверьте результат — см. проверку и подтверждение результата.

Подробности по каждому типу постпроцессора — в рабочем цикле SPPX и рабочем цикле .NET.

Просите показывать

Ассистент умеет открыть именно тот объект, на который ссылается, поэтому вы видите текущий исходный код или входные данные, а не фрагмент, процитированный в чате:

- команду в CLData Inspector — нужный файл, команду и параметр;
- обработчик на нужной строке, с подсвеченным участком и короткой подписью;
- панель Generate NC с подготовленной конфигурацией, и с запуском, если вы попросили.

Так и просите: «покажи эту команду в инспекторе», «открой обработчик, где ты поменял смещение».

Учтите, что собственный прогон ассистента молчаливый: он запускает постпроцессор через сервер и сам читает результат — так он проверяет свою работу. Панель Generate NC предназначена вам: конфигурация, входные параметры поста, ход выполнения, ошибки в Problems со ссылками в код обработчиков и сравнение с предыдущим прогоном и с эталонной программой.

Пусть у ассистента будет своя конфигурация запуска

Панель Generate NC хранит именованные конфигурации: какой проект обрабатывать, куда писать файл УП, какие входные параметры поста использовать. Попросите ассистента создать свою — например `Agent check` — вместо того чтобы пользоваться вашей. Ваши настройки при этом останутся как были, а два прогона можно будет сравнить.

Контекст и безопасность

Давайте ассистенту небольшой набор достоверных источников: нужный обработчик, соответствующую страницу справочника CLData, уменьшенный тестовый проект и заведомо правильную УП. Полный проект заказчика ответ улучшает редко.

Требования, которые действуют всегда, записывайте в файл инструкций воркспейса (`CLAUDE.md`, `AGENTS.md`, `.clinerules` — смотря какой клиент): правила именования, формат вывода, участвующие станки, обязательный запуск постпроцессора до доклада об успехе. Не помещайте туда пароли, токены и архивы заказчиков, а сформированные УП держите вне папок с исходниками.

Перед изменением нескольких файлов просите план, храните каждое изменение в системе контроля версий и никогда не позволяйте ассистенту передавать программу на станок.

Что дают расширения

Видимую работу делают три расширения: одно для редактирования и запуска SPPX-постпроцессоров, одно для просмотра входных данных, одно для постпроцессоров на C#. Ассистент пользуется теми же командами, но это обычные возможности VS Code — здесь описано, чего от них ждать и где их найти.

Команды запускаются из палитры команд: `F1` (или `Ctrl+Shift+P`), затем начните набирать название. Команды, у которых в названии есть (agent API), предназначены ассистенту — вручную они вам не нужны.

SPPX Postprocessor Tools

Редактирование

Откройте файл `*.sppx` — структура XML и код обработчиков подсвечиваются как два разных языка. XML-обвязка свёрнута, поэтому файл читается как список обработчиков с их кодом. Панель Outline перечисляет обработчики, подпрограммы и регистры и переходит к ним, `F12` ведёт к определению, автодополнение и подсказки сигнатур охватывают встроенный язык, а синтаксические ошибки подчёркиваются по мере ввода.

Если VS Code восстановил старую раскладку сворачивания и файл выглядит непривычно, выполните SPPX: Unfold Code (reset folding to default layout).

Генерация УП

Панель Generate NC — основное рабочее место. При открытом файле `*.sppx` она вызывается кнопкой-треугольником в заголовке редактора — при наведении подсказка показывает SPPX: Generate NC Program — или одноимённой командой из палитры.

В панели:

- именованные конфигурации запуска — обрабатываемый проект CAM, выходной файл и собственные входные параметры постпроцессора, показанные с их заголовками, а не служебными именами;
- Run и ход выполнения во время генерации;
- ошибки и предупреждения в панели Problems, каждое со ссылкой на строку обработчика, которая его породила;
- сформированная программа со сравнением с предыдущим прогоном и с эталонной программой, которую вы укажете.

Кнопки рядом с Run открывают CLData текущей конфигурации в инспекторе: имя проекта — в боковой панели, стрелка — во вкладке под редактором. То же доступно командой SPPX: Inspect CLData of the Current Run Configuration.

Ещё две команды стоит знать: SPPX: Open in Postprocessors Generator IDE открывает тот же постпроцессор в InP, а SPPX: Clean Stored Run Data сбрасывает сохранённое состояние прогонов, если конфигурация пришла в негодность.

CLData Inspector

Здесь видно, что фактически приходит в постпроцессор. Его значок на панели активности открывает три представления:

- Files — файлы открытого проекта с количеством команд. Кнопки представления открывают проект CAM (`*.stcp`, `*.stc`) или отдельный файл `*.inpcld`, повторно открывают недавний, перечитывают проект после перегенерации и открывают широкую вкладку инспектора в области редактора.
- Commands — команды выбранного файла, сгруппированные по секциям структуры; в файлах подпрограмм УП — по границам подпрограмм. `Ctrl+F` в представлении или команда CLData: Find Command… ищет на стороне сервера, поэтому поиск остаётся пригодным и на файлах с сотнями тысяч команд.
- Parameters — `Key = Value` для выбранной команды: массив, именованные параметры и типизированное .NET-представление. Последнее включается и выключается командой CLData: Toggle .NET Parameter View.

Щёлкните параметр правой кнопкой, чтобы скопировать нужное для кода: Copy Name (sppx) и Copy Name (.NET) дают точную форму для вставки в обработчик, Copy Value — само значение. На команде Copy Caption копирует её человекочитаемое имя.

Ассистент может открыть здесь конкретный файл, команду или параметр, когда хочет вам что-то показать, — см. разработку в Visual Studio Code.

DotNet Posts

Для постпроцессоров на C#, с той же панелью Generate NC, что и у SPPX Tools: конфигурации, входные параметры, ход выполнения, результаты и сравнение.

Два способа начать генерацию:

- DotNet Posts: Generate NC Program или кнопка-треугольник в заголовке редактора для файла C# (в подсказке то же название) — работает с открытым в воркспейсе проектом постпроцессора и сам находит собранную сборку;
- DotNet Posts: Generate NC Program from DLL… или щелчок правой кнопкой на `.dll` в Explorer — запускает скомпилированный постпроцессор без исходников. Несколько постпроцессоров можно держать в отдельных вкладках панели и сравнивать на одном проекте.

Команда DotNet Posts: Reveal Postprocessor Handler открывает обработчик нужной команды. Настройка `dotnetPosts.autoBuild` собирает проект перед каждым запуском, чтобы изменение не проверялось на устаревшей сборке.

Postprocessor Tools updater

Своего интерфейса у модуля обновления нет. Когда доступна новая версия, в панелях постпроцессора появляется действие обновления: согласитесь на него и перезагрузите окно. Та же проверка доступна из палитры — см. расширенную настройку.

Настройки

Страница параметров каждого расширения начинается с папки, в которую установлена CAM-система, — это единственная настройка, без которой ничего не работает. В расширенной настройке перечислены все три и выбор папки через диалог.

Рабочий цикл SPPX

SPPX-постпроцессор — это набор обработчиков (по одному на команду CLData), а также подпрограммы, объекты и регистры, написанные на встроенном Паскале-подобном языке, с форматированием вывода, описанным масками. Ассистент работает именно с этими элементами через сервер InP MCP, и цикл всегда один и тот же: прочитать, изменить, скомпилировать, запустить, сравнить.

Справочник по языку и маскам — в документации Генератора постпроцессоров, смысл входных команд — в справочнике CLData.

Цикл

1. Подготовьте базу. Берите за основу ближайший из существующих постпроцессоров — тот, что используется для этого станка, или дистрибутивный для похожей системы ЧПУ, — а не пустой файл. Скопируйте его или зафиксируйте текущее состояние, чтобы у сравнения была отправная точка. Сформируйте УП один раз до любых изменений и сохраните её.
2. Найдите данные. Определите команду CLData, из которой получаются нужные кадры УП, и прочитайте её фактические параметры в тестовом проекте. Названия команды недостаточно: в разных проектах одна и та же команда несёт разные значения.
3. Найдите обработчик. В структуре постпроцессора перечислены обработчики, подпрограммы и объекты. Обработчик, названный по команде, — это место, где начинается обработка, но вывод может формироваться дальше: в подпрограмме, в маске или при следующей смене модального состояния.
4. Прочитайте перед записью. Ассистент обязан прочитать текущий код того элемента, который собирается изменить, и описания регистров, если дело касается регистров. Правка непрочитанного — это то, как исчезает посторонний код.
5. Меняйте минимум. Сохраняйте порядок и формат регистров, модальное поведение, разделители, состояние в `Common` и локальных переменных и принятые в посте соглашения о подпрограммах. Донорский код из другого постпроцессора нужно проверять по данным этого проекта, а не вставлять как есть.
6. Скомпилируйте. Все выданные ошибки нужно исправить, прежде чем идти дальше: запускать несобравшийся постпроцессор бессмысленно.
7. Запустите. Интерпретируйте тот же тестовый проект и получите УП. Изменение, которое ни разу не запускали, не завершено, как бы ни выглядел код.
8. Сравните. Сопоставьте новую УП с исходной, затем с эталонной программой со станка. Проследите каждое отличие до команды и обработчика — включая отличия, которых вы не ожидали.
9. Проверьте. Перейдите к проверке и подтверждению результата.

Что ассистент может изменить, а что нет

На каждую команду CLData есть один обработчик: обработчики можно править, но нельзя создать или удалить. Создавать можно подпрограммы и объекты. Удаление элемента необратимо, поэтому ассистент сначала спрашивает — прочитайте запрос прежде, чем подтвердить.

Наблюдение за работой

Просите оконный экземпляр InP, когда хотите видеть среду: изменённый обработчик, сообщения компиляции и сформированная программа оказываются перед вами. Безоконный экземпляр нужен для пакетных проверок и параллельных прогонов — на экране при этом ничего не появляется. Если открыто несколько окон InP, скажите, в каком работать.

В VS Code вы также можете попросить открыть изменённый обработчик в редакторе или подготовить прогон Generate NC, чтобы самому увидеть панель, параметры и сравнение.

Что проверять в коде

Формат регистров и модальный вывод — то место, где черновики ИИ ошибаются чаще всего: значение выводится безусловно вместо вывода по изменению, модальный регистр остаётся установленным, слово оси выводится с неверным знаком или не в том порядке. Просмотрите каждую условную ветвь и каждый оператор прямого вывода и проверьте значения, которых нет в тестовом наборе: пропущенные параметры, повторяющиеся команды, границы и максимумы, переходы в начале и конце программы.

Не допускайте появления имён свойств .NET в коде SPPX. Обе подсистемы читают одни и те же данные разным синтаксисом: страницы CLData описывают смысл, страницы SPPX — форму доступа, и нумерация элементов в них различается.

Рабочий цикл .NET

.NET-постпроцессор — это проект на C#, собираемый с SDK постпроцессирования: типизированные обработчики, `.csproj` и скомпилированная сборка, которую исполняет пакетный генератор. Ассистент работает здесь так же, как в любом проекте на C# — языковой сервис даёт расширение C#, сборку выполняет `dotnet`, — а специфика постпроцессора сводится к шаблону, файлу настроек и генератору.

Справочник по API — в документации по .NET-постпроцессорам, входные данные — в справочнике CLData.

С чего начинать

Чаще всего не с нуля. Возьмите ближайший из уже существующих постпроцессоров — поставляемый с CAM-системой или написанный ранее для похожего станка или системы ЧПУ, — скопируйте и доработайте. Вы начинаете с кода, который уже работает, и разбираться приходится именно с теми отличиями, которые действительно важны.

Если новый постпроцессор действительно нужен, создайте его в CLData Viewer: CLData viewer → Create new postprocessor → Simple postprocessor⋯, затем укажите пустую папку — её имя станет именем постпроцессора. Проект создаётся за вас, а из контекстного меню он открывается в VS Code. В документации по .NET-постпроцессорам это описано в разделе «How to prepare the computer to start writing the postprocessors». Просить ассистента создать проект из шаблонов стоит только если вы хотите именно этого — это его работа, а не ваша.

Если вы копируете .NET-постпроцессор в папку с новым именем, переименуйте и файл `.csproj`, задав ему то же имя. Имя папки и имя `.csproj` должны совпадать: задача сборки из шаблона берёт путь к проекту из имени папки, а конфигурация отладки — имя сборки из него. Если имена различаются, обе конфигурации ищут несуществующие файлы.

Работайте с ним, открыв в VS Code папку самого постпроцессора: этого ждут языковой сервис C#, сборка и отладчик. Чтобы постоянная памятка и заметки при этом оставались доступны, нужен один приём — он описан в организации рабочего пространства.

В любом случае начинайте с известного состояния:

1. Поставьте постпроцессор под контроль версий, прежде чем что-либо менять.
2. Соберите его без изменений. Если он не собирается как есть, дальнейшая диагностика бессмысленна.
3. Сформируйте им УП и сохраните её как исходный результат для сравнения.

Цикл

1. Исследуйте CLData тестового проекта и найдите нужную команду. Инспектор показывает каждую команду и в типизированном .NET-представлении — интерфейс, имя обработчика и значения свойств — и умеет копировать имя параметра в том виде, в котором его вставляют в код C#.
2. Найдите типизированный обработчик этой команды. Попросите ассистента перечислить обработчики проекта и открыть нужный.
3. Внесите одно небольшое изменение, заранее сформулировав ожидаемый результат в УП. Пользуйтесь именованными типизированными свойствами SDK, а не индексами, там где SDK их предоставляет.
4. Соберите проект. Относитесь к предупреждениям как к сведениям о вашем изменении, а не как к шуму, который надо заглушить.
5. Запустите на тестовом проекте и прочитайте отчёт о запуске: сформированный файл УП, сообщения, код результата.
6. Сравните полученную УП с эталоном и перейдите к проверке и подтверждению результата.

Запуск из VS Code

DotNet Posts выполняет запуск и показывает отчёт в панели Generate NC — так же, как это делает SPPX Tools. Две точки входа покрывают оба случая: Generate NC Program работает с открытым в воркспейсе проектом постпроцессора и сам находит собранную сборку, а Generate NC Program from DLL… принимает скомпилированный постпроцессор, на который вы указываете, без исходников — что позволяет держать несколько постпроцессоров в отдельных вкладках Generate NC и сравнивать их вывод на одном проекте.

Настройка `dotnetPosts.autoBuild` заставляет расширение собрать проект перед запуском, чтобы правка не проверялась на устаревшей сборке. В панели есть и команды, которыми ассистент работает вместе с вами: перечислить и открыть обработчики, подсветить участок кода, сообщить, где стоит ваш курсор, и открыть текущий CLData в инспекторе.

Запуск постпроцессора, к которому нет исходников

Постпроцессор несёт настройки по умолчанию внутри сборки, и генератор умеет выгрузить их в виде файла `Settings.xml` рядом с входными данными — именно это делает возможной команду Generate NC Program from DLL…. Пользуйтесь файлом, который сформировал генератор: формат не совпадает с форматом SPPX, поэтому написанный вручную не подойдёт. Если настройки извлечь не удаётся, установленная CAM-система старее, чем требует этот сценарий, — проверьте её версию, прежде чем подозревать постпроцессор.

Запуск считается завершённым только тогда, когда отчёт и результат процесса согласуются. Молча завершившийся прогон не дал ничего, что стоит сравнивать.

Что проверять в коде

Просмотрите обработку null и границ, единицы измерения, зависящее от локали форматирование чисел, кодировку и переводы строк выходного файла, модальное состояние между командами, коррекцию инструмента, преобразования координат и ветви обработки исключений. Сформированный код на C# нужно собрать и прогнать на представительных данных, прежде чем ему доверять: для типизированного API компилятор ловит меньшую долю реальных ошибок, чем кажется.

Проверка и подтверждение результата

Помощь ИИ меняет то, как получается черновик. Она не меняет того, как принимается постпроцессор. Успешная сборка и чистый прогон доказывают, что программы сработали, — и не более того.

Указывайте, до какого предела дошла проверка

Пользуйтесь одним из четырёх состояний и сохраняйте подтверждение для него. Смысл в том, чтобы незавершённая проверка была видна, а не чтобы всё называлось «готово».

Состояние	Что означает
Не проверено	Есть план, изменения или сборка, но постпроцессор не запускался на зафиксированном проекте
Технически проверено	Запуск завершился ожидаемо, сохранены лог, код результата и сформированный файл; содержание УП не разбиралось
Проверено на тестовом проекте	Исходная и новая УП сравнены на обычных тестовых операциях и применимых граничных сценариях — например, в начале и конце программы, при переходах между операциями и инструментами, повторе модальных команд, а также на минимальных и максимальных значениях параметров; параметры станка сверены
Одобрено для производства	Квалифицированный специалист подтвердил УП для конкретного станка, системы ЧПУ и области применения

Применять в производстве позволяет только последнее состояние, и присвоить его может только человек. Никогда не описывайте результат как готовый или безопасный на основании кода завершения процесса.

Проверка изменений в исходном коде

- Изменения содержат то, о чём просили, и ничего больше: ни переформатирования, ни посторонних переписываний.

- Названия команд, параметры, единицы измерения, системы координат и предположения о модальности соответствуют справочнику CLData и фактическим данным проекта.
- Постпроцессор компилируется, и сообщения компиляции прочитаны, а не пролистаны.
- Проверены обычные, пропущенные, повторяющиеся, максимальные и неподдерживаемые входные значения, а не только один случай из тестового набора.
- Имя выходного файла, кодировка, переводы строк и обработка ошибок такие, каких ожидает станок.

Проверка УП

Сравнивайте с заведомо правильной программой и начинайте с первого отличающегося кадра: отличие в начале программы часто объясняет всё, что идёт после. Каждое отличие проследите до команды CLData и кода, который его породил.

Проверьте программу в целом:

- выбранный станок, систему ЧПУ, единицы измерения и рабочую систему координат;
- начало и конец программы, включая финальное позиционирование и останов;
- смену инструмента, шпиндель, подачи, охлаждение и безопасные позиции;
- знаки, порядок осей, коррекции, смещения нулей и отходы;
- циклы, подпрограммы, вращательные перемещения, ограничения и синхронизацию;
- отсутствие неожиданных перемещений, пропущенных команд и лишних кадров;
- соответствие программы исходной траектории — на тестовом участке и на пограничных случаях.

Затем прогоните её через симулятор и через ту проверку для конкретного станка, которой вы обычно пользуетесь. Предупреждения разбирайте, а не подавляйте.

Сохраняйте подтверждения

Храните вместе изменения исходного кода, входной проект, версию постпроцессора, отчёт о запуске и сформированную программу и записывайте, кто и когда проверял. Когда проблема всплывёт позже, именно это позволит отличить изменение постпроцессора от изменения данных.

Чего ассистент делать не должен

Он не должен утверждать код, менять настройки станка, чтобы вывод «прошёл», обходить проверку и передавать программу в систему ЧПУ. Между сформированным результатом и применением в производстве остаётся утверждение человеком.

Устранение проблем

Ассистент не видит инструменты для постпроцессоров

Спросите, какие инструменты у него сейчас есть. Если инструментов CLData или InP среди них нет, значит конфигурация MCP не загрузилась: проверьте, что вы правили тот файл, который действительно читает ваш клиент, что пути абсолютные и существуют, что в JSON обратные слэши удвоены, и перезапустите клиент. Клиент, запущенный до сохранения конфигурации, продолжает работать со старым набором.

Ассистент не может подключиться к InP

InP MCP управляет запущенным экземпляром среды разработки постпроцессоров. Дайте ассистенту запустить его самому или откройте InP из CAM-системы с загруженным постпроцессором и попросите подключиться. Если он сообщает, что менеджер экземпляров недоступен, установленная CAM-система не поддерживает запуск экземпляров — откройте InP вручную.

Безоконный экземпляр, который не компилирует и не запускает пост, сообщая об ошибке фокуса невидимого окна, — это старая сборка InP. Используйте оконный экземпляр.

Запущено несколько экземпляров InP

Ассистент обязан указать конкретный экземпляр, и запрос без такого указания отклоняется, а не отправляется в произвольный экземпляр. Закройте ненужные экземпляры или скажите, в каком работать, — при осознанном параллельном прогоне нескольких постпроцессоров это нормальная ситуация.

Индексы команд перестали соответствовать данным

Индексы CLData действительны для одной ревизии данных. Когда проект перегенерировали, сервер перечитывает его и просит повторить вызов; результаты, полученные раньше, указывают на другие команды. Всегда храните проект вместе с индексами и повторяйте исследование после перегенерации CLData.

Расширение сообщает, что папка установки не задана

Задайте её в параметрах расширения или выполните его команду Select Installation Folder… и выберите папку. Принимается и папка продукта, и её подпапка `Bin64`, кавычки вокруг вставленного пути тоже допустимы. Если установлено несколько версий CAM-системы, убедитесь, что путь указывает на ту, против которой вы действительно хотите проверять, — это самая частая причина расхождений «в CAM-системе работает иначе».

Результат не совпадает с тем, что даёт CAM-система

Если установлено несколько версий CAM, запуск мог пройти через другую — и никакого предупреждения при этом не бывает: он завершается успешно и просто даёт другой результат. Проверьте, что все три настройки расширений указывают на ту версию, против которой вы разрабатываете.

Для .NET-постпроцессора расширение удерживает отладку на той же установке, что и панели, и сообщает, когда ему приходится перекрыть переменную окружения `InpCoreDir`. Если же вы запускаете постпроцессор мимо расширения — из терминала или из собственной конфигурации, — решает именно она, а в ней оказывается та установка, чей CLData Viewer запускался последним. Её изменение не действует на уже запущенные программы; чтобы новое значение подействовало везде, нужен выход из сеанса Windows и повторный вход.

Generate NC не создаёт файл

Сначала прочитайте ошибки в Problems: постпроцессор, который не скомпилировался или остановился на ошибке выполнения, ничего не формирует. Затем проверьте, что путь вывода доступен для записи и что в конфигурации указан именно тот проект, который вы имеете в виду. Для .NET проверьте, что сборка была пересобрана после последней правки, — это делает `dotnetPosts.autoBuild`.

Не удаётся извлечь настройки из .NET-постпроцессора

Извлечение настроек по умолчанию из сборки требует достаточно свежего пакетного генератора. Если установленный так не умеет, возьмите проект, у которого файл настроек есть, или обновите CAM-систему; не пишите файл настроек вручную в обход этого.

install.cmd закрывается раньше, чем я успеваю прочитать

Он ждёт нажатия клавиши, поэтому окно, закрывшееся сразу, означает, что запускался не он. Запустите его из командной строки в распакованной папке, чтобы вывод остался на экране.

Если он сообщает, что команда `code` не найдена, значит команда VS Code недоступна в `PATH`. Установщик VS Code обычно добавляет её сам; иначе добавьте содержащую её папку в `PATH` или установите `.vsix` через Extensions → Install from VSIX⋯.

Модуль обновления ничего не предлагает

Обновления настраивает сам набор, поэтому расширениям, установленным вручную из отдельных `.vsix`, они никогда не предлагаются. Переустановите их через `install.cmd` из распакованного архива, затем выполните Postprocessor Tools: Check for Updates.

Ассистент отвечает по документации вместо проекта

Так происходит, когда инструменты недоступны, когда в запросе не назван проект или когда от ответа не требуют ссылок на данные. Дайте проект, спросите, какую команду и какие значения параметров он на самом деле прочитал, и дайте skills сделать свою работу — они требуют исследования до правки.

Ассистент докладывает об успехе, ничего не запустив

Компиляция — это не результат. Спросите сформированную УП и сравнение с исходной; если их нет, состояние — не проверено, см. Проверка и подтверждение результата.

Документация, которую читает ассистент, устарела

Локальная копия — это снимок: обновляйте её командой `git pull` и записывайте ревизию, когда на ней основано решение о выпуске. Зависящие от версии подробности — версии SDK, названия команд, вид панелей — сверяйте с установленным продуктом, а не с копией.

Расширенная настройка

Эта страница — полная процедура настройки и справка для проверки того, что было сделано. Передайте её ассистенту, как описано в настройке инструментов, выполняйте по ней вручную, если автоматическая настройка недоступна, или читайте, когда команде нужен воспроизводимый результат.

О чём идёт речь

Если эту страницу читает ассистент, чтобы настроить инструменты, — вот поле, на котором он находится.

CAM-система рассчитывает траекторию и выгружает её в виде CLData — потока команд и параметров, описывающего обработку независимо от конкретного станка. Постпроцессор превращает этот поток в управляющую программу для одного конкретного станка и его системы ЧПУ: её коды, её форматы, её соглашения. Каждому станку в цеху нужен свой, и доработка такого постпроцессора — ежедневная работа того человека, которому вы помогаете.

Постпроцессоры бывают двух видов, и они не смешиваются:

- SPPX — набор обработчиков, по одному на команду CLData, на встроенном Паскале-подобном языке, исполняется InP;
- .NET — проект на C# с SDK постпроцессирования, исполняется `InpCore.exe`.

Оба читают один и тот же CLData. Именно поэтому чтение фактических данных проекта всегда идёт до правки кода.

Для чего настраиваются инструменты: читать входные данные через сервер CLData, читать и изменять постпроцессор через сервер InP или через проект на C#, компилировать его, запускать на реальном тестовом проекте и сравнивать сформированную УП с заведомо правильной. Расширения VS Code — окно человека во всё это: они показывают найденные данные, изменённый код и полученную программу. Правильность результата определяет человек, с которым вы работаете; вы готовите основания, по которым он судит.

Поэтому настройка состоит из четырёх частей, в таком порядке: расширения, папка, которая нужна каждому из них, MCP-серверы и знания — эта документация и skills. После этого приведите в порядок само рабочее пространство: см. организацию рабочего пространства и skill `prepare-workspace`.

Сообщая о проблеме, зафиксируйте версии: CAM-системы, набора расширений, CAM Agent, MCP-серверов и AI-клиента. Большая часть сообщений «вчера работало» объясняется различием версий.

Установка расширений

Требуется Visual Studio Code 1.85 или новее. В распакованном архиве `install.cmd` устанавливает все лежащие рядом `.vsix` и заменяет прежние версии; ему нужна команда `code` в `PATH`, которую установщик VS Code добавляет по умолчанию. Без сценария установите каждый файл командой `code --install-extension <файл>.vsix` или через Extensions → Install from VSIX… в VS Code. Затем перезагрузите окно: `F1` (или `Ctrl+Shift+P`) открывает палитру команд, в ней выполняется Developer: Reload Window. Всё, что на этой странице выделено жирным и содержит двоеточие, — команда VS Code и запускается так же.

В наборе четыре расширения: SPPX Postprocessor Tools, CLData Inspector, DotNet Posts и модуль обновления.

Папка установки CAM для расширений

Каждому расширению нужна папка, в которую установлена CAM-система: там лежат программы, которыми оно управляет. Это первая настройка на странице параметров каждого расширения:

Настройка	Для чего	Программа
`sppx.installationFolder`	SPPX Tools	`InP.exe`
`cldata.installationFolder`	CLData Inspector	`InpCoreMCP.exe`
`dotnetPosts.installationFolder`	DotNet Posts	`InpCore.exe`

Принимается как папка продукта, так и её подпапка `Bin64`, а также путь, вставленный вместе с кавычками. Вместо ввода пути выполните команду расширения — например DotNet Posts: Select Installation Folder… — и выберите папку в диалоге; на странице параметров тот же выбор доступен ссылкой. SPPX Tools и CLData Inspector берут значение друг у друга, пока их собственная настройка пуста, поэтому при обычной установке путь задаётся один раз.

Подключение MCP-серверов

MCP — это то, что даёт ассистенту возможность действовать. Подходит любой MCP-клиент, и конфигурация везде одинакова — различается только файл, в который она попадает.

CLData MCP — `InpCoreMCP.exe`, в папке `Bin64` установленной CAM-системы. Работает автономно: открывает проект (`*.stcp`, `*.stc`) или отдельный файл `*.inpcld`, запущенная CAM-система не нужна, данные он не изменяет.

InP MCP — `inp-mcp-server.exe`, поставляется вместе с CAM Agent; при установке по умолчанию это `%LOCALAPPDATA%\CamAgent3\bin\inp-mcp-server.exe`. Он управляет средой разработки постпроцессоров и может либо запустить экземпляр cam, в оконном или безоконном режиме, либо подключиться к тому, который вы открыли из CAM-системы.

CAM MCP — `cam-mcp-server.exe`, рядом с предыдущим. Необязательный и не относящийся к постпроцессорам: он управляет запущенной CAM-системой. Подключайте его, когда нужные ответы лежат в исходном проекте, а не в CLData: станок и его схема, параметры операции, инструменты и их свойства, системы координат и установка заготовки. Он же умеет готовить то, что вы постпроцессируете: пересчитать траектории, выгрузить CLData активного проекта, создать и открыть тестовый проект. Полезен, когда тестовый случай нужно построить или поправить, а не просто прочитать.

```
{
  "mcpServers": {
    "inpcld": {
      "command": "C:\\Program Files\\<папка установки CAM>\\Bin64\\InpCoreMCP.exe",
      "args": []
    },
    "inp": {
      "command": "C:\\Users\\<пользователь>\\AppData\\Local\\CamAgent3\\bin\\inp-
mcp-server.exe",
      "args": []
    },
    "cam": {
      "command": "C:\\Users\\<пользователь>\\AppData\\Local\\CamAgent3\\bin\\cam-
mcp-server.exe",
      "args": []
    },
    "knowledge": {
      "command": "C:\\Users\\<пользователь>\\AppData\\Local\\CamAgent3\\bin\\rag-
mcp-server.exe",
      "args": []
    }
  }
}
```

Работа с постпроцессорами держится на первых двух; `cam` и `knowledge` — полезные дополнения, ненужное не включайте.

Пути указывайте абсолютные и помните, что в JSON обратные слэши удваиваются. Куда положить конфигурацию, зависит от клиента:

Клиент	Куда попадает конфигурация
Kilo Code	`kilo.jsonc` в корне проекта или `.kilo/kilo.jsonc`; глобально `~/.config/kilo/kilo.jsonc`. Файл пишет сам интерфейс: значок настроек → Agent Behaviour → MCP Servers
Roo Code	`.roo/mcp.json` в проекте или глобальный `mcp_settings.json` — MCP Servers → Edit Project MCP / Edit Global MCP
Cline	`cline_mcp_settings.json` — MCP Servers → Configure
Claude Code	`.mcp.json` в корне проекта или `claude mcp add`
Copilot Chat в VS Code	`.vscode/mcp.json`, секция `servers`
Codex CLI	`~/.codex/config.toml`, секция `[mcp_servers.inpcld]` с `command = "…"`

Там, где клиент поддерживает оба уровня, файл проекта имеет приоритет над глобальным. После изменения файла клиент нужно перезапустить. В таком файле оказываются локальные абсолютные пути, поэтому не коммитьте его в общий репозиторий.

Это обычные программы на том же компьютере: клиент запускает такую программу и разговаривает с ней напрямую через её стандартный ввод и вывод, поэтому ни порт, ни служба, ни отдельная учётная запись никому из них не нужны — слово «сервер» здесь ничего не говорит о сети. Поэтому одна конфигурация подходит любому клиенту, а если сервер не отвечает, дело в пути или в правах. Исключение одно — сервер знаний: поиск по документации уходит в онлайн-сервис, а всё, что он закэшировал, продолжает работать и без него.

Проверка подключения

Возьмите непроизводственный проект и по одному небольшому запросу на чтение к каждому серверу: список файлов проекта CLData и пинг InP со структурой открытого постпроцессора. Сверьте ответы с тем проектом, который назвали. Никогда не включайте в проверку связи запись, правку исходников, передачу УП или подключение к станку.

Документация и skills для ассистента

Кроме инструментов ассистенту нужен справочный материал.

Источников два, и они отвечают на разные вопросы.

- Сервер знаний — `rag-mcp-server.exe`, в папке CAM Agent рядом с остальными серверами. Добавляется в конфигурацию MCP тем же способом и больше ничего не требует. Он держит снимок публичной документации — по продукту в целом и по постпроцессированию в частности — и ищет по нему по ключевым словам и по смыслу, читает страницу целиком, перечисляет и ищет публикуемые skills. Он же умеет проиндексировать проект CAM, чтобы искать по его операциям, инструментам и станку. Именно по нему отвечает CAM Agent, и тот же сервер работает с любым MCP-клиентом. Будучи снимком, он статичен по своей природе: он говорит, что умеет продукт, а не что решила ваша команда. Он держит локальный кэш, поэтому попросите ассистента однажды его обновить — после этого поиск по skills работает и без сети.

  Публикуемый набор может пополняться и с вашей стороны: написанный вами skill можно отправить на рассмотрение, и после принятия он войдёт в него для всех. Отправка выполняется из чата CAM Agent и только для того материала, который вы готовы опубликовать.

- Документация рабочего пространства — ваша собственная, и она пополняется. Ваши проекты, приёмы, которые работают на ваших станках, соглашения, принятые в вашей команде, разобранные вами случаи. Это та часть, которую ассистенту больше взять негде, и именно она делает его ответы вашими, а не общими. Как её организовать — отдельная тема: организация рабочего пространства.

  Актуальное руководство открывается через центр документации СПРУТКАМ. Передайте ассистенту ссылку на нужную статью или используйте сервер знаний. Если нужна локальная копия для работы без сети, сохраните материалы с указанием версии и даты.

Skills, публикуемые с этой документацией, лежат в папке `skills` этого раздела — каждый отдельной папкой с файлом `SKILL.md` внутри. Берите набор целиком: он пополняется вместе с инструментами, а ассистент использует тот skill, который подходит текущей задаче. В Kilo Code скопируйте папки в `.kilo/skills/` внутри проекта или в `~/.kilo/skills/` (`%USERPROFILE%\.kilo\skills\` в Windows), чтобы они были во всех проектах; новая сессия их подхватит, а команда `/reload` — без перезапуска. В остальных клиентах способ зависит от клиента: папка skills воркспейса или пользователя, запись в конфигурации агента или добавление вручную. Если в клиенте механизма skills нет, вставьте нужный `SKILL.md` в чат или сошлитесь на него из файла инструкций клиента (`CLAUDE.md`, `AGENTS.md`, `.clinerules` и подобных). CAM Agent ведёт собственный набор и хранит ваши рядом с ним.

Каким бы способом вы их ни подключили, проверьте, что они загружены: спросите у ассистента, какие skills и инструменты он сейчас видит.

Обновления по требованию

Когда новая версия доступна, панели предлагают действие обновления. Та же проверка запускается из палитры: Postprocessor Tools: Check for Updates, затем Postprocessor Tools: Install Available Updates, затем Developer: Reload Window. Обновления настраивает сам набор, поэтому расширениям, установленным вручную из отдельных `.vsix`, они никогда не предлагаются.

Несколько установок CAM-системы на одном компьютере

Каждое расширение указывает на одну папку установки, и ассистент запускает то, что в ней лежит. Если рядом стоят выпущенная и бета-версия, определитесь, против какой из них разрабатывается постпроцессор, и задайте её во всех трёх настройках. Переключаясь, переключайте все — смешанная конфигурация, в которой CLData читает одна версия, а постпроцессор запускает другая, даёт результаты, которые невозможно воспроизвести.

Есть один путь, которым эти настройки не владеют: переменная окружения `InpCoreDir`. Её записывает CLData Viewer при запуске, поэтому в ней оказывается та установка, чей просмотрщик запускался последним. Конфигурация отладки из шаблона .NET написана через эту переменную, но DotNet Posts подменяет папку своей настройкой в момент запуска отладки и один раз предупреждает при расхождении — так что отладка идёт по той же установке, что и панели. Переменная используется только тогда, когда настройка пуста.

Её изменение не мгновенно, и это стоит знать, когда вы всё же на неё опираетесь: уже запущенные программы держат старое значение, и на практике, чтобы новое дошло до всех, нужен выход из сеанса Windows и повторный вход.

Фиксация документации

Для работы, которая должна быть воспроизводимой, держите копию документации на известной ревизии, а не обновляйте её постоянно, и записывайте эту ревизию вместе с версией постпроцессора. Сервис поиска всегда отвечает по текущему индексу: это удобно для исследования и бесполезно, когда нужно объяснить, почему ответ изменился.

Отдельное рабочее место для ассистента

- Своя конфигурация запуска в панели Generate NC, чтобы ваши настройки остались как были.
- Своя папка для сформированных УП, вне дерева исходников.
- Ветвь или копия постпроцессора, никогда не производственный файл.
- Уменьшенный тестовый проект вместо полного проекта заказчика. Меньше данных — читаемое сравнение и более быстрый цикл.

- Никаких паролей, токенов, архивов заказчиков и посторонних репозиториев в воркспейсе и в файлах инструкций.

Автоматические проверки

Безоконный прогон подходит для воспроизводимой проверки: фиксированный входной проект, фиксированная ревизия постпроцессора, сохранённые логи и сохранённая УП. Полезной её делают два правила. Запускайте её в отдельной рабочей папке, которую не жаль потерять, чтобы неудачный прогон не оставил после себя полуизменённый постпроцессор. И разбирайте различия и вывод УП вне автоматики: автоматический сценарий может сообщить, что вывод изменился, но никогда — что изменение верно.

Не подключайте автоматическую цепочку к системе ЧПУ и не давайте ей права на запись в производственные постпроцессоры.

Правила, которые стоит записать в файл инструкций

Требования, действующие всегда, размещайте там, где ассистент читает всегда: в файле инструкций воркспейса или в skills. На практике оправдывают себя такие:

- читать фактические данные проекта до предложения изменений и приводить использованные значения;
- читать текущий код элемента перед его правкой;
- скомпилировать, затем запустить, затем сравнить с исходным результатом — до любого доклада о готовности;
- держать изменение минимальным и объяснять, что оно не покрывает;
- никогда не отправлять программу на станок и не менять настройки станка.