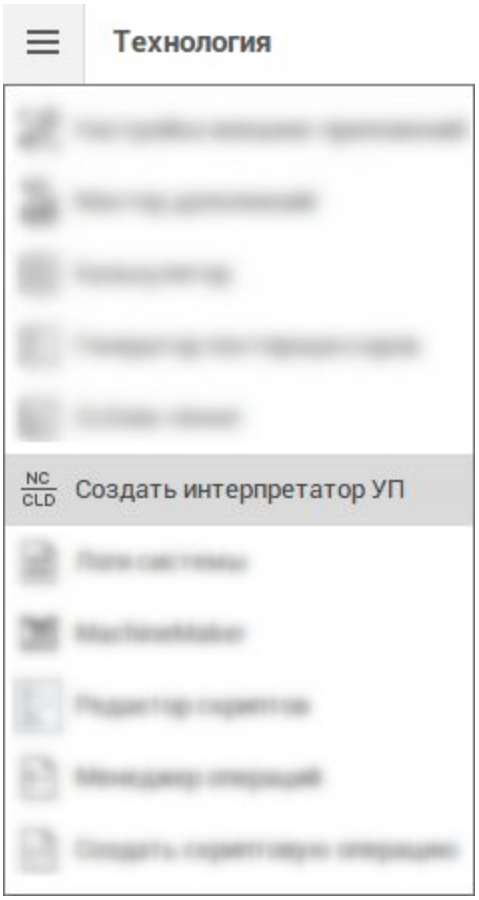
Создание собственного интерпретатора

Для создания собственного интерпретатора необходимо выполнить следующие шаги:

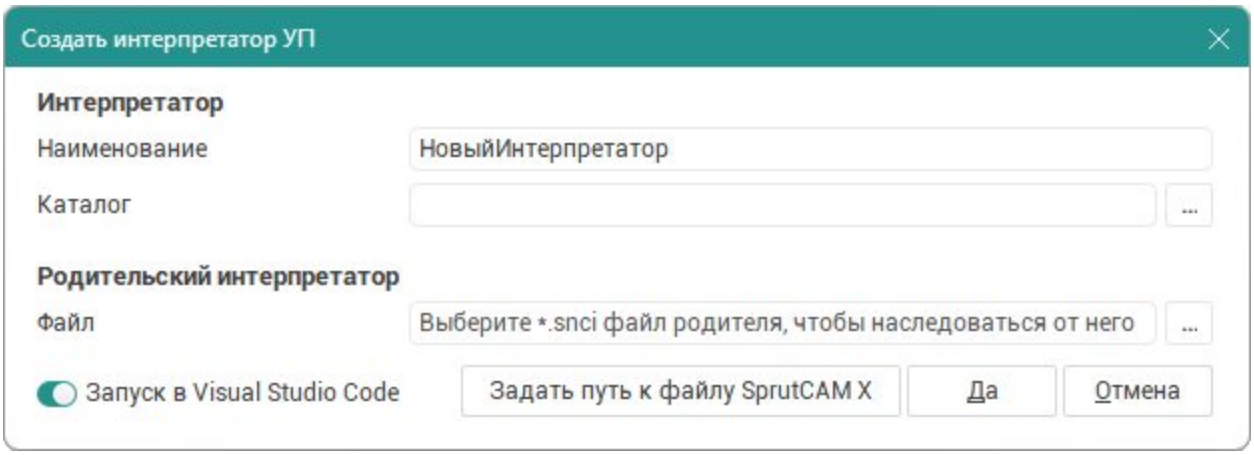
- Ознакомьтесь со статьями в текущем разделе, которые описывают взаимодействие CAM системы с интерпретатором при интерпретации текста управляющей программы.
- Создайте файл настройки интерпретатора и разработайте для него программную библиотеку в соответствии с описанием, представленным на следующих страницах: Файл настроек интерпретатора, Программная библиотека интерпретатора.
- Если вы хотите вести разработку с ИИ-ассистентом, начните с раздела Разработка интерпретаторов с помощью ИИ.

Разработка собственного интерпретатора на (.NET) C#

Для создания собственных интерпретаторов G-кода на основе .NET (C#) рекомендуется воспользоваться специальным инструментом. Это пункт Создать интерпретатор УП в списке Расширения.



При выборе пункта Создать интерпретатор УП появится соответствующее окно:



В нём необходимо заполнить лишь несколько параметров, на основе которых будет подготовлен шаблон для вашего интерпретатора:

- Шаблон — шаблон интерпретатора;
- Наименование — имя разрабатываемого интерпретатора;

> ⓘ ВНИМАНИЕ
>
> *На основании данного имени создаётся каталог с идентичным названием, в котором будут находиться все исходные файлы нового интерпретатора.*

- Каталог — директория, в которой расположится папка вашего интерпретатора (можно ввести значение вручную или воспользоваться диалоговым окном выбора каталога, нажав на соответствующую кнопку ).
- Файл — полный путь к файлу настройки родительского интерпретатора (*.snci), от которого вы хотите наследоваться (необязательное поле).
- Запуск в Visual Studio Code — переключатель: если он включён, папка интерпретатора сразу после создания откроется в Visual Studio Code.

После заполнения всех необходимых параметров, необходимо нажать кнопку <Да>. В результате в указанном вами каталоге будет создана папка с наименованием интерпретатора и данная директория сразу откроется в Visual Studio Code (в случае, если переключатель Запуск в Visual Studio Code был включён на тот момент).

> ⓘ ПРИМЕЧАНИЕ
>
> Visual Studio Code должен быть предварительно установлен на компьютере.

Что поправить после создания

Пункт Создать интерпретатор УП копирует шаблон и может заполнить не всё. Прежде чем собирать интерпретатор и запускать отладку (`F5`), проверьте поля ниже и исправьте те, что заполнены неверно. Если все поля уже верны, этот шаг пропустите.

- Версия пакета `SprutTechnology.NCInterpreterSDK.Net` в файле `*.csproj`: если стоит предварительная версия (с окончанием вроде `-dev` или `-rc`) или версия предыдущей основной версии, замените её выпущенной версией для вашей версии продукта, как описано в разделе Версии продукта, иначе проект может не собраться.
- Поле `ID` в файле `*.snci`: если оно заполнено нулями, впишите новый GUID.
- У дочернего интерпретатора — ссылка на родителя: в `ParentID` должно стоять значение `ID` родителя, а в `ParentFileName` — имя файла `*.snci` родителя; без него родитель не загружается. Если в `ParentID` стоит `LibraryID` родителя или `ParentFileName` пуст, исправьте их.

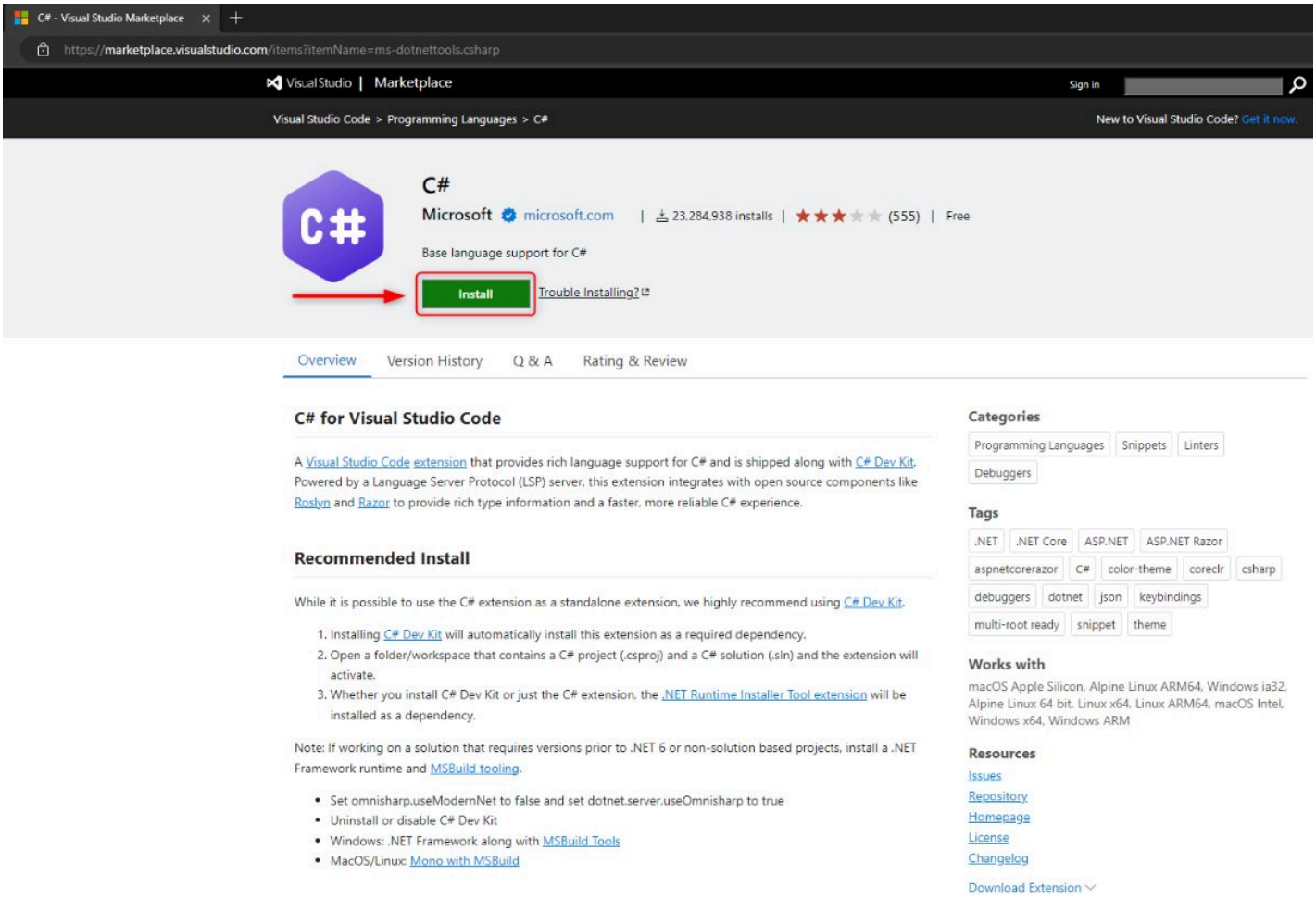
Подробнее — в разделе С чего начинать рабочего цикла .NET.

Отладка интерпретатора в Visual Studio Code

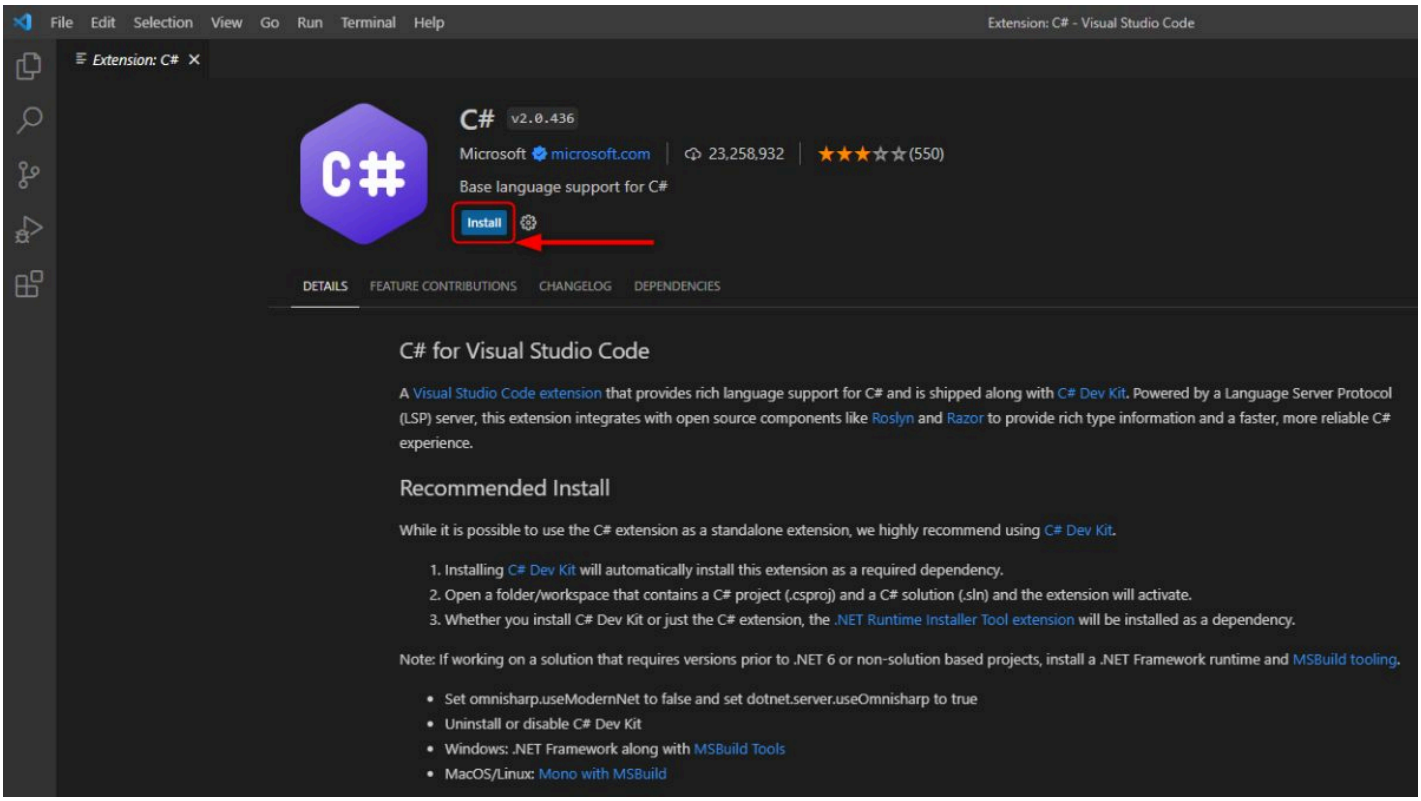
Для отладки собственных интерпретаторов G-кода на основе .NET (C#) необходимо выполнить следующие действия:

1. Скачать и установить последнюю доступную версию Visual Studio Code (https://code.visualstudio.com/download).

2. Установить расширение C# for Visual Studio Code из магазина расширений Visual Studio Code (https://marketplace.visualstudio.com/items?itemName=ms-dotnettools.csharp).



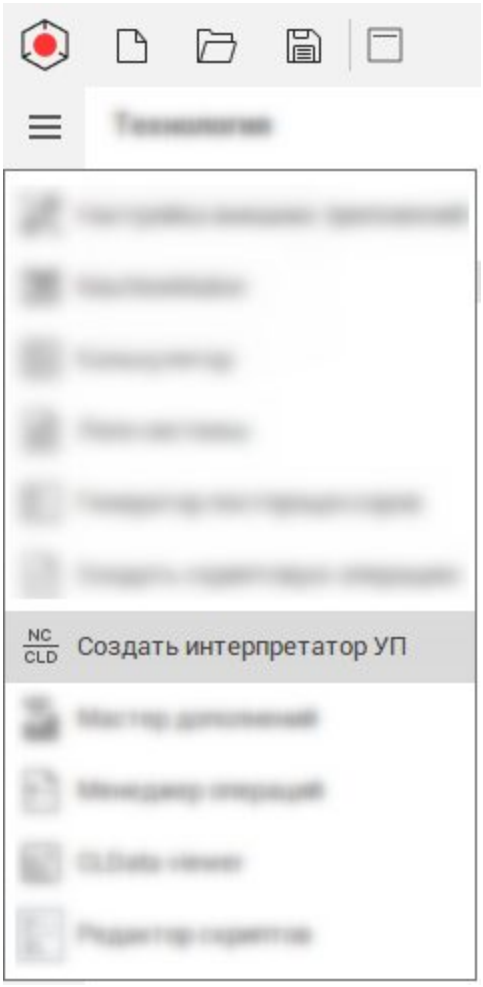
3. После нажатия на кнопку Install откроется VS Code (сокращение от Visual Studio Code) на вкладке Расширения. Просто нажмите кнопку Install рядом с названием расширения.



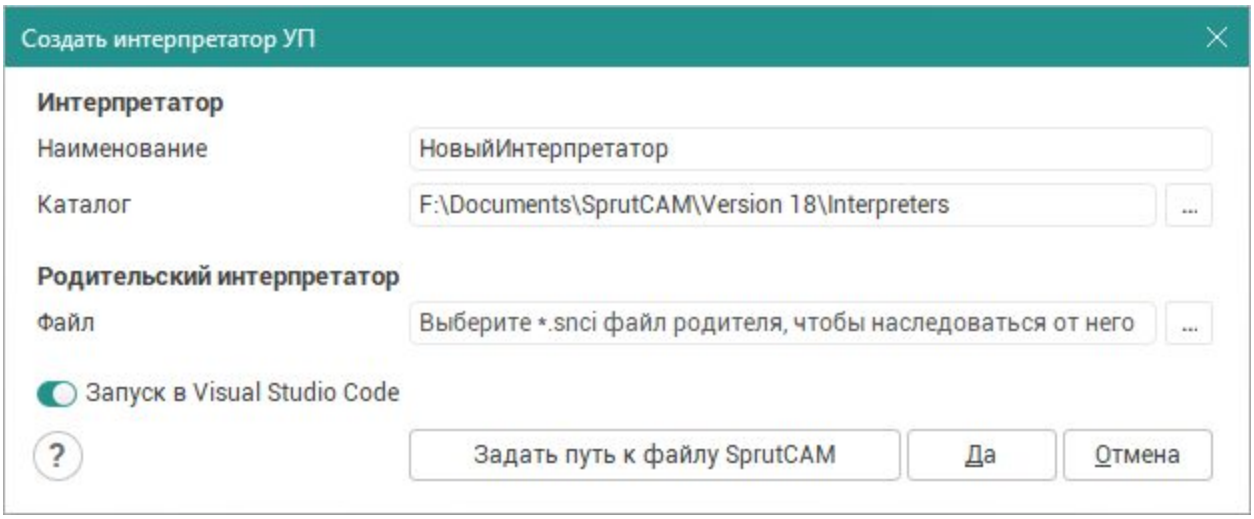
4. Для более удобного редактирования snci файлов рекомендуется (но не обязательно) установить расширение XML Language Support by Red Hat (https://marketplace.visualstudio.com/items?itemName=redhat.vscode-xml).

5. Если вы еще этого не сделали, необходимо скачать и установить последнюю версию СПРУТКАМ (требуется версия 20) с официального сайта (https://sprut.ru/download).

6. Далее нужно запустить СПРУТКАМ и выбрать пункт Создать интерпретатор УП в списке Расширения.



7. В появившемся окне требуется нажать кнопку Задать путь к файлу СПРУТКАМ и задать путь к файлу sc.exe, расположенному в каталоге установки СПРУТКАМ (по умолчанию это "C:\Program Files\Sprut Technology\SprutCAM 20\Bin64\sc.exe").



Этот процесс добавляет переменную среды 'NCKernelDir' в переменные среды окружения Windows. Чтобы можно было запустить Visual Studio Code из СПРУТКАМ, переменная среды "Path" должна включать папку установки VS Code. Обычно установщик Visual Studio Code добавляет её по умолчанию.

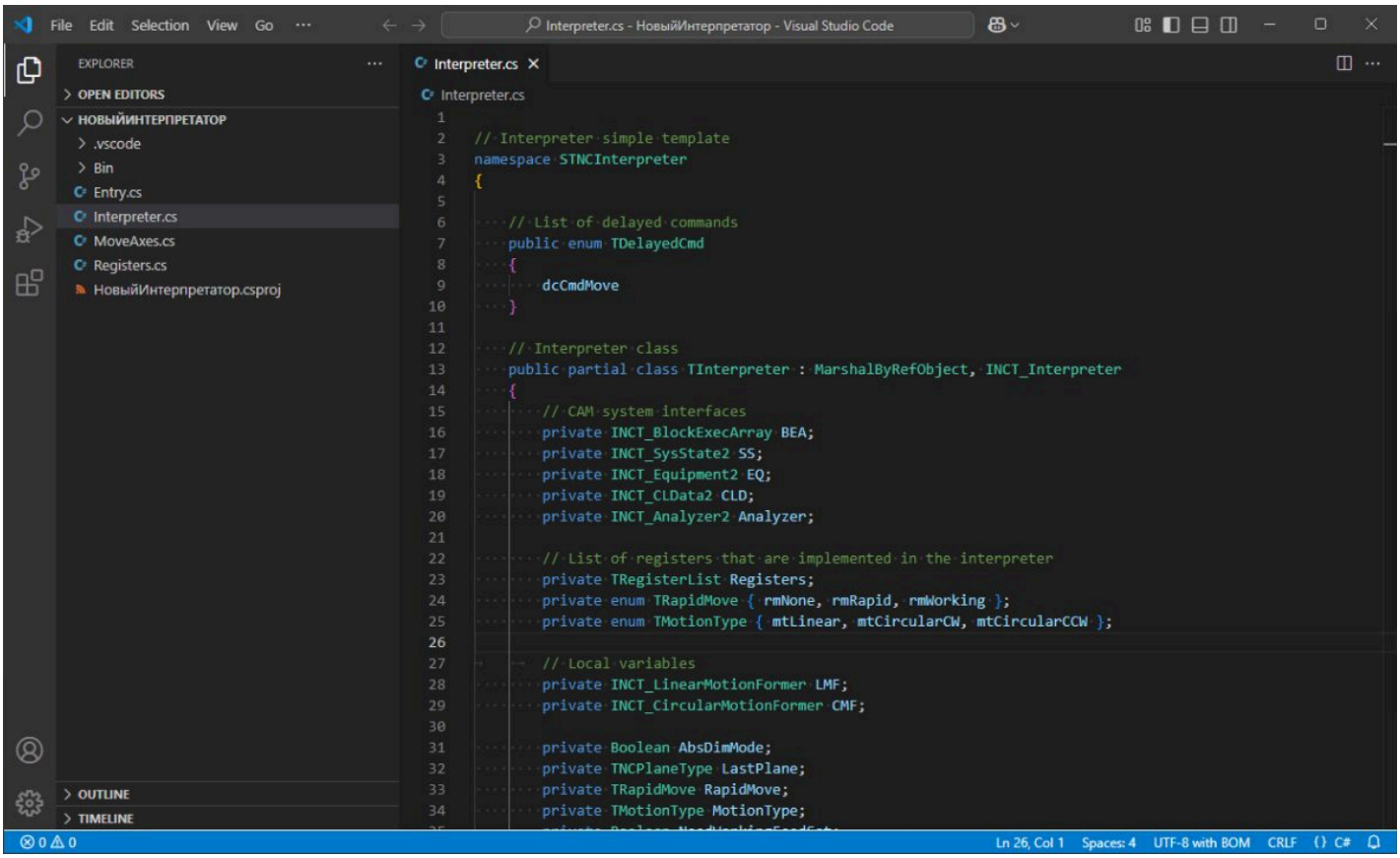
Чтобы отлаживать интерпретатор в NC Тюнере, выберите в NC Тюнере пункт Управление интерпретаторами (он открывает то же окно создания интерпретатора), нажмите в окне

кнопку Задать путь к файлу NCTuner 3 и укажите файл ltuner.exe из папки `bin64` каталога установки NC Тюнера. Так задаётся переменная среды 'TunerDir', по которой конфигурация отладки 'Launch Tuner' запускает NC Тюнер.

> ⓘ ПРИМЕЧАНИЕ
>
> Для вступления в силу изменений в настройках переменных среды окружения Windows, после задания пути к исполняемому файлу sc.exe (для NC Тюнера — ltuner.exe) и окончания установки VS Code выйдите из сеанса Windows и войдите снова или хотя бы запустите VS Code заново из меню «Пуск» или Проводника, а не из открытого раньше терминала.

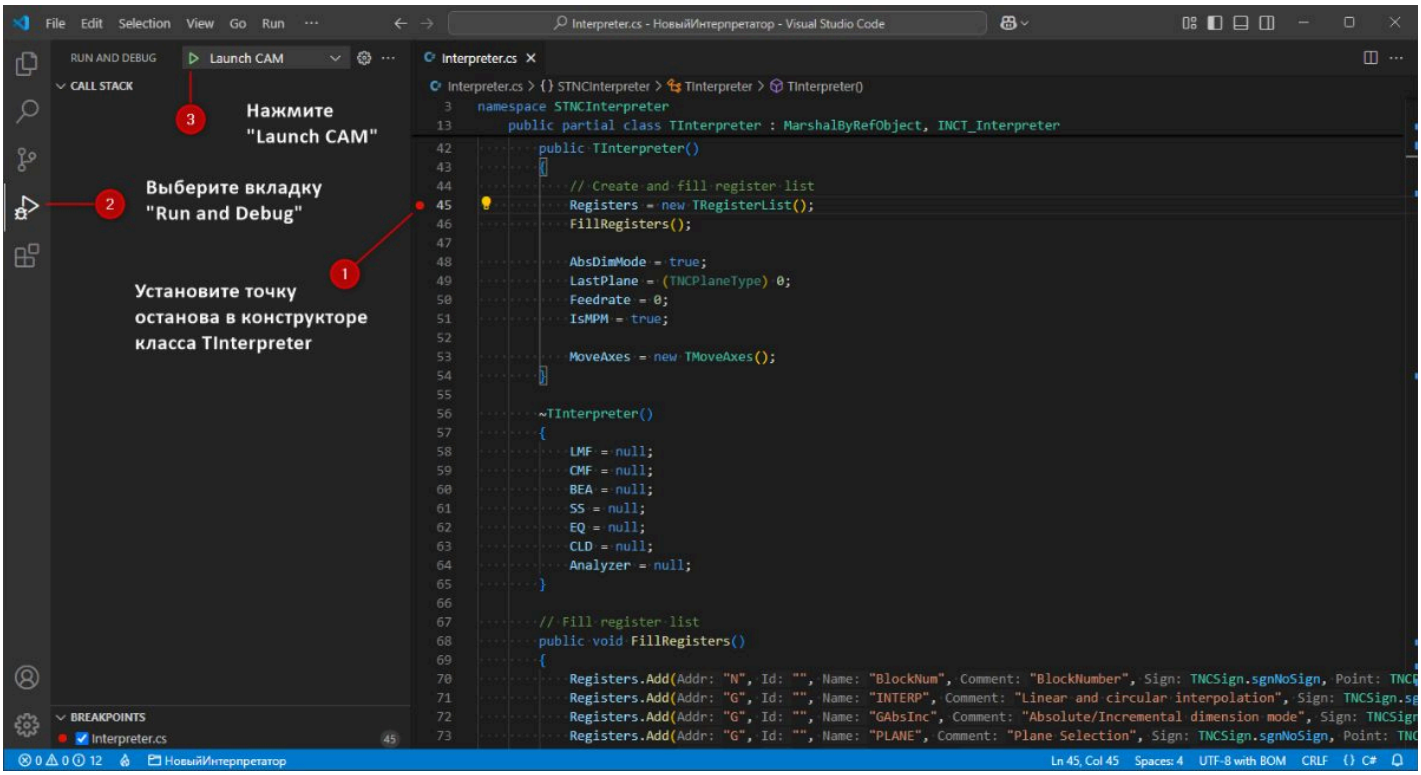
8. Затем снова запустите СПРУТКАМ и выберите пункт Создать интерпретатор УП в списке Расширения. Теперь вы можете создать новый интерпретатор, задав его имя в поле Наименование и директорию в поле Каталог, в котором будет создана папка, соответствующая имени интерпретатора, со всеми необходимыми исходными файлами.

9. После нажатия кнопки <Да> откроется новое окно Visual Studio Code с открытой вкладкой главного исходного файла интерпретатора Interpreter.Core.cs.



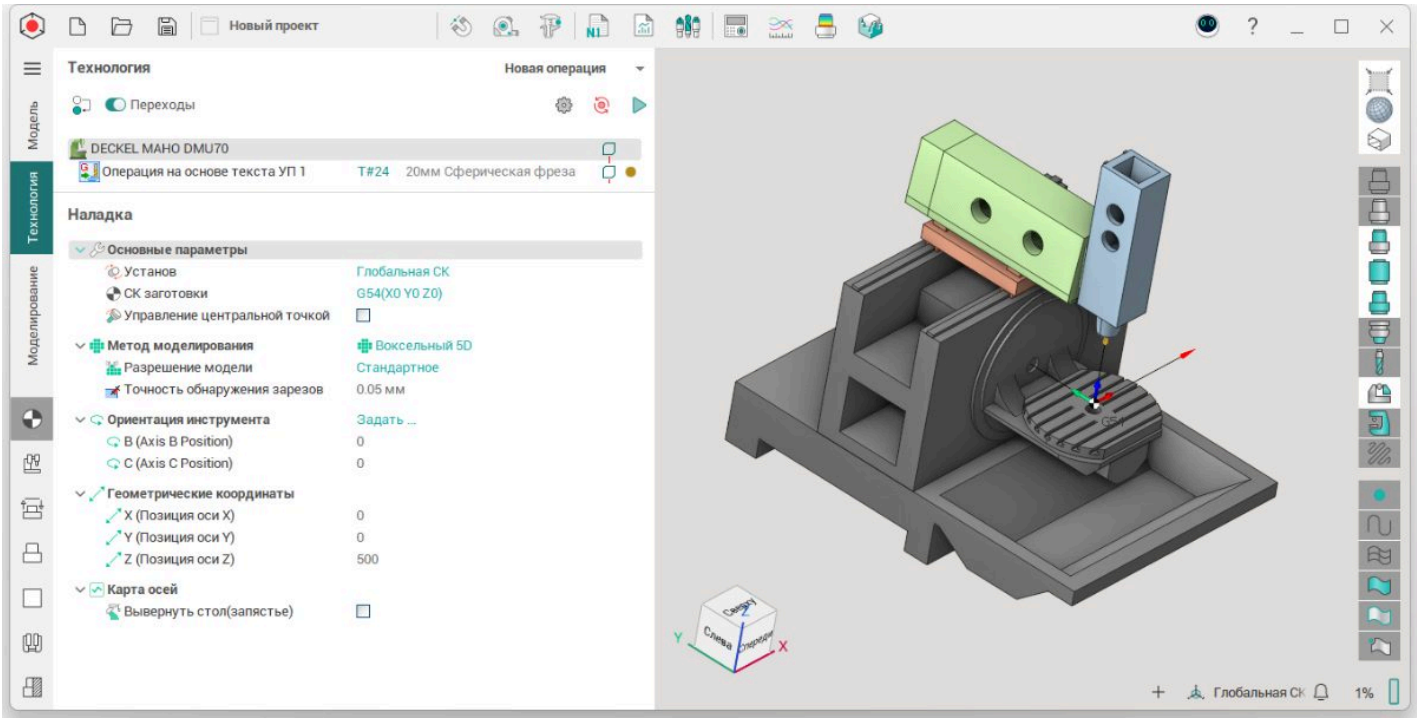
> ⊗ ВАЖНО
>
> Прежде чем запускать отладку, проверьте, что заполнил пункт Создать интерпретатор УП. Если версия пакета SDK в файле `*.csproj` предварительная (`-dev`, `-rc`) или предыдущей основной версии, поле `ID` в файле `*.snci` заполнено нулями, а у дочернего интерпретатора в `ParentID` стоит `LibraryID` родителя или `ParentFileName` пуст — исправьте эти поля. Если всё уже верно, этот шаг пропустите. Что и как проверить — в разделе Что поправить после создания.

10. Потом вы можете установить точку останова на любом методе (например, конструкторе класса NCInterpreter), переключиться на вкладку "Run and Debug" и нажать на кнопку "Launch CAM" (клавиша F5; для NC Тюнера выберите "Launch Tuner").



11. В результате запустится сборка интерпретатора из исходников, запустится СПРУТКАМ (при Launch Tuner — NC Тюнер) и начнётся отладка исходного кода.

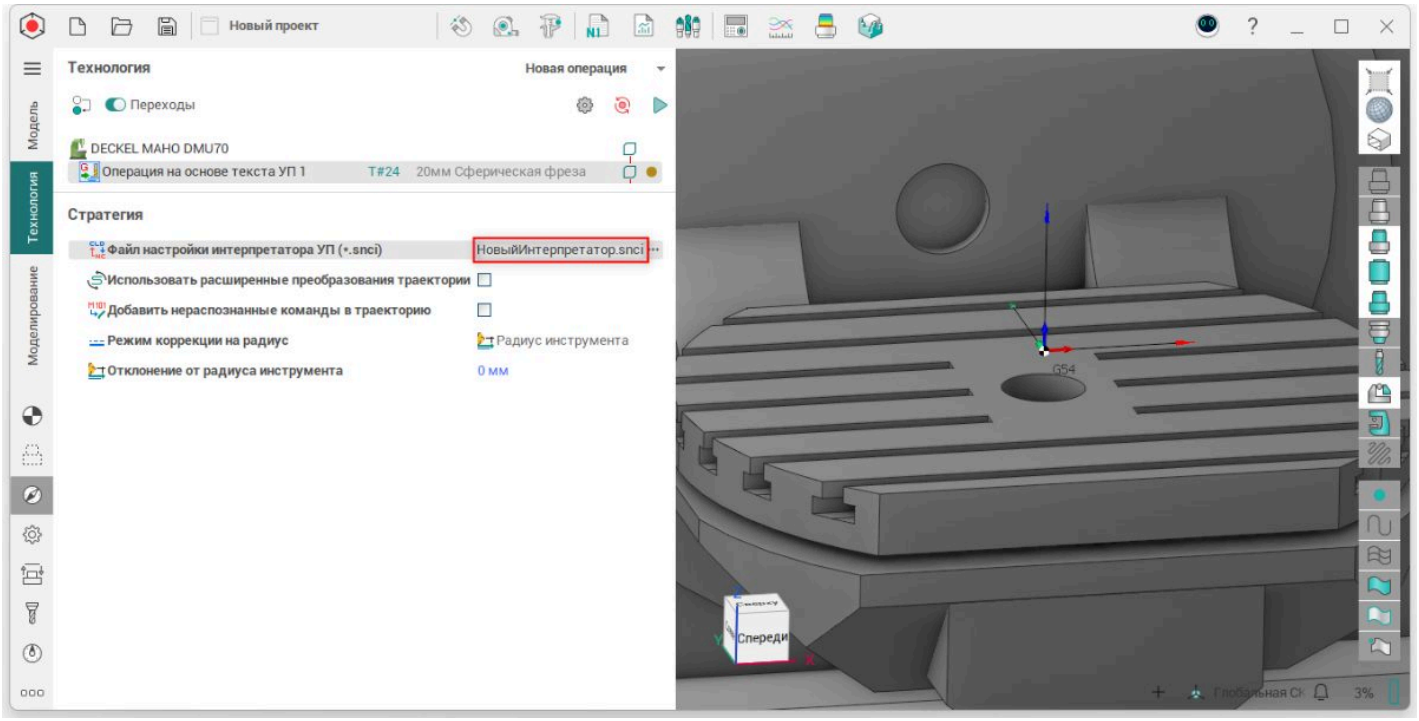
12. Для проверки работы интерпретатора необходимо создать операцию на основе текста УП в открывшемся окне СПРУТКАМ.



13. Задать какой-нибудь текст УП на панели параметров <Рабочее задание> операции на основе текста УП, например:

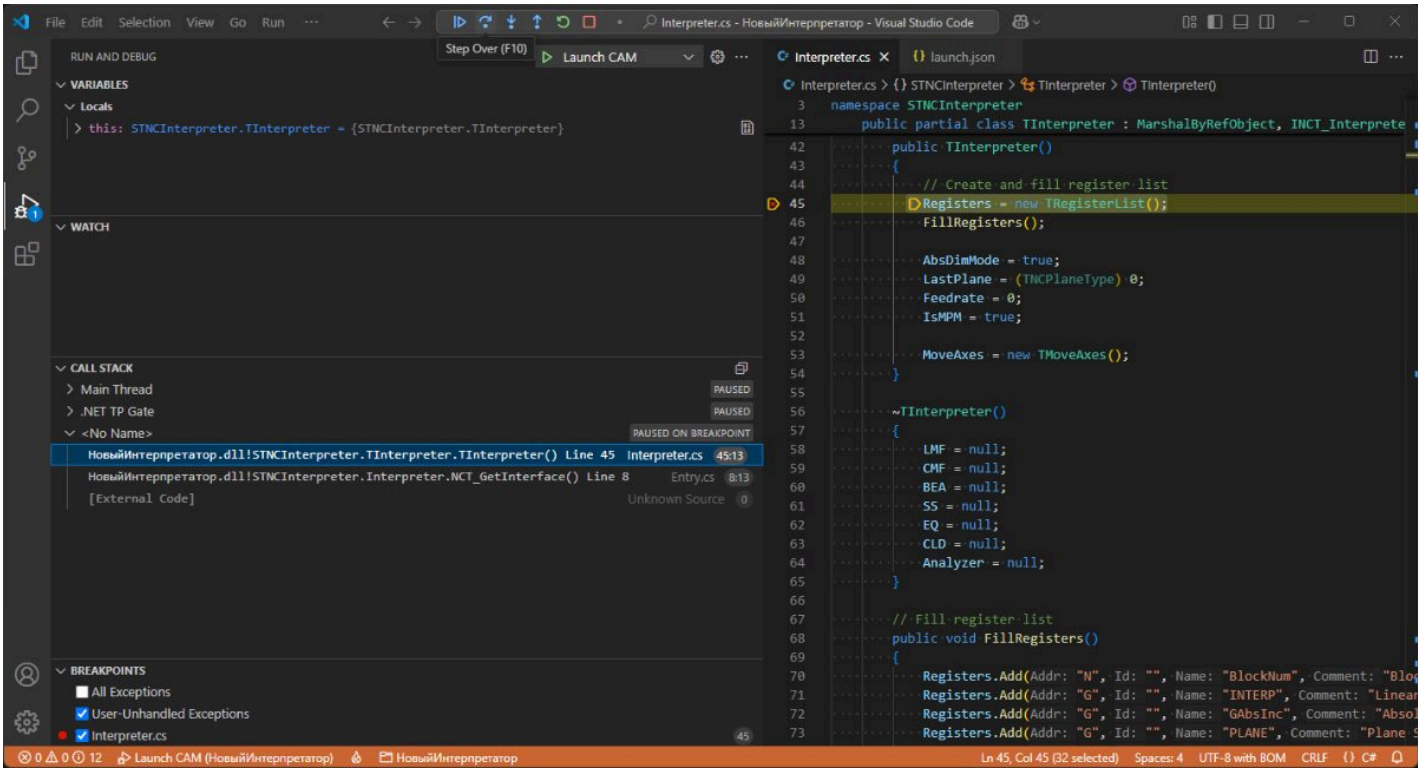
```
G0 G17 X0 Y0 Z0
X100
Y100
X0
Y0
```

14. Выбрать созданный интерпретатор на панели параметров <Стратегия>:



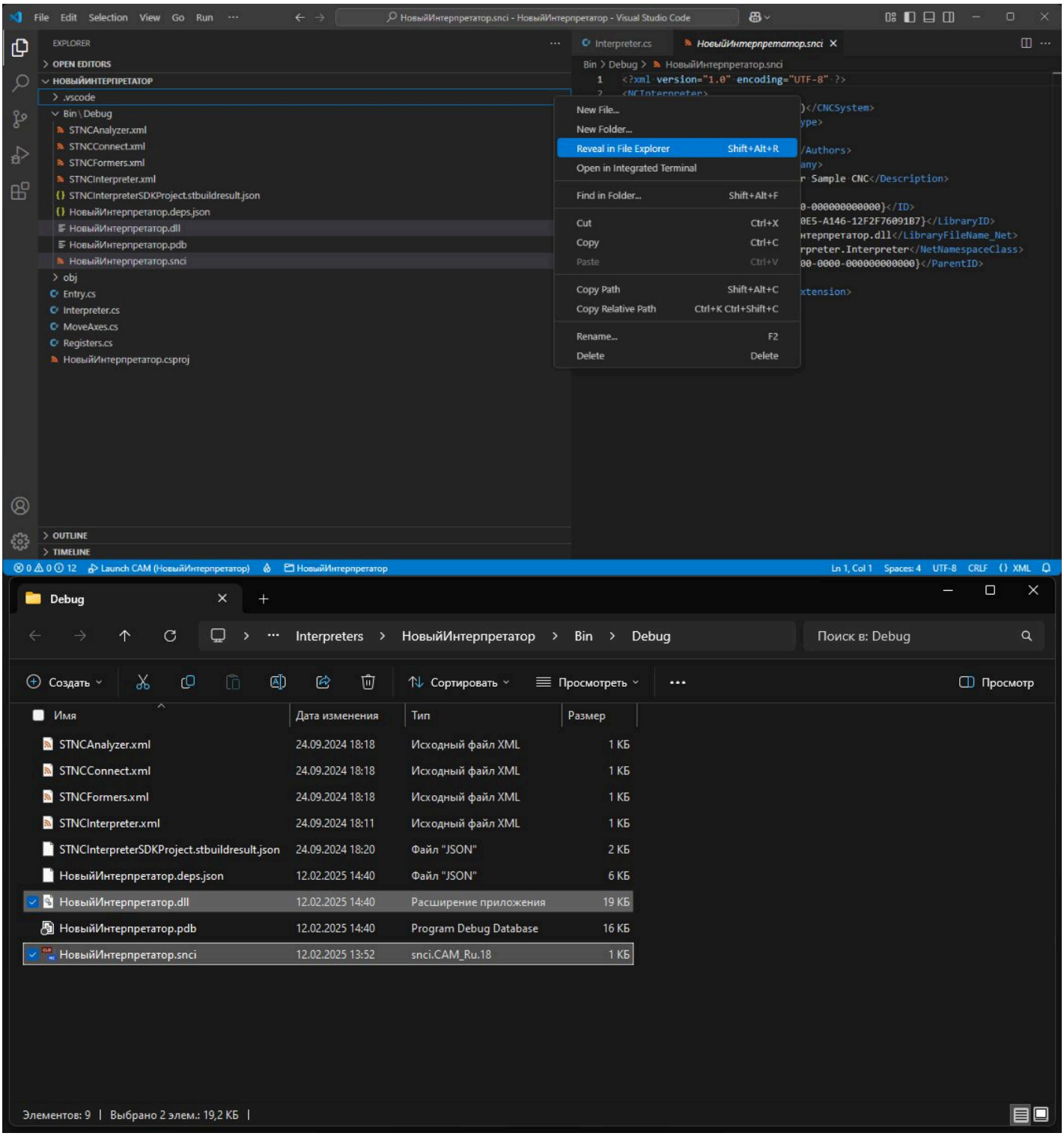
15. И нажать кнопку расчета операции 

16. Выполнение должно остановиться в заданной точке останова. Теперь вы можете использовать Шаг с обходом (F10) на панели инструментов, чтобы перемещаться по строкам кода интерпретатора.



17. Наконец, вы можете изменить код интерпретатора в файлах Interpreter.Core.cs, Interpreter.Registers.cs и НовыйИнтерпретатор.snci по мере необходимости и отладить его, используя все доступные функции кода Visual Studio Code. Если вы еще не знакомы с возможностями VS Code, рекомендуем вам посетить раздел его официальной документации «Начало работы» (https://code.visualstudio.com/docs/getstarted/introvideos).

18. После того как вы написали и отладили исходный код интерпретатора, вы можете получить и использовать файлы динамически подключаемой библиотеки (*.dll) и настройки интерпретатора (*.snci). Просто выберите «Reveal in File Explorer» в контекстном меню соответствующих файлов (например, НовыйИнтерпретатор.dll и НовыйИнтерпретатор.snci в данном случае) проводника Visual Studio Code. При этом в проводнике будут выбраны указанные файлы интерпретатора, которые вы сможете скопировать и при необходимости отправить клиенту.


File Edit Selection View Go Run
НовыйИнтерпретатор.snci - НовыйИнтерпретатор - Visual Studio Code
EXPLORER
OPEN EDITORS
НОВЫЙИНТЕРПЕТАТОР
.vscode
Bin \ Debug
STNCAnalyzer.xml
STNCConnect.xml
STNCFormers.xml
STNCInterpreter.xml
STNCInterpreterSDKProject.stbuildresult.json
НовыйИнтерпретатор.deps.json
НовыйИнтерпретатор.dll
НовыйИнтерпретатор.pdb
НовыйИнтерпретатор.snci
obj
Entry.cs
Interpreter.cs
MoveAxes.cs
Registers.cs
НовыйИнтерпретатор.csproj
OUTLINE
TIMELINE
Interpreter.cs
НовыйИнтерпретатор.snci
Bin > Debug > НовыйИнтерпретатор.snci
New File...
New Folder...
Reveal in File Explorer Shift+Alt+R
Open in Integrated Terminal
Find in Folder... Shift+Alt+F
Cut Ctrl+X
Copy Ctrl+C
Paste Ctrl+V
Copy Path Shift+Alt+C
Copy Relative Path Ctrl+K Ctrl+Shift+C
Rename... F2
Delete Delete
Launch CAM (НовыйИнтерпретатор)
НовыйИнтерпретатор
Ln 1, Col 1 Spaces: 4 UTF-8 CRLF XML
Debug
Interpreters > НовыйИнтерпретатор > Bin > Debug
Поиск в: Debug
Создать
Сортировать
Просмотреть
Просмотр
Имя
Дата изменения
Тип
Размер
STNCAnalyzer.xml 24.09.2024 18:18 Исходный файл XML 1 КБ
STNCConnect.xml 24.09.2024 18:18 Исходный файл XML 1 КБ
STNCFormers.xml 24.09.2024 18:18 Исходный файл XML 1 КБ
STNCInterpreter.xml 24.09.2024 18:11 Исходный файл XML 1 КБ
STNCInterpreterSDKProject.stbuildresult.json 24.09.2024 18:20 Файл "JSON" 2 КБ
НовыйИнтерпретатор.deps.json 12.02.2025 14:40 Файл "JSON" 6 КБ
НовыйИнтерпретатор.dll 12.02.2025 14:40 Расширение приложения 19 КБ
НовыйИнтерпретатор.pdb 12.02.2025 14:40 Program Debug Database 16 КБ
НовыйИнтерпретатор.snci 12.02.2025 13:52 snci.CAM_Ru.18 1 КБ
Элементов: 9 | Выбрано 2 элем.: 19,2 КБ

Интерпретация текста управляющей программы

Под интерпретацией будем понимать процесс преобразования текста управляющей программы, её слов и кадров, в технологические команды CLDATA, формирующие траекторию движения инструмента. В процессе интерпретации CAM система, используя выбранный для станка интерпретатор:

- последовательно считывает кадры управляющей программы;
- используя полученную от интерпретатора информацию о регистрах, выделяет из кадра управляющей программы слова (или смысловые конструкции) либо предлагает интерпретатору выделить слова самостоятельно;
- передаёт слова интерпретатору в порядке их расположения в кадре.

Интерпретатор, учитывая особенности станка, преобразует слова управляющей программы в технологические команды CLDATA, которые образуют восстановленную траекторию движения инструмента.

Описание интерпретатора приведено в разделе Структура интерпретатора.

Взаимодействие CAM системы с интерпретатором для получения вторичной (восстановленной) траектории инструмента описано в разделе Процесс интерпретации текста управляющей программы.

Структура интерпретатора

Под интерпретатором текста управляющей программы мы будем понимать файл с настройками и программную библиотеку, на которую ссылается один из параметров файла настроек.

Файл настроек интерпретатора

Файл с настройками имеет расширение SNCI (System Numerical Control Interpreter) и формат XML. Описание его элементов приведено ниже. Обязательные к заполнению параметры отмечены признаком в соответствующей колонке.

Значения собственных параметров интерпретатора хранятся не в файле SNCI, а в отдельном файле `*.sncix`, который ссылается на интерпретатор; как его создать и где выбрать — в разделе о типах для интерпретатора с собственными параметрами на странице Список используемых типов.

Описание элементов SNCI файла

Название элемента	Формат поля	Описание	Обязательные
NCInterpreter	Тег	Корневой элемент	*
CNCSystem	Строка	Название стойки/ станка	
MachineType	Строка	Тип станка, варианты: • Lathe — токарный • Mill — фрезерный • TurnMill — токарно-фрезерный • Robots — робот • WireEDM — электроэрозионный	
UseGeomCLData	Логическое	Устанавливает признак использования интерфейса построения траектории движения инструмента в геометрических осях.	

Название элемента	Формат поля	Описание	Обязательные
		True — для формирования траектории движения инструмента из CAM системы в интерпретатор передаётся интерфейс INCT_GeomCLData. False — для формирования траектории движения инструмента из CAM системы в интерпретатор передаётся интерфейс INCT_CLData. По умолчанию установлено значение False.	
Date	Дата	Дата интерпретатора в формате dd.mm.yyyy	
Authors	Строка	Разработчик(-и)	
Company	Строка	Компания	
Description	Строка	Описание	
Version	Число	Версия	*
ID	GUID	Идентификатор интерпретатора (должен быть уникальным)	*

Название элемента	Формат поля	Описание	Обязательные
LibraryID	GUID	Идентификатор программной библиотеки (должен быть уникальным)	*
LibraryFileName_Net	Строка	Ссылка на программную библиотеку для платформы .NET	**
NetNamespaceClass	Строка	Пространство имён и имя класса, реализующего интерпретатор (если заполнено поле LibraryFileName_Net)	***
LibraryFileName_x64	Строка	Ссылка на нативную программную библиотеку версии x64 (используется если не заполнено поле LibraryFileName_Net)	**
LibraryFileName_x32	Строка	Ссылка на нативную программную библиотеку версии x32 (нужна только для 32-разрядного продукта старой версии)	
ParentID	GUID	Идентификатор интерпретатора-родителя (поле ID в SNCI-файле родителя)	*

Название элемента	Формат поля	Описание	Обязательные
ParentFileName	Строка	Ссылка на файл настроек интерпретатора-родителя (SNCI-файл)	*
FileExtension	Строка	Расширение файлов с текстом управляющей программы	*
Registers	Тег	Список регистров интерпретатора	*

* — обязательное поле;

** — должно быть заполнено либо поле LibraryFileName_Net, либо LibraryFileName_x64;

*** — поле должно быть заполнено, если заполнено поле LibraryFileName_Net.

Описание каждого элемента из списка регистров (Registers):

Название элемента	Формат поля	Описание
Address	Строка	Адрес регистра
Name	Строка	Имя регистра
Id	Строка	Идентификатор регистра
Comment	Строка	Комментарий

Название элемента	Формат поля	Описание
AddrOnly	Строка	Варианты значений: • aoValue — Слово текста УП состоит только из адреса и значения • aoAddr — Слово текста УП состоит только из адреса • aoBoth — Слово текста УП может состоять либо только из адреса, либо из адреса и значения
Sign	Целое число	Знак значения: • 0 — без знака • 1 — только '-' • 2 — '+' и '-'
NumeralsBeforePoint.Count	Целое число	Знаков до точки
NumeralsBeforePoint.Zeros	Логическое	Наличие лидирующих нулей
DecimalPoint	Целое число	Десятичная точка: • 0 — без '.' • 1 — '.' • 2 — всегда '.'
NumeralsAfterPoint.Count	Целое число	Знаков после точки
NumeralsAfterPoint.Zeros	Логическое	Незначащие нули
Scale	Число	Масштаб
Modal	Логическое	Признак модальности регистра
ModalGroupList	Список значений	Список значений, которые может принимать регистр

Тег Version заполняется, как в примере ниже, но его значение CAM система не использует (зарезервировано для будущего использования).

Важными элементами файла настроек интерпретатора являются: его собственный идентификатор (ID), идентификатор (LibraryID) и ссылка на файлы программной библиотеки .NET/x64 версии (LibraryFileName_*), идентификатор (ParentID) и ссылка на интерпретатор родителя (ParentFileName). Идентификаторы файла настроек и программной библиотеки должны быть уникальными (см. статью о GUID: Globally Unique Identifier).

Поле LibraryFileName_* должно указывать на файлы, с программной библиотекой, относительно каталога размещения SNCI-файла. Рекомендуется размещать файл с программной библиотекой в каталоге с SNCI-файлом.

Существует два варианта подключения программной библиотеки к CAM системе:

1. Программная библиотека разработана для платформы .NET (например, на C#).
   В этом случае необходимо заполнить поля LibraryFileName_Net и NetNamespaceClass, где:

   - LibraryFileName_Net - ссылка на программную библиотеку.
   - NetNamespaceClass - пространство имён и класс, реализующий интерфейс взаимодействия с интерпретатором. Формат: Namespace.Class, например: Sample.Interpreter.
2. Программная библиотека является нативной (написана на Delphi, C++ и пр.)

   В этом случае должно быть заполнено поле LibraryFileName_x64 (ссылка на программную библиотеку x64). Версию x64 программной библиотеки обязательно следует называть, используя окончание "64". Например: Sample_x64.dll или Sample_Win64.dll. Где "Sample" — название программной библиотеки.

Если интерпретатор не является наследником другого интерпретатора, то поля родителя заполняются пустыми значениями:

<ParentID>{00000000-0000-0000-0000-000000000000}</ParentID>

<ParentFileName/>

Тег FileExtension заполняется, как в примере ниже, но его значение CAM система не использует (зарезервировано для будущего использования).

В поле Registers находится список регистров для автоматического выделения слов из текста управляющей программы. Порядок трансляции кадра управляющей программы описан в следующей статье. В файле настроек интерпретатора допускается не заполнять список регистров, он будет получен от программной библиотеки интерпретатора, тем не менее тег <Registers/> должен присутствовать.

Использование файла настроек интерпретатора приведено в разделе Процесс интерпретации текста управляющей программы.

Пример файла настроек интерпретатора для нативной библиотеки (x64):

```
<?xml version="1.0" encoding="UTF-8" ?>
<NCInterpreter>
 <CNCSystem>Sample CNC (mill)</CNCSystem>
 <MachineType>Mill</MachineType>
 <Date>01.01.2019</Date>
 <Authors>Ivan Danko</Authors>
 <Company>SPRUT Technology</Company>
 <Description>Interpreter for Sample CNC</Description>
 <Version>1</Version>
 <ID>{79B1410C-67D5-42B6-867B-A3BF30AB3D8A}</ID>
 <LibraryID>{80F754D9-EE7B-47CF-A802-A9424D9F6749}</LibraryID>
 <LibraryFileName_x64>Sample_x64.dll</LibraryFileName_x64>
 <ParentID>{00000000-0000-0000-0000-000000000000}</ParentID>
 <ParentFileName/>
 <FileExtension>*.cnc</FileExtension>
 <Registers/>
</NCInterpreter>
```

Скачать пример Sample.snci.

> ПРИМЕЧАНИЕ
>
> Это образец формата файла: библиотеки Sample_x64.dll к нему нет. Поля ID и LibraryID в нём совпадают с файлом sample.snci интерпретатора `sample` из раздела Примеры интерпретаторов с исходными кодами, а интерпретатор `sample_inheritance` ищет родителя по имени файла sample.snci и его ID. Не кладите образец туда, где CAM система ищет интерпретаторы, а если делаете из него свой интерпретатор, впишите в ID и LibraryID новые GUID.

Программная библиотека интерпретатора

Программная библиотека интерпретатора представляет собой библиотеку динамической компоновки (DLL). Вне зависимости от платформы, для которой реализована библиотека (.NET/нативная), библиотека должна экспортировать две функции:

NCT_GetInterface — возвращает в CAM систему объект, реализующий интерфейс для взаимодействия ядра CAM системы с программной библиотекой (INCT_Interpreter);

NCT_FreeMemory — освобождает память от всех объектов, созданных самостоятельно или полученных от CAM системы в процессе работы.

Пример класса, описывающего функции библиотеки интерпретатора на языке C# (сборка для платформы .NET):

```
namespace sample
{
    public class Interpreter: MarshalByRefObject
    {
        public static INCT_Interpreter NCT_GetInterface()
        {
            /* реализация */
        }
        public static void NCT_FreeMemory()
        {
            /* реализация */
        }
    }
}
```

В сборке .NET эти функции могут быть как статическими, так и методами экземпляра: CAM система создаёт объект класса, заданного тегом NetNamespaceClass файла настроек интерпретатора, вызывает у него NCT_GetInterface, а при выгрузке библиотеки - NCT_FreeMemory. Поэтому у класса должен быть открытый конструктор без параметров.

Описание функций нативной библиотеки интерпретатора на языке Delphi:

```
function NCT_GetInterface: INCT_Interpreter; stdcall;
procedure NCT_FreeMemory; stdcall;
```

Описание функций нативной библиотеки интерпретатора на языке C++:

```
void __stdcall NCT_GetInterface(INCT_Interpreter** Result);
void __stdcall NCT_FreeMemory();
```

Функция NCT_GetInterface объявлена в CAM системе как функция Delphi, возвращающая интерфейс. Такой результат передаётся не через регистр, а скрытым параметром: CAM система передаёт адрес, по которому библиотека должна записать указатель на объект INCT_Interpreter.

Все типы, необходимые для взаимодействия CAM системы с программной библиотекой интерпретатора, приведены в библиотеках типов. Они распространяются пакетами NuGet из публичного источника пакетов производителя `https://nexus.sprut.ru/repository/master/index.json` (см. Версии продукта); берите пакеты той же основной версии, что пакет SDK, с которым собран продукт: для C++ и других языков — пакет `SprutTechnology.NCInterpreterSDK.tlb` (файлы `*.tlb`, `*.h` и `*.idl`), для Delphi — `SprutTechnology.NCInterpreterSDK.bpl.x64` (модули `IDL.<Имя>.pas`). Оба пакета подтягивают по пакету на каждую библиотеку типов.

Следующие TLB-файлы содержат описание типов, которые используются при взаимодействии с библиотекой интерпретатора:

- STNCInterpreter.tlb — основной перечень типов;
- STNCAnalyzer.tlb — типы, используемые для управления анализом текста управляющей программы;
- STNCFormers.tlb — типы, используемые для упрощения построения траектории движения инструмента (линейные/круговые перемещения, циклы);
- STNCConnect.tlb — типы, используемые для защиты интерпретатора (см. Интерпретаторы с защитой);
- STNCCustomInterpreter.tlb — типы для интерпретатора с собственными параметрами (см. Список используемых типов).

Для подключения библиотеки типов к вашей среде разработки, обратитесь к соответствующему разделу документации среды разработки.

ⓘ ПРИМЕЧАНИЕ

Регистрация библиотеки типов

Для регистрации библиотеки типов в Microsoft Visual Studio попробуйте следующее:

Найдите файл tlbImp.exe на вашем компьютере. Выполните команду "tlbimp.exe tlb-filename". В результате выполнения команды будет создан файл "tlb-filename.dll" (например, "STNCAnalyzer.dll"). Добавьте этот dll файл в ваш проект через пункт меню "Add Reference".

На странице Примеры интерпретаторов с исходными кодами приведён пример исходных кодов библиотеки интерпретатора, собранной из них 'dll', и пример файла настроек.

Примеры интерпретаторов с исходными кодами

Проекты примеров созданы в Microsoft Visual Studio для .NET Framework 4 и ссылаются на сборки взаимодействия с библиотеками типов STNCInterpreter, STNCAnalyzer и STNCFormers: в `sample_dc` — на файлы `..\tlb\STNC*.dll`, в остальных — ссылками COM на зарегистрированные библиотеки типов. В архивы эти сборки не входят: файлы `*.tlb` берутся из пакета NuGet (см. Программная библиотека интерпретатора), а сборки взаимодействия создаются из них утилитой `tlbimp`. Новый интерпретатор на C# удобнее создавать из шаблона CAM системы, см. Разработка собственного интерпретатора на (.NET) C#.

1. Интерпретаторы для моделирования и импорта траектории (на основе машинной CLData)

1.1. Интерпретатор `sample`.

Реализует линейные и круговые перемещения и команду изменения подачи для стойки Fanuc 30i (фрезерной).

Исходные коды библиотеки интерпретатора, разработанной для Microsoft Visual Studio (C#), расположены по этой ссылке.

Интерпретатор на основе откомпилированной библиотеки (sample.dll) и файл настроек для неё (sample.snci), можно получить по этой ссылке.

1.2. Интерпретатор с наследованием `sample_inheritance`.

Наследник интерпретатора sample. В дополнение к командам родителя, реализует несколько основных команд: M, S, T, G04 X P (выстой).

Исходные коды библиотеки интерпретатора-наследника, разработанной для Microsoft Visual Studio (C#), расположены по этой ссылке.

Интерпретатор-наследник, на основе откомпилированной библиотеки (sample_inheritance.dll) и файл настроек для неё (sample_inheritance.snci), можно получить по этой ссылке.

1.3. Интерпретатор с наследованием `sample_Fanuc30i_Mill`.

Наследник интерпретатора Fanuc30i_Mill (входит в дистрибутив CAM системы). В дополнение к командам родителя, реализует команду G04 X P (выстой).

Исходные коды библиотеки интерпретатора-наследника, разработанной для Microsoft Visual Studio (C#), расположены по этой ссылке.

Интерпретатор-наследник на основе откомпилированной библиотеки (sample_Fanuc30i_Mill.dll) и файл настроек для неё (sample_Fanuc30i_Mill.snci), можно получить по этой ссылке.

1.4. Интерпретатор с использованием формирователя циклов сверления `sample_dc` .

Реализует команду CYCLE81 для фрезерной стойки Sinumerik 840D.

Исходные коды библиотеки интерпретатора, разработанной для Microsoft Visual Studio (C#), расположены по этой ссылке.

Интерпретатор на основе откомпилированной библиотеки (sample_dc.dll) и файл настроек для неё (sample_dc.snci), можно получить по этой ссылке.

Для демонстрации работы интерпретатора предназначен следующий проект. Необходимо скачать проект и интерпретатор. Открыть проект, в настройках операции на основе текста УП задать интерпретатор sample_dc.snci, выполнить расчёт операции, смоделировать.

2. Интерпретаторы для импорта траектории (на основе геометрической CLData)

2.1. Интерпретатор `sample_geom`.

Реализует линейные и круговые перемещения для 3-х осевой фрезерной обработки и команду изменения подачи для стойки Fanuc 30i.

Исходные коды библиотеки интерпретатора, разработанной для Microsoft Visual Studio (C#), расположены по этой ссылке.

Интерпретатор, на основе откомпилированной библиотеки (sample_geom.dll) и файл настроек для неё (sample_geom.snci), можно получить по этой ссылке.

2.2. Интерпретатор `sample_geom_5x`.

Реализует линейные и круговые перемещения для 5-и осевой фрезерной обработки (команды: X, Y, Z, TRAORI, TRAFOOF, A3, B3, C3) и команду изменения подачи для стойки Sinumerik.

Исходные коды библиотеки интерпретатора, разработанной для Microsoft Visual Studio (C#), расположены по этой ссылке.

Интерпретатор, на основе откомпилированной библиотеки (sample_geom_5x.dll) и файл настроек для неё (sample_geom_5x.snci), можно получить по этой ссылке.

Для демонстрации 5-и осевых перемещений предназначен следующий проект. Необходимо скачать проект и интерпретатор. Открыть проект, в настройках операции на основе текста УП задать интерпретатор sample_geom_5x.snci, выполнить расчёт операции, смоделировать.

Интерпретаторы с защитой

Для обеспечения использования интерпретатора в соответствии с выданной лицензией, необходимо выполнить следующие действия:

1. Интерпретатор должен быть разработан с использованием интерфейса защиты.
2. Интерпретатор должен быть помещён в контейнер.
3. Вместе с интерпретатором в контейнер должен быть помещён файл или файлы с секретными данными.

После выдачи лицензии на контейнер и её активации, пользователю станет доступно использование интерпретатора, см. Моделирование обработки по управляющей программе.

Процесс интерпретации текста управляющей программы

Интерпретация текста управляющей программы, сформированной по операциям, состоит из следующих этапов:

1. Перед интерпретацией текста управляющей программы первой операции: ядро CAM системы загружает интерпретатор в память, производит инициализацию интерпретатора.
2. Для каждой операции: производится трансляция и интерпретация текста управляющей программы.
3. После интерпретации последней операции: CAM система выгружает интерпретатор из памяти.

Особенности использования уже разработанных интерпретаторов для интерпретации текста управляющей программы других, похожих, стоек описаны в статье Наследование интерпретатора.

Список используемых и реализуемых в интерпретаторе интерфейсов, приведён в разделе Список используемых типов.

Управление протоколированием процесса интерпретации текста управляющей программы описано в разделе Журнал взаимодействия с интерпретатором.

Загрузка и инициализация интерпретатора

В CAM системе процесс загрузки интерпретатора в память состоит из следующих шагов:

- Чтение файла настроек интерпретатора.
- Загрузка списка регистров из файла настроек интерпретатора.
- Загрузка интерпретатора-родителя в память (процесс загрузки интерпретатора-родителя повторяет все шаги загрузки интерпретатора в память в той же последовательности).
- Загрузка программной библиотеки интерпретатора в память, получение интерфейса INCT_Interpreter (путём вызова функции NCT_GetInterface загруженной программной библиотеки).
- Проверка совпадения идентификаторов файла настроек и программной библиотеки (получение идентификатора программной библиотеки производится путём вызова функции INCT_Interpreter.GetIdentifier), и идентификаторов интерпретатора-родителя.
- Передача значений собственных параметров, если объект интерпретатора реализует интерфейс INCT_CustomInterpreter (вызов метода INCT_CustomInterpreter.SetCustomValues, значения берутся из файла `*.sncix`; см. Список используемых типов). Шаг выполняется, только когда интерпретатор загружается для работы, а не для проверки и не для чтения его параметров в окне Конфигуратор интерпретатора УП.
- Загрузка списка регистров из программной библиотеки (вызов функции INCT_Interpreter.GetRegisters).
- Загрузка списка регистров интерпретаторов-родителей.

В результате выполнения загрузки интерпретатора будет загружена программная библиотека интерпретатора и её библиотеки-родители (если они есть), сформирован результирующий список регистров (базовый). При этом регистры файла настроек имеют больший приоритет перед регистрами программной библиотеки (т.е. переопределяют их), а регистры интерпретаторов-наследников имеют больший приоритет перед регистрами интерпретаторов-родителей, если регистр с таким именем присутствует в обоих списках.

Инициализация интерпретатора состоит из следующих шагов:

- Инициализация интерпретаторов-родителей (процесс инициализации интерпретатора-родителя повторяет все шаги инициализации интерпретатора в той же последовательности).
- Последовательная передача интерфейсов CAM системой программной библиотеке (метод программной библиотеки интерпретатора INCT_Interpreter.SetInterface будет вызван несколько раз, по количеству передаваемых интерфейсов).
- Инициализация программной библиотеки (вызов метода INCT_Interpreter.Initialize).

В результате инициализации интерпретатору будут переданы объекты CAM системы, реализующие интерфейсы:

- доступа к сформированному (базовому) списку регистров INCT_SysState;
- управления настройками анализа текста управляющей программы INCT_Analyzer;
- доступа к списку объектов для интерпретации INCT_BlockExecArray;
- получения настроек оборудования INCT_Equipment;
- формирования траектории движения инструментов с использованием осей станка (если в файле настроек интерпретатора `UseGeomCLData` = False) INCT_CLData;
- формирования траектории движения инструментов с использованием геометрических осей (если `UseGeomCLData` = True) INCT_GeomCLData;
- взаимодействия с ядром CAM системы, обслуживающим интерпретатор INCT_Kernel;
- интерпретатора-родителя INCT_Interpreter (если он есть);
- контура (интерфейс, переданный в INCT_Kernel.InterpreteContour), если интерпретатор запущен в режиме формирования контура;
- защиты INCT_Connect, только для защищённых интерпретаторов.

Будет выполнен метод инициализации библиотеки интерпретатора.

Трансляция и интерпретация текста управляющей программы

Интерпретация текста управляющей программы.

До тех пор, пока кадры управляющей программы не закончатся или библиотека интерпретатора не выставит признак окончания программы (метод INCT_Interpreter.EndOfProgram), CAM система будет считывать кадры управляющей программы и для каждого кадра выполнять следующие шаги:

- Трансляцию кадра управляющей программы (анализ кадра на предмет знакомых конструкций - объектов для интерпретации);
- Если трансляция кадра привела к появлению хоть одного объекта в списке объектов текущего кадра для интерпретации (т.е. список объектов для интерпретации INCT_BlockExecArray не пуст), выполняется последовательная интерпретация каждого объекта из списка INCT_BlockExecArray;
- Список объектов текущего кадра для интерпретации очищается.

Трансляция кадра управляющей программы.

До тех пор, пока символы кадра не закончились, CAM система выполняет следующие шаги:

- Передача библиотеке интерпретатора необработанной части кадра (то есть, на первом шаге будет передан весь кадр). В ответ CAM система ожидает получить количество символов с начала переданной строки, которые интерпретатор распознал как отдельное слово или другую понятную интерпретатору конструкцию (метод INCT_Interpreter.TransLine). Распознанные символы признаются обработанными и удаляются из кадра. Этот шаг будет повторяться, пока интерпретатор распознаёт смысловые конструкции (слова) в начале оставшегося кадра, т.е. возвращает в CAM систему количество распознанных символов больше нуля, или кадр не закончится.
- Как только интерпретатор не смог распознать смысловую конструкцию с начала оставшегося кадра, CAM система пытается определить регистр и его значение по описанному в этой статье алгоритму.
- Если регистр был определён, он будет добавлен в конец списка объектов для интерпретации текущего кадра (INCT_BlockExecArray, с признаком регистр). В ином случае считанная и нераспознанная часть кадра будет пропущена, но она так же будет считаться обработанной и удалена из начала оставшегося кадра;

Примечание.

Для распознавания смысловых конструкций кадра, там, где это возможно, рекомендуем использовать механизм определения регистров и значений CAM системы (Алгоритм

определения регистра и значения в кадре текста управляющей программы). А в интерпретаторе (в методе INCT_Interpreter.TransLine) обрабатывать только те конструкции, которые невозможно или неэффективно распознавать механизмом CAM системы. В этом случае рутинная работа по определению типовых конструкций будет выполнена ядром CAM системы, в интерпретаторе будут обрабатываться редкие случаи смысловых конструкций, не укладывающихся в общие правила алгоритма.

Примечание.

После того как интерпретатор распознал слово или смысловую конструкцию, он может самостоятельно запомнить информацию об этом (используя свои внутренние структуры данных) или воспользоваться списком объектов для интерпретации текущего кадра (INCT_BlockExecArray). Т.е. добавить в список свой элемент с признаком объект или регистр. Рекомендуем воспользоваться вторым вариантом, в этом случае будет сохранён порядок следования объектов в кадре, вне зависимости от того каким образом объект был определён: ядром CAM системы или интерпретатором.

Интерпретация объектов списка для интерпретации состоит из следующих шагов:

- Перед интерпретацией списка объектов вызывается метод INCT_Interpreter.BeforeInterprete, он информирует интерпретатор о начале процесса интерпретации текущего блока;
- Для каждого элемента списка, если у него не установлен признак IsUsed и не установлен признак окончания трансляции (INCT_Interpreter.EndOfProgram), выполняется метод INCT_Interpreter.Interprete. Затем сынтерпретированному элементу списка устанавливается признак IsUsed;
- После интерпретации всего списка объектов вызывается метод INCT_Interpreter.AfterInterprete, он информирует интерпретатор о завершении процесса интерпретации текущего блока;

Примечание.

При выполнении метода INCT_Interpreter.Interprete, по переданному объекту интерпретатор должен определить его тип и добавить в список команд CLDATA соответствующую команду.

Например, если переданный для стойки Fanuc объект является регистром с адресом F (подача), то нужно выполнить метод INCT_CLData.OutFeed, в качестве параметра передать значение регистра.

Алгоритм определения регистра и значения в кадре текста управляющей программы

На этапе распознавания кадра текста управляющей программы ядро CAM системы оперирует лексемами - последовательностями допустимых символов языка, имеющих смысл для транслятора.

Алгоритм определения регистра и значения:

1. Для определения регистра ядро считывает лексему из строки управляющей программы (порядок чтения лексемы приведён ниже).
2. Если лексема не является регистром, то работа алгоритма завершается. Полученная из текста управляющей программы лексема будет считаться обработанной и нераспознанной.
3. Если лексема является регистром без значения (т.е. функция INCT_Register.Get_ValueAvail регистра возвращает vatOnlyAddr), то работа алгоритма завершается. Полученная из текста управляющей программы лексема будет считаться обработанной, регистр определён.
4. В ином случае ядро считывает лексемы последовательно, ожидая встретить число (со знаком или без) или символ '=' (если он разрешён настройкой INCT_Analyzer.CanUseEqualToAssignRegister), после которого ожидается число или строка в кавычках. Скобки (настройка INCT_Analyzer.Brackets) при определении значения регистра не используются. Чтение лексем производится до тех пор, пока ядро не определит значение регистра или встретит некорректную лексему. В обоих случаях работа алгоритма завершается, лексемы будут считаться обработанными. В первом случае успешно — регистр найден, значение определено. Во втором — неуспешно, лексемы считаются нераспознанными.

Порядок чтения лексемы:

- Из текста управляющей программы считывается первый символ строки;
- Если символ является числом или точкой, считается, что вычитываемая лексема - число, далее из текста управляющей программы считываются символы, до тех пор, пока они являются числами. Чтение лексемы закончено успешно;
- Если символ является одним из элементов списка: '+', '-', '=', ',' (а при включённой настройке INCT_Analyzer.CanUseEqualToAssignRegister - ещё и одной из скобок, заданных в настройке INCT_Analyzer.Brackets), считается, что лексема состоит из одного символа. Чтение лексемы закончено успешно.
- Если символ является кавычкой (т.е. включен в список INCT_Analyzer.Quotes), то ядро ищет вторую кавычку в текущем кадре управляющей программы. При её нахождении, считает лексему строкой, чтение которой закончено успешно. Если вторая кавычка не найдена,

весь текст до конца кадра считается обработанным, а чтение лексемы признаётся неуспешным.

- Если символ является первым символом комментария (т.е. включен в список INCT_Analyzer.Comments), то ядро ищет закрывающий символ комментария. При его нахождении, считает лексему комментарием, чтение которого закончено успешно. Если закрывающий символ не найден, весь текст до конца кадра считается обработанным и нераспознанным, а чтение лексемы признаётся неуспешным.
- Если символ является строчной или прописной буквой латинского алфавита, то ядро попытается определить регистр. Ядро считывает подряд идущие буквы. Если в сформированном (базовом) списке регистров есть составные адреса (например, 'TOOL CALL', 'DL+', 'DR2'), ядро продолжает считывать следующие части кадра (буквы, числа, '+' или '-', пробелы), пока они совпадают с частями этих адресов. Настройки INCT_Analyzer.RegistersContainSpaces, INCT_Analyzer.RegistersCanEndWithPlusOrMinus и INCT_Analyzer.RegistersCanEndWithNumber устарели и на чтение лексемы не влияют. Полученный набор символов направляется интерпретатору для определения регистра (функция INCT_Interpreter.DefineRegister). Если интерпретатор вернул отрицательный ответ (регистр не определён), ядро ищет регистр в сформированном (базовом) списке регистров по полю "Адрес". Если регистр найден в списке или на предыдущем шаге интерпретатор вернул положительный ответ (регистр найден), то лексема считается определённой — это регистр. Чтение лексемы закончилось успешно, в ином случае считанные символы считаются обработанными, а чтение лексемы признаётся неуспешным.

Примечание.

При реализации функции (INCT_Interpreter.DefineRegister) рекомендуется выполнять поиск только тех регистров, которые не могут быть определены ядром CAM системы однозначно. В остальных случаях поиск не выполнять, возвращать отрицательный результат.

Выгрузка интерпретатора из памяти

Выгрузка интерпретатора из памяти состоит из следующих шагов:

- Освобождение интерфейса INCT_Interpreter;
- Освобождение памяти от всех объектов, созданных библиотекой самостоятельно или полученных от CAM системы в процессе работы (вызов функции библиотеки NCT_FreeMemory);
- Выгрузка библиотеки интерпретатора из памяти;
- Выгрузка из памяти интерпретаторов-родителей (если они есть).

В результате выгрузки будет освобождена память от созданных объектов, из памяти будут выгружены библиотека интерпретатора и его интерпретаторы-родители.

Наследование интерпретатора

Механизм наследования предназначен для повторного использования уже разработанных интерпретаторов при трансляции и интерпретации управляющих программ с похожим, но отличающимся синтаксисом. Степень необходимого отличия управляющей программы от оригинальной для реализации наследника, а не нового интерпретатора, определяется разработчиком самостоятельно.

Предназначение интерпретатора-наследника состоит в трансляции и интерпретации различий в управляющей программе, а также передаче управления интерпретатору-родителю для трансляции и интерпретации похожих участков текста управляющей программы.

Отличия реализации интерпретатора-наследника от обычного интерпретатора, заключаются в следующем:

1. В файле настроек интерпретатора (SNCI) заполняются поля ParentID, ParentFileName.
2. Если присутствует необходимость, переопределяются существующие или добавляются новые регистры. Сделать это можно, добавив изменения в список регистров SNCI-файла или в список регистров, возвращаемых функцией INCT_Interpreter.GetRegisters программной библиотеки интерпретатора-наследника.
3. Реализуется программная библиотека интерпретатора-наследника. При реализации библиотеки, вызов функций и методов программной библиотеки интерпретатора-родителя осуществляется или не осуществляется наследником в зависимости от алгоритмов, закладываемых для трансляции и интерпретации текста управляющей программы. Интерфейс интерпретатора-родителя (INCT_Interpreter) передаётся наследнику при инициализации.

Наследование возможно только от одного родителя. Количество интерпретаторов-наследников от одного родителя и глубина наследования не имеют ограничений.

Интерпретатор с собственными параметрами (его объект реализует интерфейс INCT_CustomInterpreter, см. Список используемых типов) не может быть родителем: CAM система прекращает загрузку наследника с ошибкой.

Наследование регистров.

Регистры интерпретаторов-наследников переопределяют регистры интерпретаторов-родителей. Наследование регистров реализовано таким образом, что в результирующий список (список базовых регистров) производится:

1. Загрузка списка регистров из файла настроек интерпретатора-наследника.
2. Загрузка списка регистров из программной библиотеки интерпретатора-наследника (INCT_Interpreter.GetRegisters). Если регистр с таким именем уже существует в списке

базовых регистров, то он не загружается.
3. Загрузка списка регистров из файла настроек интерпретатора-родителя. Если регистр с таким именем уже существует в списке базовых регистров, то он не загружается.
4. Загрузка списка регистров из программной библиотеки интерпретатора-родителя. Если регистр с таким именем уже существует в списке базовых регистров, то он не загружается.
5. Загрузка списка регистров из файла настроек родителя интерпретатора-родителя. Если регистр с таким именем уже существует в списке базовых регистров, то он не загружается.
6. Загрузка списка регистров из программной библиотеки родителя интерпретатора-родителя. Если регистр с таким именем уже существует в списке базовых регистров, то он не загружается.
7. И так далее, до самого первого родителя.

Наследование программной библиотеки.

При инициализации программной библиотеки интерпретатора-родителя, ряд методов и функций вызывается ядром CAM системы и не требует вызова этих функций из наследника:

1. INCT_Interpreter.GetIdentifier.
2. INCT_Interpreter.GetRegisters.
3. INCT_Interpreter.SetInterface.
4. INCT_Interpreter.Initialize.

Вызов остальных функций и методов программной библиотеки интерпретатора-родителя осуществляется или не осуществляется наследником в зависимости от алгоритмов, закладываемых для трансляции и интерпретации текста управляющей программы. Для вызова методов родителя, при инициализации программной библиотеки интерпретатора-наследника, ему, в том числе передаётся интерфейс интерпретатора-родителя (функция INCT_Interpreter.SetInterface).

Ниже представлены рекомендации по вызову функций и методов интерпретатора-родителя из интерпретатора-наследника.

1. INCT_Interpreter.TransLine - вызывать TransLine родителя в случае, если анализ строки управляющей программы не привёл к уменьшению исходной строки. Все параметры, которые вернул метод родителя (ProcessedChars, NextLine, BreakLine) передать в выходные параметры метода наследника.
2. INCT_Interpreter.DefineRegister - вызывать DefineRegister родителя в случае, если определение регистра по адресу не произведено в наследнике. Параметр Index и результат работы, которые вернула функция родителя передать в выходной параметр и результат работы функции наследника.

3. INCT_Interpreter.BeforeInterprete - вызывать BeforeInterprete родителя всегда.
4. INCT_Interpreter.Interprete - вызывать Interprete родителя в случае, если интерпретация объекта интерпретации не производится наследником.
5. INCT_Interpreter.AfterInterprete - вызывать AfterInterprete родителя всегда.
6. INCT_Interpreter.EndOfProgram - вызывать EndOfProgram родителя всегда, если наследником не определён конец управляющей программы. Результат работы функции родителя передавать в качестве результата работы функции наследника.
7. INCT_Interpreter.NextOperation - САМ система вызывает NextOperation только у наследника, в начале интерпретации очередной операции. Вызывать NextOperation родителя всегда.

На странице Примеры интерпретаторов с исходными кодами приведён пример исходных кодов библиотеки интерпретатора-наследника, собранной из них 'dll', и пример файла настроек.

Список используемых типов

Все типы, необходимые для взаимодействия CAM системы с программной библиотекой интерпретатора, приведены в библиотеках типов. Где взять пакеты с ними и какую версию выбрать, описано на странице Программная библиотека интерпретатора.

Следующие TLB-файлы содержат описание типов, которые используются при взаимодействии с библиотекой интерпретатора:

STNCInterpreter.tlb — основной перечень типов

STNCAnalyzer.tlb — типы, используемые для управления анализом текста управляющей программы

STNCFormers.tlb — типы, используемые для упрощения построения траектории движения инструмента (линейные/круговые перемещения, циклы).

STNCConnect.tlb — типы, используемые для защиты интерпретатора.

STNCCustomInterpreter.tlb — типы для интерпретатора с собственными параметрами.

Основной перечень типов (STNCInterpreter.tlb)

Типы	Описание типов
TNCExecObj INCT_BlockExecObject INCT_BlockExecArray	Список объектов для интерпретации
TNCSpindleCommand INCT_BaseReg INCT_SysState INCT_SysState2 INCT_SpindleState	Список регистров
TNCOperationGroup INCT_Equipment INCT_Equipment2 INCT_Equipment_Channels INCT_Channel	Параметры оборудования
TNC3DPoint TNCFeedType TNCPlaneType	Формирование траектории движения инструмента в осях станка

Типы	Описание типов
TNCLengthCompensationMode TNCRadiusCompensationMode TNCGotoMode TNCDrillingCycleType TNCLatheCycleType TNCSystemMeasure TNCFeedrateMeasure TNCThreadOrientation TNCThreadLeadType TNCTooltipMode TNCEulerAnglesConvention TNCFlipType TNCMillingCycleType TNCAddOriginFlipMode TNCSPositioningMode TNCOriginSolutionState INCT_CLData INCT_CLData2 INCT_CLData_CutTo5d INCT_CLData_AutoAxesBrakes INCT_CLData_Comment INCT_CLData_BreakCommand INCT_CLData_Insert INCT_CLData_Print INCT_CLData_Wait INCT_CLData_LngID INCT_AddOriginByShiftAndRotate INCT_AddOriginByPhysicAxes INCT_OriginSolver	
INCT_CLData_PocketCycleFormer	Цикл фрезерования кругового кармана
TNC3DMatrix INCT_GeomCLData	Формирование траектории движения инструмента в геометрических осях
TNCMessageType INCT_Kernel INCT_Kernel2	Интерфейс ядра CAM системы

Типы	Описание типов
TNCGetPartDirection TNCGetPartOrigin INCT_ProgramManager INCT_ProgramManager2 INCT_Program INCT_GetPartByUser	Взаимодействие с текстом управляющей программы
TNCSign TNCPoint TNCValueAvailType INCT_RegisterValues INCT_Register INCT_RegisterList	Список регистров интерпретатора
INCT_InterpreterState	Токарный цикл обработки резанием
INCT_Interpreter	Интерфейс интерпретатора

Типы, используемые для управления анализом текста управляющей программы (STNCAnalyzer.tlb)

Типы	Описание типов
INCT_Analyzer INCT_Analyzer2 INCT_Comments INCT_Brackets INCT_Quotes INCT_UnionLineChars INCT_EndOfLineChars	Управление анализом текста управляющей программы

Типы, используемые для упрощения построения траектории движения инструмента (STNCFormers.tlb)

Типы	Описание типов
INCT_LinearMotionFormer INCT_GeomLinearMotionFormer	Линейные перемещения
TNCCircularMode TNCConstructArcMode	Перемещения по дуге

Типы	Описание типов
TNCAxisType INCT_CircularMotionFormer	
TNCDCAxisIndexes TNCDrillingCycleMoveType TNCReturnLevel INCT_DrillingCycleEvents INCT_DrillingCycleFormer	Цикл сверления
TNCDrillingCycleShiftType TNCShiftAxis TNCShiftValues TNCShiftPath SetOfTNCShiftPath INCT_DrillingCycleWithShiftFormer	Цикл сверления со сдвигом
TNCStartLevelType TNCDegressionType TNCClearanceType TNCRetractionType INCT_PeckDrillingCycleEvents INCT_PeckDrillingCycleFormer	Цикл сверления с ломкой стружки
TNC2DPoint TNCLCAxisIndexes TNCLatheCycleMoveType TNCGrooveDirection INCT_LatheCycleEvents INCT_LatheGroovingCycleFormer	Токарный цикл точения канавок
TNCThreadApproach TNCThreadCuttingDirection TNCThreadSequenceType INCT_LatheThreadCuttingCycleEvents INCT_LatheThreadCuttingCycleFormer INCT_LatheThreadCuttingCycleFormer2	Токарный цикл резьбонарезания
TNCLSRPointType TNCCutDepthCalcMethod TNCPassReturnMethod	Токарный цикл обработки резанием

Типы	Описание типов
TNCMachiningProcessing TNCLatheMachiningType TNCLatheMachiningDirection TNCLatheStepType TNCLatheStartPointMode TNCLatheMachiningStyle INCT_LatheCycleArcEvents INCT_LatheStockRemovalCycleEvents INCT_LatheStockRemovalCycleEvents2 INCT_LatheStockRemovalCycleEvents3 INCT_LatheStockRemovalCycleFormer INCT_LatheStockRemovalCycleFormer2	
TNCMillingThreadType TNCMillingCycleMoveType INCT_MillingCycleEvents INCT_ThreadMillingCycleFormer	Цикл фрезерования резьбы
TNCFlatSpiralStepType TNCSpiralDirection TNCTypeOfProcessing TNCPocketCycleMoveType INCT_PocketCycleEvents INCT_PocketCycleFormer	Цикл фрезерования кругового кармана

Типы, используемые для защиты интерпретатора (STNCConnect.tlb)

Типы	Описание типов
INCT_Connect	Интерфейс защиты

Типы для интерпретатора с собственными параметрами (STNCCustomInterpreter.tlb)

Типы	Описание типов
TCustomParameterType INCT_CustomParameter INCT_CustomParameters	Описание параметров: имя (Name) и тип (ParamType: `cpInt`, `cpFlt`, `cpStr`, `cpBol`, `cpArray`, `cpComplex`) каждого параметра; у параметров типов `cpArray` и `cpComplex` есть

Типы	Описание типов
	вложенные параметры (Count, Childs, ChildsByName)
INCT_CustomValue INCT_CustomValues	Значения параметров: имя (Name) и значение (Int, Flt, Str); вложенные значения — Count, Childs, ChildsByName
INCT_CustomInterpreter	Интерфейс интерпретатора с собственными параметрами (см. ниже)

INCT_CustomInterpreter — интерфейс, который реализует объект интерпретатора (тот же, что возвращает NCT_GetInterface), если у интерпретатора есть собственные параметры. Значения параметров хранятся в файле `*.sncix`: файл ссылается на интерпретатор и хранит значения его параметров, своей библиотеки у него нет.

Файл `*.sncix` создают в окне Конфигуратор интерпретатора УП. Окно открывается пунктом Конфигуратор интерпретатора УП в списке Расширения CAM системы или пунктом Настроить интерпретатор в NC Тюнере. В окне выберите файл интерпретатора в поле Интерпретатор, задайте значения параметров и сохраните их в файл `*.sncix` кнопкой Сохранить как. Чтобы интерпретатор работал с этими значениями, выбирайте файл `*.sncix` вместо файла `*.snci` там же, где выбирают файл интерпретатора: в окне выбора файла интерпретатора доступны оба типа файлов.

Метод или свойство	Описание
Parameters	Свойство только для чтения, возвращает INCT_CustomParameters — перечень параметров интерпретатора. По нему окно Конфигуратор интерпретатора УП показывает параметры и сохраняет их значения в файл `*.sncix`.
SetCustomValues(Values: INCT_CustomValues)	CAM система вызывает метод после загрузки библиотеки интерпретатора, до чтения его регистров, и передаёт значения параметров из файла `*.sncix`. Если интерпретатор выбран своим файлом `*.snci`, а не файлом `*.sncix`, значение Values пустое (nil / null).

Интерпретатор, объект которого реализует INCT_CustomInterpreter, не может быть родителем: CAM система прекращает загрузку наследника с ошибкой.

Интерфейсы, реализуемые в интерпретаторе

В этом разделе описаны интерфейсы, которые должны быть реализованы в интерпретаторе, и интерфейс защиты, который CAM система передаёт защищённому интерпретатору.

Интерфейс интерпретатора

INCT_Interpreter - интерфейс, который должен быть реализован в программной библиотеке интерпретатора. Библиотека интерпретатора возвращает CAM системе объект, реализующий интерфейс в качестве результата при вызове функции NCT_GetInterface . Через этот интерфейс CAM система взаимодействует с интерпретатором на всех этапах интерпретации текста управляющей программы.

Описание методов интерфейса:

Метод	Описание метода	Описание использования
GetIdentifier	Функция должна возвратить CAM системе строку с идентификатором библиотеки интерпретатора. Входных параметров нет.	При загрузке интерпретатора в память CAM система вызывает функцию GetIdentifier и ожидает получить идентификатор программной библиотеки. Затем CAM система сравнивает его с идентификатором библиотеки файла настроек интерпретатора, если идентификаторы совпали, продолжает процесс загрузки.
GetRegisters	Функция должна возвратить CAM системе интерфейс INCT_RegisterList , который реализует список регистров интерпретатора. Входных параметров нет.	При загрузке интерпретатора в память CAM система вызывает функцию GetRegisters и ожидает получить список регистров интерпретатора, затем вызывает эту функцию для родителей интерпретатора. В результате CAM система формирует общий список регистров с учётом наследования.
SetInterface	Метод должен получить и, если это необходимо	При инициализации интерпретатора CAM система

Метод	Описание метода	Описание использования
	интерпретатору, сохранить передаваемые от CAM системы интерфейсы для последующего использования. Метод будет вызван несколько раз: по числу передаваемых интерфейсов. Входной параметр: Intf (интерфейс IUnknown) Выходных параметров нет.	последовательно передаёт библиотеке интерпретатора свои интерфейсы посредством метода SetInterface. Передаваемые интерфейсы могут использоваться при интерпретации текста управляющей программы и обеспечивают доступ к: • сформированному списку регистров INCT_SysState ; • управлению настройками анализа текста управляющей программы INCT_Analyzer ; • списку объектов для интерпретации INCT_BlockExecArray ; • настройкам оборудования INCT_Equipment ; • формирователю траектории движения инструментов в осях станка (если в файле настроек интерпретатора `UseGeomCLData` = False) INCT_CLData ; • формирователю траектории движения инструментов в геометрических осях (если `UseGeomCLData` = True) INCT_GeomCLData ; • взаимодействию с ядром CAM системы, обслуживающим

Метод	Описание метода	Описание использования
		интерпретатор INCT_Kernel; • интерпретатору-родителю INCT_Interpreter (если он есть); • контуру (интерфейс, переданный в INCT_Kernel.InterpreteContour), если интерпретатор запущен в режиме формирования контура; • защите INCT_Connect, только для защищённых интерпретаторов.
Initialize	Метод информирует интерпретатор об окончании передачи всех интерфейсов CAM системы и возможности выполнения инициализации интерпретатора. Входных параметров нет. Выходных параметров нет.	При завершении инициализации интерпретатора CAM система вызывает метод Initialize. Вызов метода информирует интерпретатор о том, что все интерфейсы CAM системы переданы, интерпретатор может выполнить необходимые действия для инициализации (например, получить номера осей станка).
NextOperation	Метод информирует интерпретатор о начале интерпретации следующей операции. Входных параметров нет. Выходных параметров нет.	Перед началом интерпретации очередной операции CAM система вызывает этот метод. Интерпретатор может выполнить необходимые действия (например, инициализировать внутренние переменные, установить порядок

Метод	Описание метода	Описание использования
		обработки управляющих программ операции).
TransLine	Метод передаёт интерпретатору кадр управляющей программы (или часть кадра, в зависимости от текущей итерации трансляции кадра) для попытки чтения лексемы или смысловой конструкции из его начала. В случае, если интерпретатор распознал лексему или конструкцию, он должен вернуть количество символов с начала строки, которые составляет распознанная лексема или конструкция. В ином случае возвращает количество равное 0. Входные параметры: CurrentLine (строка) - передаваемый кадр или часть кадра. Выходные параметры: • ProcessedChars (целое число), количество символов с начала строки, которые удалось распознать интерпретатору (CAM система не выполняет анализ этих символов). 0 - если лексема или смысловая конструкция не распознана.	В процессе трансляции кадра управляющей программы CAM система даёт возможность интерпретатору распознать лексему или смысловую конструкцию из начала текущего кадра. Для этого CAM система вызывает метод TransLine интерпретатора, передаёт ему в качестве входного параметра CurrentLine текущий кадр. В ответ в параметре ProcessedChars интерпретатор должен вернуть количество символов от начала переданной строки, составляющих лексему или смысловую конструкцию, если она распознана. Иначе, если лексема или конструкция не распознаны, интерпретатор возвращает значение, равное 0. Если интерпретатор распознал лексему или смысловую конструкцию, то она удаляется из текущей строки CurrentLine, и строка вновь передаётся интерпретатору для попытки чтения следующей лексемы или смысловой конструкции,

Метод	Описание метода	Описание использования
	• NextLine (логическое), после трансляции текущего кадра перейти к трансляции следующего кадра управляющей программы без интерпретации текущего. Все объекты INCT_BlockExecArray текущего кадра будут сохранены и обработаны вместе с объектами следующего кадра единым блоком. • BreakLine (логическое), прервать трансляцию текущего кадра с сохранением позиции в кадре и перейти к интерпретации сформированного INCT_BlockExecArray. Следующая трансляция начнётся с сохранённой позиции текущего кадра.	посредством вызова метода TransLine. Если интерпретатор не распознал лексему или смысловую конструкцию, то CAM система пытается определить регистр и его значение по описанному в этой статье алгоритму. После чего все считанные символы удаляются из строки CurrentLine, и строка передаётся интерпретатору для попытки чтения следующей лексемы или смысловой конструкции, посредством вызова метода TransLine. Процесс продолжается до завершения строки или, если параметр BreakLine принял значение Истина. Если после вызова TransLine выходной параметр BreakLine принимает значение Истина, то трансляция текущего кадра останавливается и CAM система выполняет процесс интерпретации оттранслированной части кадра. То есть, сформированный к этому моменту INCT_BlockExecArray будет сынтерпретирован. Если после вызова TransLine выходной параметр NextLine

Метод	Описание метода	Описание использования
		имеет значение Истина, то после завершения трансляции текущего кадра, CAM система считывает и транслирует следующий кадр управляющей программы, без интерпретации текущего кадра. Сформированный по результатам трансляции текущего кадра INCT_BlockExecArray не очищается, и дополняется объектами, созданными при трансляции следующего кадра. После завершения трансляции следующей строки и, если параметр NextLine вновь не принимает значение Истина, производится интерпретация накопленного в результате трансляции двух кадров INCT_BlockExecArray. Если же параметр NextLine снова Истина, то после завершения трансляции второго кадра, CAM система считывает и транслирует третий кадр и т.д.
DefineRegister	Функция должна выполнить поиск регистра по адресу, возвратить результат поиска и индекс найденного регистра. Входной параметр: Addr (строка) - адрес регистра.	При чтении лексемы CAM система пытается определить регистр в считанном из кадра управляющей программы наборе символов. Для этого CAM система вызывает функцию

Метод	Описание метода	Описание использования
	Выходной параметр: Index (целое число) - индекс найденного регистра. Результат (логическое): Истина - регистр найден, Ложь - не найден.	DefineRegister интерфейса интерпретатора, куда в качестве адреса регистра Addr передаёт считанный набор символов. При положительном ответе интерпретатора, регистр считается определённым и его индекс содержится в выходном параметре Index. При отрицательном ответе интерпретатора CAM система пытается найти регистр по адресу в списке регистров. При реализации функции (INCT_Interpreter.DefineRegister) рекомендуется выполнять поиск и определение только тех регистров, которые не могут быть определены ядром CAM системы однозначно. В остальных случаях поиск не выполнять, возвращать отрицательный результат.
BeforeInterprete	Метод информирует интерпретатор об окончании трансляции текущего кадра и начале его интерпретации. Входных параметров нет. Выходных параметров нет.	Перед интерпретацией текущего кадра управляющей программы CAM система вызывает метод BeforeInterprete интерпретатора. Метод предназначен для инициализации переменных и объектов, связанных с текущим кадром, если это необходимо.

Метод	Описание метода	Описание использования
Interprete	Метод передаёт интерпретатору номер элемента из списка объектов для интерпретации текущего кадра. Интерпретатор должен сынтерпретировать переданный объект, т.е. по номеру объекта определить его тип и, по необходимости, добавить команду по формированию элемента(-ов) траектории движения инструмента в INCT_CLData. Входной параметр: ExecNum (целое число) - номер элемента списка объектов для интерпретации (INCT_BlockExecArray). Выходных параметров нет.	В процессе интерпретации текущего кадра управляющей программы, если список INCT_BlockExecArray не пуст, CAM система последовательно выполняет метод Interprete, для каждого из объектов списка. Метод предназначен для интерпретации объекта из списка, т.е. для формирования элемента траектории движения инструмента по оттранслированному ранее объекту. Пример для стойки Fanuc. Если переданный объект является регистром с адресом F(подача), то для изменения подачи текущей траектории движения инструмента, необходимо выполнить метод INCT_CLData.OutFeed. В качестве параметра подачи передать значение регистра.
AfterInterprete	Метод информирует интерпретатор об окончании интерпретации текущего кадра. Входных параметров нет. Выходных параметров нет.	После интерпретации текущего кадра управляющей программы, CAM система вызывает метод AfterInterprete интерпретатора. Метод предназначен для освобождения памяти от

Метод	Описание метода	Описание использования
		переменных и объектов, связанных с обработанным кадром, если это необходимо.
EndOfProgram	Функция должна возвратить CAM системе признак окончания программы, если при интерпретации текста управляющей программы была обработана команда окончания программы. Трансляция и интерпретация управляющей программы завершается. Входных параметров нет. Результат (логическое): Истина - программа окончена.	Перед выполнением очередного метода Interprete, CAM система вызывает функцию EndOfProgram. Если функция возвращает значение Истина, то интерпретация текущего кадра и трансляция последующего текста управляющей программы прекращаются.

Список регистров интерпретатора

INCT_RegisterList - интерфейс, который возвращает функция интерпретатора INCT_Interpreter.GetRegisters. Интерфейс обеспечивает работу со списком регистров интерпретатора.

Описание методов интерфейса:

Метод	Описание метода	Описание использования
Get_Count	Функция должна возвратить в CAM систему количество регистров интерпретатора. Входных параметров нет.	При формировании общего списка регистров, CAM система вызывает функцию Get_Count, получает количество регистров, а затем последовательно, для каждого регистра, вызывает функцию Get_Item.
Get_Item	По номеру регистра функция должна возвратить в CAM систему интерфейс INCT_Register (см. ниже). Входной параметр: Index (целое число) - номер регистра. Нумерация начинается с 0.	

INCT_Register - интерфейс, который возвращает функция интерпретатора INCT_RegisterList.Get_Item. Интерфейс обеспечивает работу с параметрами выбранного регистра интерпретатора.

Описание методов интерфейса:

Метод	Описание метода	Описание использования
Get_Addr	Функция должна возвратить в CAM систему адрес регистра (строка). Входных параметров нет.	Адрес регистра используется CAM системой для поиска регистра в списке регистров.
Get_Id	Функция должна возвратить в CAM систему	Идентификатор регистра в кинематической схеме

Метод	Описание метода	Описание использования
	идентификатор регистра в кинематической схеме станка (строка). Входных параметров нет.	станка, если применимо. Иначе строка пустая.
Get_Name	Функция должна возвратить в CAM систему имя регистра (строка). Входных параметров нет.	Уникальное имя регистра. Используется для однозначного определения регистра в списке.
Get_Comment	Функция должна возвратить в CAM систему комментарий регистра (строка). Входных параметров нет.	Функция вызывается, но в текущей версии значение, возвращаемое функцией, не используется.
Get_Sign	Функция должна возвратить в CAM систему тип знака регистра (перечисляемый тип TNCSign). sgnNoSign - без знака; sgnMinus - только минус; sgnPlusAndMinus - плюс и минус. Входных параметров нет.	Функция вызывается, но в текущей версии значение, возвращаемое функцией, не используется.
Get_Point	Функция должна возвратить в CAM систему тип точки регистра (перечисляемый тип TNCPoint). pntNoPoint - без точки; pntPoint - точка может присутствовать, может нет; pntAlwaysPoint - точка всегда присутствует. Входных параметров нет.	Функция вызывается, но в текущей версии значение, возвращаемое функцией, не используется.
Get_BeforePoint	Функция должна возвратить в CAM систему количество знаков до точки регистра	Функция вызывается, но в текущей версии значение, возвращаемое функцией, не используется.

Метод	Описание метода	Описание использования
	(целое число). Входных параметров нет.	
Get_AfterPoint	Функция должна возвратить в CAM систему количество знаков после точки регистра (целое число). Входных параметров нет.	Функция используется для определения правильности дуг, поэтому важно правильно задать количество знаков после точки. Формирователь траектории использует данную функцию внутри ядра.
Get_LeadingZeros	Функция должна возвратить в CAM систему признак наличия лидирующих нулей регистра (логическое). Входных параметров нет.	Функция вызывается, но в текущей версии значение, возвращаемое функцией, не используется.
Get_TrailingZeroes	Функция должна возвратить в CAM систему признак наличия незначащих нулей регистра (логическое). Входных параметров нет.	Функция вызывается, но в текущей версии значение, возвращаемое функцией, не используется.
Get_ValueAvail	Функция должна возвратить в CAM систему тип регистра (перечисляемый тип TNCValueAvailType). vatValue - регистр состоит из адреса и значения; vatOnlyAddr - регистр состоит только из адреса; vatBoth - регистр может состоять либо только из адреса, либо из адреса и значения. Входных параметров нет.	Используется при определении регистра и его значения.
Get_Values	Функция должна возвратить в CAM систему список	INCT_RegisterValues.Get_Count - количество значений

Метод	Описание метода	Описание использования
	значений регистра (интерфейс INCT_RegisterValues). Входных параметров нет.	INCT_RegisterValues.Get_Value - значение по индексу (Index - целое число) Список значений регистра используется при определении регистра и его значения.
Get_Modal	Функция должна возвратить в CAM систему признак модальности регистра (логическое). Входных параметров нет.	Используется при копировании значений регистров после интерпретации текущего кадра. Модальные регистры попадают в INCT_BlockExecArray только, если значение регистра изменилось в текущем кадре. Немодальные всегда, если присутствуют в текущем кадре.

Интерфейс защиты

Если в контейнере с размещённым интерпретатором присутствует файл имя_библиотеки_интерпретатора.dat (далее файл с секретными данными), то при инициализации такому интерпретатору дополнительно передаётся интерфейс INCT_Connect.

INCT_Connect - интерфейс, который обеспечивает доступ интерпретатора к данным из файла с секретными данными. При этом файл с данными не извлекается из контейнера и поэтому доступен для чтения только через переданный интерпретатору интерфейс.

Интерпретатор нужно разрабатывать таким образом, чтобы его работа была обеспечена только после прохождения проверки данных, полученных через интерфейс INCT_Connect. Места проверок, их количество и алгоритм определяются разработчиком самостоятельно.

Описание методов интерфейса:

Метод	Описание метода	Описание использования
Get_Count	Функция возвращает количество байт, содержащихся в файле с секретными данными (целое число).	Пользуясь предложенными методами необходимо выполнить проверку данных, размещённых разработчиком интерпретатора в контейнере, с данными, передаваемыми CAM системой интерпретатору (должны совпадать). Если разработчик разместил в контейнере файл sample.dat (библиотека интерпретатора называется sample.dll), размером 5 байт со следующими байтами (в шестнадцатеричной записи): 01 02 03 04 05 Функция Get_Count вернёт значение 5. Функция Get_Data(0) вернёт значение 1, Get_Data(2)
Get_Data	Функция возвращает байт из файла с секретными данными, с номером, заданным во входном параметре. Входные параметры: Index (целое число) - номер возвращаемого байта. Нумерация байт от 0. Результат: байт из файла с секретными данными, с номером, заданным в параметре Index.	

Метод	Описание метода	Описание использования
Get_SCData	Метод возвращает указатель на секретные данные из файла. Выходной параметр: Data (указатель) - указатель на секретные данные.	вернёт значение 3, Get_Data(4) вернёт значение 5. Метод Get_SCData вернёт в параметре Data указатель на память с размещёнными данными 01 02 03 04 05.

Интерфейсы CAM системы

В этом разделе описаны интерфейсы, которые реализует CAM система.

Основные

В разделе описаны основные интерфейсы CAM системы, реализованные для использования в интерпретаторе.

Интерфейс ядра CAM системы

INCT_Kernel - интерфейс ядра CAM системы. Интерфейс обеспечивает работу с текстом управляющей программы и перезапуском интерпретатора в режиме получения контура.

Описание методов интерфейса:

Метод	Описание метода	Описание использования
Get_ProgramManager	Функция возвращает интерфейс взаимодействия с УП текущей операции (INCT_ProgramManager)	1. Если для текущей операции постпроцессор формирует несколько файлов, то при помощи интерфейса взаимодействия с УП можно настроить порядок передачи текста УП интерпретатору. 2. Для получения только части текста УП (например, по меткам в программе).
InterpreteContour	Метод запускает новый экземпляр интерпретатора для анализа текста управляющей программы и получения контура. Входные параметры: NCProgram (интерфейс INCT_Program) - текст управляющей программы; Contour (интерфейс IUnknown) - интерфейс контура.	Если для формирования траектории движения инструмента токарного цикла обработки контура (аналог CYCLE95 для стойки Sinumerik) будет недостаточно встроенного в ядро формирователя (Токарный цикл обработки резанием), существует возможность запуска новой копии интерпретатора для самостоятельного получения геометрии по управляющей программе траектории контура. Для этой цели предназначен метод InterpreteContour.

Метод	Описание метода	Описание использования
		Для формирования геометрии контура необходимо: 1. Получить текст управляющей программы по меткам контура, при помощи соответствующей функции (например, GetPartByLabels). 2. Реализовать объект **Contour**, который должен иметь интерфейс для передачи и получения траектории движения инструмента в том виде и формате, в котором это нужно разработчику интерпретатора. 3. Вызвать метод InterpreteContour, которому в качестве параметров передать полученные на первом и втором шагах объекты (текст управляющей программы контура и объект Contour). В процессе исполнения метода InterpreteContour, CAM система получит от библиотеки интерпретатора новый экземпляр INCT_Interpreter, выполнит инициализацию нового интерпретатора, в методе SetInterface, в дополнение к обычному списку интерфейсов, передаст

Метод	Описание метода	Описание использования
		новому экземпляру ещё и интерфейс Contour, выполнит трансляцию и интерпретацию переданного текста управляющей программы аналогично обычной интерпретации. В процессе интерпретации, интерпретатор, понимая, что находится в режиме формирования контура (см. Get_IsContourMode) должен формировать траекторию контура используя собственный объект Contour и разработанный для сохранения геометрической траектории интерфейс. После завершения работы метода InterpreteContour в объекте Contour будет содержаться геометрическая траектория контура, готовая к использованию для формирования траектории движения инструмента в первичном экземпляре интерпретатора.
Get_IsContourMode	Функция возвращает признак работы экземпляра интерпретатора в режиме формирования контура. Входных параметров нет. Результат: (логический) Истина - в режиме	Если в процессе работы интерпретатор самостоятельно запускает свою новую копию (подробнее см. InterpreteContour), то вновь созданному экземпляру интерпретатора необходимо понимать в каком режиме он

Метод	Описание метода	Описание использования
	формирования контура, Ложь - в режиме формирования траектории движения инструмента.	запущен. Если новый интерпретатор запущен в режиме формирования контура, тогда по тексту управляющей программы интерпретатор должен формировать геометрию контура (см. объект Contour). Если новый интерпретатор запущен не в режиме формирования контура, тогда по тексту управляющей программы интерпретатор должен формировать траекторию движения инструмента.

INCT_Kernel2 - интерфейс ядра CAM системы.

Описание методов интерфейса:

Метод	Описание метода	Описание использования
DoSubprogram	Метод переключает трансляцию и интерпретацию на заданный параметром NCProgram текст УП, заданное параметром Count количество раз. После завершения будет продолжена трансляция текста УП с места останова. Входные параметры: NCProgram (интерфейс INCT_Program) - текст управляющей программы (подпрограммы); Count (целое число) - количество повторений.	Применяется для выполнения подпрограмм (CALL для Sinumerik, M98 для Fanuc). Для получения текста подпрограммы нужно воспользоваться одной из следующих функций: GetPartByLabels, GetPartByLabelsFromCurPos, GetPartByRegisters.

Метод	Описание метода	Описание использования
Log	Метод выводит сообщение в протокол CAM системы Входные параметры: Msg (строка) - текст сообщения; MsgType (тип TNCMessageType) - тип сообщения: nmtInfo - информационное nmtSuccess - успешно nmtWarning - внимание nmtError - ошибка nmtCritical - критично nmtException - исключение nmtDebug - отладочное	Используется для отладки работы интерпретатора
Notify	Метод выводит сообщение в панель уведомления CAM системы Входные параметры: Msg (строка) - текст сообщения; MsgType (тип TNCMessageType) - тип сообщения (описание см. в методе Log).	Используется для предупреждения пользователя при работе интерпретатора.

Взаимодействие с текстом управляющей программы

INCT_ProgramManager - интерфейс для взаимодействия с текстом управляющей программы (УП). Интерфейс обеспечивает работу с текстом управляющей программы, порядком передачи текстов УП интерпретатору.

Описание методов интерфейса:

Метод	Описание метода	Описание использования
Get_Count	Функция возвращает количество текстов УП (файлов*) сформированных постпроцессором для текущей операции. Входных параметров нет.	В случае, если постпроцессор генерирует несколько текстов с УП (файлов*), может потребоваться изменить порядок передачи текстов для обработки интерпретатором. Для определения наличия нескольких текстов с УП нужно воспользоваться функцией Get_Count. В некоторых случаях постпроцессор может генерировать уникальные имена файлов* со вспомогательными данными (без текста УП), которые не нужно интерпретировать. Или напротив, формировать заголовочный файл из которого осуществляется вызов остальных текстов УП. Для идентификации таких файлов по имени предназначена функция Get_Name.
Get_Name	Функция возвращает имя файла* с текстом управляющей программы (строка) по индексу текста с УП (файла*), передаваемому во входном параметре. Входной параметр: Index (целое число) - номер текста УП (файла*), сформированного постпроцессором для текущей операции. Значение от 0 до Get_Count -1	
AddToOrderList	Метод добавляет текст УП (файл*) с заданным индексом в список интерпретации. Тексты УП будут передаваться интерпретатору в порядке,	

Метод	Описание метода	Описание использования
	заданном в этом списке от первого добавленного в список к последнему. Тексты с УП не добавленные в список не будут переданы интерпретатору. Если в список УП не был добавлен ни один текст с УП, в интерпретатор будут переданы все файлы в порядке, сформированном постпроцессором для текущей операции. Входной параметр: Index (целое число) - номер текста УП (файла*), сформированного постпроцессором для текущей операции. Значение от 0 до Get_Count -1	Если для интерпретации требуется изменить порядок обработки текстов УП, сформированных постпроцессором, то для этого предназначена функция добавления текста УП в список интерпретации (AddToOrderList). Тексты УП будут передаваться интерпретатору в порядке, заданном в списке, от первого добавленного в список к последнему.
GetPartByIndex	Функция возвращает текст управляющей программы (интерфейс INCT_Program) по индексу текста с УП (файла*), передаваемому во входном параметре. Входной параметр: Index (целое число) - номер текста УП (файла*), сформированного постпроцессором для текущей операции. Значение от 0 до Get_Count -1	Для формирования порядка обработки текстов с УП (файлов*) интерпретатором, может потребоваться выполнение анализа содержимого одного или нескольких текстов с УП текущей операции. Выполнение функции с номером текста вернёт интерфейс INCT_Program, при помощи функций которого нужно осуществлять чтение строк искомого текста УП.
GetPartByLabels	Функция возвращает текст управляющей программы	Если для формирования траектории движения

Метод	Описание метода	Описание использования
	(интерфейс INCT_Program) по меткам, передаваемым во входных параметрах. Поиск производится с начала текста УП. Входные параметры: StartLabel (строка), StopLabel (строка) - метки начала и окончания текста управляющей программы.	инструмента токарного цикла обработки контура (аналог CYCLE95 для стойки Sinumerik) будет недостаточно встроенного в ядро формирователя (Токарный цикл обработки резанием), существует возможность получения текста управляющей программы с траекторией цикла для её последующего самостоятельного анализа. Для этой цели предназначена описываемая функция.

* - при постпроцессировании для интерпретации файлы с текстами управляющих программ не создаются на диске. Тексты УП существуют только в памяти. Имена файлов и нумерация сохранены в связи с существующей технологией и для формирования порядка обработки текстов с УП.

INCT_ProgramManager2 - интерфейс для взаимодействия с текстом управляющей программы (УП).

Описание методов интерфейса:

Метод	Описание метода	Описание использования
GetPartByLabelsFromCurPos	Функция возвращает текст управляющей программы (интерфейс INCT_Program) по меткам, передаваемым во входных параметрах. Поиск производится с текущей позиции. Входные параметры: StartLabel (строка), StopLabel (строка) - метки начала и	Смотрите применение к GetPartByLabels.

Метод	Описание метода	Описание использования
	окончания текста управляющей программы; Direction (TNCGetPartDirection) - направление поиска: gpdForward - вперёд от текущей позиции, gpdBackward - назад от текущей позиции.	
GetPartByRegisters	Функция возвращает текст управляющей программы (интерфейс INCT_Program) по регистрам, передаваемым во входных параметрах. Входные параметры: StartReg (INCT_Register), StartValues (INCT_RegisterValues), StopReg (INCT_Register), StopValues (INCT_RegisterValues) - регистры и значения начала и окончания текста управляющей программы; Origin (TNCGetPartOrigin) - откуда искать: gpoFromCurPos - с текущей позиции управляющей программы, gpoEntireScope - с начала управляющей программы.	

INCT_GetPartByUser - интерфейс для взаимодействия с текстом управляющей программы (УП).

Описание методов интерфейса:

Метод	Описание метода	Описание использования
GetPartByUser	Функция возвращает интерфейс INCT_Program, с помощью которого можно	Функция позволяет определить начало/конец подпрограммы по любым

Метод	Описание метода	Описание использования
	работать с управляющей программой. Входной параметр: Origin (TNCGetPartOrigin) - откуда искать: gpoFromCurPos - произвести поиск с текущей позиции управляющей программы gpoEntireScope - произвести поиск с начала управляющей программы	заданным меткам. 1) С помощью GetPartByUser получаем интерфейс INCT_Program 2) Например в INCT_InterpreterState можно передать значения меток начала/конца подпрограмм. Или использовать для этих целей регистры. 3) После запуска процедуры DoSubprogram интерфейса INCT_Kernel2 происходит поиск подпрограммы. Если был использован INCT_InterpreterState, то можно получить ранее сохраненные метки, или обнаружить в методе TransLine нужную метку начала подпрограммы, и вызвать IsStart. Аналогичным способом используется процедура IsEnd.
IsStart	Процедура определяет начало подпрограммы	
IsEnd	Процедура определяет конец подпрограммы	

INCT_Program - интерфейс для получения текста УП. Интерфейс реализует работу с текстом УП, возвращаемым некоторыми функциями INCT_ProgramManager .

Описание методов интерфейса:

Метод	Описание метода	Описание использования
EndOfFile	Функция возвращает признак окончания строк. Истина - текст УП закончился. Ложь - текст УП не вычитан полностью.	После получения интерфейса, каждый следующий вызов GetNextLine возвращает очередную строку текста УП, начиная с 1-ой и до последней строки.

Метод	Описание метода	Описание использования
GetNextLine	Функция возвращает очередную строку УП	После получения последней строки функция EndOfFile вернёт значение Истина.

Список регистров

INCT_SysState - интерфейс состояния системы станок-стойка. В настоящий момент интерфейс обеспечивает работу со сформированным списком регистров (с учётом файла настроек, программной библиотеки и родителей интерпретатора). Для отличия сформированного списка регистров от исходных регистров интерпретатора, файла настроек и интерпретаторов-родителей, сформированный список регистров называется базовый список регистров.

Описание методов интерфейса:

Метод	Описание метода	Описание использования
FindFirstRegister	Функция возвращает первый, найденный по адресу, базовый регистр (интерфейс INCT_BaseReg). Входной параметр: Addr (строка) - адрес регистра	Функция может использоваться для поиска базового регистра по адресу при анализе текста управляющей программы для реализации INCT_Interpreter.TransLine.
FindNextRegister	Функция возвращает следующий, начиная с FromIdx, найденный по адресу базовый регистр (интерфейс INCT_BaseReg). Входной параметр: FromIdx (целое число) - начальный индекс, Addr (строка) - адрес регистра	Функция может использоваться для поиска базового регистра по адресу при анализе текста управляющей программы для реализации INCT_Interpreter.TransLine.
Get_BR_ByIndex	Функция возвращает базовый регистр по индексу (интерфейс INCT_BaseReg) Входной параметр: Index (целое число) - индекс регистра.	Функция может использоваться для получения базового регистра по индексу при переборе базовых регистров (например, с целью получения начальных значений осей положения инструмента, связанных с базовыми регистрами).

Метод	Описание метода	Описание использования
Get_BR_ByName	Функция возвращает базовый регистр по имени (интерфейс INCT_BaseReg) Входной параметр: Name (строка) - имя регистра.	Функция может быть использована для поиска базового регистра по имени, для определения индекса регистра при реализации INCT_Interpreter.DefineRegister.
Get_BR_Count	Функция возвращает количество базовых регистров.	Функция может использоваться для получения количества базовых регистров для последующего перебора базовых регистров (например, с целью получения начальных значений осей положения инструмента, связанных с базовыми регистрами).

INCT_SysState2 - интерфейс состояния системы станок-стойка.

Описание методов интерфейса:

Метод	Описание метода	Описание использования
Get_SpindleState	Функция возвращает состояния шпинделя (интерфейс INCT_SpindleState)	Состояния шпинделя используется для хранения и обмена информацией между интерпретатором и его родителем(-и)
Get_Reg_ByIndex	Функция возвращает регистр по индексу (интерфейс INCT_Register) Входной параметр: Index (целое число) - индекс регистра.	Функции позволяют найти регистр по имени (Get_Reg_ByName) или перебрать регистры по индексу (Get_Reg_ByIndex, Get_Reg_Count) по необходимому параметру. Например, для получения значений регистра

Метод	Описание метода	Описание использования
Get_Reg_ByName	Функция возвращает регистр по имени (интерфейс INCT_Register) Входной параметр: Name (строка) - имя регистра.	INCT_Register.Values (недоступных в INCT_BaseReg)
Get_Reg_Count	Функция возвращает количество базовых регистров.	

INCT_SpindleState - интерфейс состояния шпинделя. Состояние шпинделя единое для интерпретатора и его родителей. Предназначено для хранения и обмена состоянием шпинделя между интерпретатором и его родителем(-и).

Описание методов интерфейса:

Метод	Описание метода
Get_MaxSpeed	Функция возвращает максимальную скорость вращения шпинделя (вещественное число)
Set_MaxSpeed	Метод задаёт максимальную скорость вращения шпинделя. Входной параметр: Value (вещественное число) - максимальная скорость шпинделя.
Get_RPMSpeed	Функция возвращает скорость вращения шпинделя (вещественное число), если режим постоянной скорости резания выключен (Get_CSS = Ложь).
Set_RPMSpeed	Метод задаёт скорость вращения шпинделя. Входной параметр: Value (вещественное число) - скорость вращения шпинделя.

Метод	Описание метода
Get_CSSSpeed	Функция возвращает скорость вращения шпинделя (вещественное число) для режима постоянной скорости резания (Get_CSS = Истина)
Set_CSSSpeed	Метод задаёт скорость вращения шпинделя для режима постоянной скорости резания. Входной параметр: Value (вещественное число) - скорость вращения шпинделя.
Get_Range	Функция возвращает номер диапазона вращения шпинделя (целое число)
Set_Range	Метод задаёт номер диапазона вращения шпинделя. Входной параметр: Value (целое число) - номер диапазона вращения шпинделя.
Get_Direction	Функция возвращает направление вращения шпинделя (логическое)
Set_Direction	Метод задаёт направление вращения шпинделя. Входной параметр: Value (логическое) - направление вращения шпинделя. Истина - по часовой стрелке Ложь - против часовой стрелки
Get_CSS	Функция возвращает признак постоянной скорости резания шпинделя (логическое)
Set_CSS	Метод задаёт признак постоянной скорости резания шпинделя. Входной параметр: Value (логическое) - признак постоянной скорости резания шпинделя. Истина - режим постоянной скорости резания включен Ложь - режим постоянной скорости резания выключен

Метод	Описание метода
Get_MaxRPM	Функция возвращает максимальное число оборотов для постоянной скорости резания шпинделя (вещественное число) (Get_CSS = Истина)
Set_MaxRPM	Метод задаёт максимальное число оборотов для постоянной скорости резания шпинделя. Входной параметр: Value (вещественное число) - максимальное число оборотов для постоянной скорости резания шпинделя.
Get_Angle	Функция возвращает угол ориентированного останова шпинделя (вещественное число)
Set_Angle	Метод задаёт угол ориентированного останова шпинделя. Входной параметр: Value (вещественное число) - угол ориентированного останова шпинделя.
Get_Command	Функция возвращает статус шпинделя (последняя выполненная команда) (TNCSpindleCommand)
Set_Command	Метод задаёт статус шпинделя (последняя выполненная команда). Входной параметр: Value (TNCSpindleCommand) - статус шпинделя. scOn - включен scOff - выключен scOrient - ориентированный останов
Get_IsChanged	Функция возвращает признак изменения любого параметра INCT_SpindleState, кроме номера шпинделя, с момента исполнения последней команды шпинделя (логическое). Истина - один из параметров изменился Ложь - параметры не менялись

Метод	Описание метода
Set_IsChanged	Метод задаёт признак изменения любого параметра INCT_SpindleState, кроме номера шпинделя, с момента исполнения последней команды шпинделя. Используется при наследовании интерпретаторов. Команду на включение/выключение/останов шпинделя выполняет один из интерпретаторов, как правило последний наследник. Входной параметр: Value (логическое) - признак изменения параметра.
Get_Num	Функция возвращает номер шпинделя (целое число, нумерация с 1, как в INCT_CLData.SetActiveSpindle).
Set_Num	Метод задаёт номер шпинделя. Входной параметр: Value (целое число) - номер шпинделя.
Get_IsChangedNum	Функция возвращает признак изменения номера шпинделя (логическое)
Set_IsChangedNum	Метод задаёт признак изменения номера шпинделя . Входной параметр: Value (логическое) - признак изменения номера шпинделя.

INCT_BaseReg - интерфейс обеспечивает работу с базовым регистром.

Описание методов интерфейса:

Метод	Описание метода	Описание использования
SetValue	Метод задаёт новое значение базовому регистру. Базовый регистр имеет три поля для хранения значения - строка, вещественное число, целое число. Если новое значение не пустое, то оно записывается в поле со	Метод может быть использован при интерпретации лексем и смысловых конструкций, которые влияют на значение регистра (например, функции).

Метод	Описание метода	Описание использования
	строкой, конвертируется и записывается в поля с целым и вещественным числом. Входной параметр: Value (строка) - новое значение регистра.	
Get_Addr	Функция возвращает адрес базового регистра (строка). Входных параметров нет.	Функции могут быть использованы для получения соответствующих параметров базового регистра при инициализации, трансляции или интерпретации текста управляющей программы.
Get_Name	Функция возвращает имя базового регистра (строка). Входных параметров нет.	
Get_Id	Функция возвращает идентификатор базового регистра в кинематической схеме станка (строка). Входных параметров нет.	
Get_Idx	Функция возвращает индекс базового регистра в списке регистров (целое число). Входных параметров нет.	
Get_Str	Функция возвращает строковое значение базового регистра (строка). Входных параметров нет.	

Метод	Описание метода	Описание использования
Get_Int	Функция возвращает числовое значение базового регистра (целое число). Входных параметров нет.	
Get_Flt	Функция возвращает числовое значение базового регистра (вещественное число). Входных параметров нет.	

Список объектов для интерпретации

INCT_BlockExecArray - интерфейс CAM системы, который реализует список объектов текущего кадра для интерпретации.

Описание методов интерфейса:

Метод	Описание метода	Описание использования
Add	Функция добавляет объект для интерпретации в конец списка объектов, возвращает индекс добавленного объекта (целое число). Входной параметр: ExecType (TNCExecObj) - тип объекта, добавляемого для интерпретации, ExecParam (указатель) параметр интерпретируемого объекта (описание см. ниже).	1. При анализе текста управляющей программы, выходящего за рамки типового анализа ядром CAM системы (реализация метода INCT_Interpreter.TransLine), распознанные лексемы и смысловые конструкции имеет смысл добавлять в список объектов (методом Add) для интерпретации с типом объект (поле ExecType = exObj), поле ExecParam заполнять индексом массива, либо указателем на соответствующую структуру хранящую данные об объекте для его последующей интерпретации. 2. При интерпретации очередной команды текущего списка INCT_BlockExecArray, зная, что текущая команда должна быть сынтерпретирована в конце кадра, возможно добавить её в список при помощи функции Add.
Insert	Функция добавляет объект для интерпретации в список объектов с номером Index, возвращает индекс	При трансляции текста управляющей программы можно добавлять оттранслированные объекты в начало списка

Метод	Описание метода	Описание использования
	добавленного объекта (целое число). Входные параметры: Index (целое число) - номер объекта в списке, ExecType (TNCExecObj) - тип объекта, добавляемого для интерпретации, ExecParam (указатель) параметр интерпретируемого объекта (описание см. ниже).	интерпретации, либо добавить объект после другого, нужного нам объекта, при помощи функции Insert, если этого требует процесс исполнения интерпретируемых объектов.
Get	Функция возвращает объект с номером Index из списка интерпретации (интерфейс INCT_BlockExecObject). Важно! Возвращаемый функцией результат будет корректным только до следующего изменения списка объектов INCT_BlockExecArray. Входные параметры: Index (целое число) - номер объекта в списке.	Функция Get востребована при интерпретации объектов списка. С помощью этой функции, по номеру объекта в списке можно получить интерфейс интерпретируемого объекта (INCT_BlockExecObject). Затем в зависимости от типа и назначения объекта выполнить действия по формированию траектории движения инструмента или по подготовке к таким действиям (сохранение значений, инициализация переменных и т.д.).
Find	Функция выполняет поиск объекта интерпретации в списке по параметрам ExecType, ExecParam, начиная с объекта под номером FromIdx, возвращает найденный объект	Функция Find может быть использована для определения наличия в уже оттранслированной части кадра регистра или объекта. Например:

Метод	Описание метода	Описание использования
	(интерфейс INCT_BlockExecObject). Важно! Возвращаемый функцией результат будет корректным только до следующего изменения списка объектов INCT_BlockExecArray. Входные параметры: ExecType (TNCExecObj) - тип интерпретируемого объекта; ExecParam (указатель) для регистра - номер базового регистра, для объекта - указатель, заполняемый интерпретатором; FromIdx (целое число) - начальный номер объекта в списке. Выходные параметры: Index (целое число) - номер объекта в списке.	1. С целью добавления после этого объекта нового объекта интерпретации (функция Insert). 2. Для определения типа регистра (при реализации INCT_Interpreter.DefineRegister). Пример. Для стойки Sinumerik в кадре управляющей программы 'G4 F100', регистр 'F' - выстой, а в кадре управляющей программы 'G01 X10 F100', регистр 'F' - подача. На этапе определения регистра, зная, что в оттранслированном кадре уже есть регистр G4, можно однозначно определить регистр 'F' это выстой, а не подача.
MoveTo	Функция возвращает результат перемещения объекта интерпретации с индексом FromIdx на место в списке с индексом ToIdx (логическое). Истина — успешно, Ложь — перемещение не произведено. Входные параметры: FromIdx (целое число) - индекс перемещаемого объекта в списке, ToIdx (целое число) - индекс места в списке, на	Функция MoveTo может быть использована для сортировки списка объектов для интерпретации после трансляции кадра. Например, у стойки Heidenhain существуют регистры, которые обрабатываются всегда в начале или в конце кадра, независимо от их расположения в тексте управляющей программы.

Метод	Описание метода	Описание использования
	которое перемещается объект.	
Get_Count	Функция возвращает количество объектов интерпретации в списке (целое число). Входных параметров нет.	Функция Get_Count может быть использована при сортировке списка для интерпретации, с целью определения количества элементов списка.

INCT_BlockExecObject - интерфейс CAM системы (возвращают функции INCT_BlockExecArray.Get, INCT_BlockExecArray.Find), который реализует объект для интерпретации. Объект создаётся функциями INCT_BlockExecArray.Add, INCT_BlockExecArray.Insert по переданным в них типу (ExecType) и параметру (ExecParam) объекта.

Описание методов интерфейса:

Метод	Описание метода	Описание использования
Get_ExecType	Функция возвращает тип интерпретируемого объекта (TNCExecObj) exReg - базовый регистр exObj - объект интерпретатора exProc - зарезервировано, ядром CAM системы не используется Входных параметров нет.	1. Функция Get_ExecType используется при интерпретации объектов списка. С помощью этой функции определяется тип объекта интерпретации. В зависимости от типа и параметра объекта интерпретации выполняются действия по формированию траектории движения инструмента или по подготовке к таким действиям (сохранение значений, инициализация переменных и т.д.). 2. Функция возвращает значение, заданное при

Метод	Описание метода	Описание использования
		добавлении объекта интерпретации в список.
Get_ExecParam	Функция возвращает параметр интерпретируемого объекта (указатель). Если интерпретируемый объект это регистр (exReg), то параметром будет номер базового регистра. Если интерпретируемый объект это объект интерпретатора, будет возвращён параметр, который был передан интерпретатором при добавлении объекта (INCT_BlockExecArray.Add, INCT_BlockExecArray.Insert) Входных параметров нет.	1. Функция Get_ExecParam используется при интерпретации объектов списка. С помощью этой функции определяется параметр объекта интерпретации, заполняемый интерпретатором. В зависимости от типа и параметра объекта интерпретации выполняются действия по формированию траектории движения инструмента или по подготовке к таким действиям (сохранение значений, инициализация переменных и т.д.). 2. Функция возвращает значение, заданное при добавлении объекта интерпретации в список.

Метод	Описание метода	Описание использования
Get_IsUsed	Функция возвращает признак 'сынтерпретирован' (IsUsed) объекта для интерпретации (логический). Истина — объект сынтерпретирован в текущем кадре, Ложь — объект не сынтерпретирован. Входных параметров нет.	Функция используется, чтобы проверить, интерпретирован ли объект в текущем кадре.
Set_IsUsed	Метод позволяет установить признак 'сынтерпретирован' (IsUsed) объекта для интерпретации. Входной параметр: Value (логический) - Истина - объект сынтерпретирован в текущем кадре, Ложь - объект не сынтерпретирован.	Функция может быть использована для принудительного изменения признака интерпретации объекта, в этом случае объект не будет сынтерпретирован.
Get_IsOwn	Функция возвращает признак 'собственный' (IsOwn) объекта для интерпретации типа exObj. Для объектов типа exReg всегда возвращает значение Ложь. Истина - объект добавлен интерпретатором, Ложь - объект добавлен наследником или родителем интерпретатора Входных параметров нет.	Функция может быть использована в интерпретаторе-наследнике, при принятии решения о самостоятельной интерпретации команды или передаче для интерпретации родителю.

Формирование траектории движения инструмента в осях станка

INCT_CLData - интерфейс CAM системы, который реализует команды формирования траектории движения инструмента, используя оси станка.

Описание методов интерфейса:

Метод	Описание метода	Описание использования
SetSystemUnits	Метод устанавливает единицы измерения проекта. Входной параметр: Units (TNCSystemMeasure) smMetrics - метрическая; smImperial - дюймовая.	Для интерпретации команд, аналогичных G20, G21 в ISO-кодах.
SetFeedrateUnits	Метод устанавливает единицы измерения подачи. Входной параметр: Units (TNCFeedrateMeasure) fmMMPerMinute - мм в минуту; fmInchPerMinute - дюймы в минуту.	Для интерпретации команд, аналогичных G700, G710 для стойки Sinumerik.
GetSystemUnits	Функция возвращает единицы измерения проекта (TNCSystemMeasure)	
GetFeedrateUnits	Функция возвращает текущие единицы измерения подачи (TNCFeedrateMeasure)	
AddOrigin	Метод добавляет в траекторию движения	Для интерпретации команд, аналогичных G54-G57 в ISO-кодах.

Метод	Описание метода	Описание использования
	команду выбора системы координат по номеру. Входной параметр: WCSNumber (вещественное число) номер системы координат.	
AddOriginByShift	Метод добавляет в траекторию движения инструмента команду смещения системы координат на заданную величину. Входной параметр: Shift (TNC3DPoint) - величина смещения по осям (X, Y, Z).	Для интерпретации команд, аналогичных G92 в ISO-кодах.
OutStandardFeed	Метод добавляет в траекторию движения инструмента команду изменения подачи на стандартную. Входной параметр: Feed (TNCFeedType) - тип подачи: ffWorking - рабочая; ffRapid - ускоренная; ffFirst - первого прохода; ffEngage - подвода; ffRetract - отхода; ffPlunge - врезания; ffFinish - чистового прохода; ffNext - перехода; ffReturn - возврата; ffApproach - подхода; ffRapid5D - ускоренная по	Для интерпретации команд, аналогичных G00 (Rapid) / G01 (Working) в ISO-кодах.

Метод	Описание метода	Описание использования
	безопасной поверхности с контролем пределов осей; ffTransitionOnSafe - перехода по безопасной поверхности; ffApproachFromSafe - подвода с безопасной поверхности; ffReturnToSafe - выхода на безопасную поверхность; ffLongNext - длинного перехода; ffThreadPitch - равная шагу резьбы. Значение подачи для каждого типа задаётся на вкладке 'Подачи' операции в CAM системе.	
OutFeed	Метод добавляет в траекторию движения инструмента команду изменения подачи с заданным типом и значением. Входные параметры: Feed (TNCFeedType) - тип подачи (см. OutStandardFeed), Value (вещественное число) - значение подачи, MPM(логическое) - признак единиц измерения подачи: Истина - мм в мин. Ложь - мм на оборот.	Для интерпретации команд, аналогичных F в ISO-кодах.
SetCurrentPlane	Метод добавляет в траекторию движения инструмента команду смены	Для интерпретации команд, аналогичных G17-G19 в ISO-кодах.

Метод	Описание метода	Описание использования
	текущей плоскости на заданную. Входной параметр: Plane (TNCPlaneType) - тип плоскости: ptXY - плоскость XY; ptYZ - плоскость YZ; ptZX - плоскость ZX.	
GetActiveSpindle	Функция возвращает номер активного шпинделя, нумерация начинается с 1.	
SetActiveSpindle	Метод делает активным шпиндель с заданным номером. Входной параметр: Number (целое) - номер шпинделя, нумерация начинается с 1.	Для интерпретации команд, аналогичных SETMS для стойки Sinumerik.
AddSpindleSpeedOnRPM	Метод добавляет в траекторию движения инструмента команду включения активного шпинделя с заданной частотой в оборотах в минуту. Если шпиндель уже включен, происходит изменение частоты вращения шпинделя на заданную. Входные параметры: RPM (вещественное число) - скорость вращения шпинделя в оборотах в	Для интерпретации команд, аналогичных S в ISO-кодах.

Метод	Описание метода	Описание использования
	минуту; Range (целое) - номер диапазона вращения шпинделя, Direction (логическое) - направление вращения (Истина - по часовой стрелке CW, Ложь - против часовой стрелки CCW).	
AddSpindleSpeedOnCSS	Метод добавляет в траекторию движения инструмента команду включения активного шпинделя с постоянной скоростью резания. Если шпиндель уже включен, происходит изменение режима и частоты вращения на заданную. Входные параметры: CSS (вещественное число) - значение постоянной скорости резания; MaxRPM (вещественное число) - максимальная частота вращения шпинделя; Range (целое) - номер диапазона вращения шпинделя, Direction (логическое) - направление вращения (Истина - по часовой стрелке CW, Ложь - против часовой стрелки CCW).	Для интерпретации команд, аналогичных G96 для стойки Sinumerik
AddSpindleOff	Метод добавляет в траекторию движения инструмента команду	Для интерпретации команд, аналогичных M05 в ISO-кодах.

Метод	Описание метода	Описание использования
	выключения активного шпинделя. Входных и выходных параметров нет.	
AddSpindleOrient	Метод добавляет в траекторию движения инструмента команду ориентированного останова активного шпинделя. Входной параметр: OrientationAngle (вещественное число) - угол поворота шпинделя.	
AddStop	Метод добавляет в траекторию движения инструмента команду прерывания выполнения управляющей программы. При достижении кадра останова исполнение управляющей программы прерывается. Входных и выходных параметров нет.	Для интерпретации команд, аналогичных M00 в ISO-кодах.
AddOpStop	Метод добавляет в траекторию движения инструмента команду опционального прерывания выполнения управляющей программы. При достижении кадра опционального останова исполнение	Для интерпретации команд, аналогичных M01 в ISO-кодах.

Метод	Описание метода	Описание использования
	программы прерывается только в том случае, если в стойке ЧПУ включен соответствующий режим. При выключении данного режима исполнение программы не прерывается. Входных и выходных параметров нет.	
AddCoolant	Метод добавляет в траекторию движения инструмента команду включение или выключения систем охлаждения станка. Входные параметры: OnOff (логический) - Истина - включение, Ложь - отключение; PipeNumber (целый) - Номер включаемого трубопровода охлаждения (1 – жидкость, 2 – туман, 3 – инструмент).	Для интерпретации команд, аналогичных M08/M09 в ISO-кодах.
AddDelay	Метод добавляет в траекторию движения инструмента команду приостановки выполнения программы на заданное время. Входной параметр: Value (вещественное число) - Время выстоя в секундах.	Для интерпретации команд, аналогичных G04 в ISO-кодах.

Метод	Описание метода	Описание использования
LoadTool	Метод меняет номер текущего инструмента на заданный. Входной параметр: ToolID (целое число) - номер инструмента.	Для интерпретации команд, аналогичных T01 в ISO-кодах.
AddLengthCompensation	Метод добавляет в траекторию движения инструмента команду компенсации на длину инструмента с заданными параметрами. Входные параметры: Mode (TNCLengthCompensationMode) - режим: lcOff - выключить компенсацию на длину инструмента; lcOn - включить положительную компенсацию на длину инструмента; lcOnNegative - включить отрицательную компенсацию на длину инструмента. OffsetNumber (целое число) - номер корректора на длину.	Для интерпретации команд, аналогичных G43, G44, G49, H в ISO-кодах.
AddLengthCompensationValue	Метод добавляет в траекторию движения инструмента команду компенсации на длину инструмента с заданными параметрами.	

Метод	Описание метода	Описание использования
	Входные параметры: Mode, OffsetNumber (см. AddLengthCompensation); Value (вещественное число) - значение коррекции длины инструмента. Внимание! В текущей версии функция работает аналогично AddLengthCompensation, без возможности задания значения.	
AddLengthCompensationWithSpecifiedAxis	Метод добавляет в траекторию движения инструмента команду компенсации на длину инструмента с заданными параметрами. Входные параметры: Mode (см. AddLengthCompensation); SpecifiedAxis (целое число) - номер оси, вдоль которой производится компенсация на длину; OffsetNumber (см. AddLengthCompensation). Внимание! В текущей версии функция работает аналогично AddLengthCompensation, без возможности выбора оси.	Для интерпретации команд, аналогичных G43 (вместе с G17-G19) в ISO-кодах.
AddRadiusCompensation	Метод добавляет в траекторию движения инструмента команду коррекции на радиус	Для интерпретации команд, аналогичных G41, G42, D в ISO-кодах.

Метод	Описание метода	Описание использования
	инструмента с заданными параметрами. Входные параметры: Mode (TNCRadiusCompensationMode) - режим: rcOff - выключить коррекцию на радиус инструмента; rcLeft - включить коррекцию на радиус инструмента слева; rcRight - включить коррекцию на радиус инструмента справа. OffsetNumber (целое число) - номер корректора на радиус.	
AddRadiusCompensationValue	Метод добавляет в траекторию движения инструмента команду коррекции на радиус инструмента с заданными параметрами. Входные параметры: Mode, OffsetNumber (см. AddRadiusCompensation); Value, Value2 (вещественное число) - значения коррекции радиуса инструмента. Внимание! В текущей версии функция работает аналогично AddRadiusCompensation, без возможности задания значения.	

Метод	Описание метода	Описание использования
OpenMultiMotion	Метод определяет начало ввода многокоординатных перемещений управляемых координат станка с заданным типом. Входной параметр: GotoMode (TNCGotoMode) - тип перемещения: gtmTooltip - перемещение относительно текущей системы координат заготовки, интерполяция базовых осей производится с контролем прямолинейности траектории перемещения в каждой из промежуточных точек; gtmPhysic - перемещение относительно машинных нулей (физические координаты станка), а при интерполяции базовых осей контроль промежуточных точек траектории не производится, контролируется лишь достижимость конечной точки.	Для интерпретации команд, аналогичных G00/G01 X Y Z A B C в ISO-кодах (gtmTooltip). Для интерпретации команд, аналогичных G53 в ISO, SUPA в Sinumerik, M91 в Heidenhain (gtmPhysic).
AddMotion	Метод добавляет перемещение по заданной оси в заданную координату. Входные параметры: AxisIndex (целое число) - индекс оси станка	

Метод	Описание метода	Описание использования
	(управляемой или физической), Value (вещественное число) - значение координаты. В качестве управляемых осей станка могут выступать как линейные, так и поворотные оси, а так же специфичные для конкретного станка (магазин инструментов, револьверная головка, патрон и т.п.).	
CloseMultiMotion	Метод определяет завершение ввода перемещений и добавляет в траекторию движения инструмента команду многокоординатных перемещений, сформированных в результате добавления перемещений командой AddMotion с начала ввода многокоординатных перемещений OpenMultiMotion.	
AddCircle	Метод добавляет в траекторию движения инструмента команду перемещения инструмента по дуге окружности с заданными параметрами. Входные параметры: P (TNC3DPoint) - конечная точка дуги (X, Y, Z); Pc (TNC3DPoint) -	Для интерпретации команд, аналогичных G02/G03 X Y Z I J K R в ISO-кодах.

Метод	Описание метода	Описание использования
	координата центра дуги (X, Y, Z); R (вещественное число) - радиус дуги (радиус больше нуля, если вращение против часовой стрелки, и радиус меньше нуля, если вращение по часовой стрелке); Plane(TNCPlaneType, см. **SetCurrentPlane**) - плоскость дуги; CanBeFull(логическое) - может быть замкнутой. Оси, связанные с параметрами X, Y, Z точек P, Pc определяются автоматически после выполнения команд смены инструмента (LoadTool) или установки активного шпинделя (SetActiveSpindle). Узнать индексы осей, связанных в настоящий момент с параметрами X, Y, Z можно при помощи функции GetXYZindex.	
OpenTooltipMotion	Метод определяет начало ввода многокоординатных перемещений с заданным типом. Входной параметр: Mode (TNCTooltipMode) - тип перемещения: tmLinear - линейное перемещение инструмента к заданной точке; tmCircularMiddlePoint,	Для интерпретации линейных перемещений и перемещений по дуге. Для линейных перемещений: • OpenTooltipMotion(tmLinear) • SetTooltipPosition • SetTooltipAttitude/SetTooltip Attitude2 • SetTooltipFlip (опционально) • AddTooltipMotion (опционально)

Метод	Описание метода	Описание использования
	tmCircularEndPoint - перемещение инструмента по дуге к заданной конечной точке через среднюю; tmPhysic - перемещение всеми физическими осями станка одновременно к заданной точке.	• CloseTooltipMotion Для перемещений по дуге: • OpenTooltipMotion(tmCircular MiddlePoint) • SetTooltipPosition • SetTooltipAttitude/ SetTooltipAttitude2 • SetTooltipFlip (опционально) • AddTooltipMotion (опционально) • CloseTooltipMotion • OpenTooltipMotion(tmCircular EndPoint) • SetTooltipPosition • SetTooltipAttitude/SetTooltip Attitude2 • SetTooltipFlip (опционально) • AddTooltipMotion (опционально) • CloseTooltipMotion Для перемещений всеми осями: • OpenTooltipMotion(tmPhysic) • SetTooltipPosition • SetTooltipAttitude/ SetTooltipAttitude2 • SetTooltipFlip (опционально) • AddTooltipMotion (опционально) • CloseTooltipMotion
SetTooltipPosition	Метод задаёт позицию определённой в OpenTooltipMotion точки в Декартовой системе координат относительно нуля детали. Входной параметр: P (TNC3DPoint) - координаты задаваемой точки.	
SetTooltipAttitude	Метод задаёт углы поворота вокруг осей X, Y, Z в Декартовой системе координат относительно нуля детали. Входные параметры: Ang1, Ang2, Ang3 (вещественное число) - заданные значения углов поворота Convention (TNCEulerAnglesConvention) - очерёдность выполнения угловых перемещений: eacXYZ - X, Y, Z eacXZY - X, Z, Y	

Метод	Описание метода	Описание использования
	eacYXZ - Y, X, Z eacYZX - Y, Z, X eacZXY - Z, X, Y eacZYX - Z, Y, X eacXYX - X, Y, X eacXZX - X, Z, X eacYXY - Y, X, Y eacYZY - Y, Z, Y eacZXZ - Z, X, Z eacZYZ - Z, Y, Z eacAxisAngle - вращение вокруг оси, направление которой задаётся нормированными значениями Ang1, Ang2, Ang3. Угол вращения вокруг оси равен модулю значений Ang1, Ang2, Ang3. RotationsAroundMovableAxes (логическое) - с поворотом вокруг подвижных осей. Истина - с поворотом. Ложь - без поворота. AnglesInDegrees (логическое) - Значение угловых перемещений задано в градусах. Истина - в градусах, Ложь - в радианах.	
SetTooltipAttitude2	Метод задаёт ориентацию инструмента кватернионом в Декартовой системе координат относительно нуля детали. Входные параметры:	

Метод	Описание метода	Описание использования
	qx, qy, qz, qw (вещественное число) - компоненты кватерниона	
SetTooltipFlip	Метод задаёт положение суставов (робота). Входные параметры: FlipType (TNCFlipType) - тип сустава ftFlipA1 - A1 ftFlipA3 - A3 ftFlipA5 - A5 Value (логическое) - Истина - Вывернутое положение, Ложь - Прямое положение.	
AddTooltipMotion	Метод добавляет перемещение по заданной дополнительной оси в заданную координату. Входные параметры: AxisIndex (целое число) - индекс дополнительной оси станка, Value (вещественное число) - значение координаты.	
CloseTooltipMotion	Метод определяет завершение ввода перемещений и добавляет в траекторию движения инструмента команду многокоординатных	

Метод	Описание метода	Описание использования
	перемещений, сформированных в результате добавления перемещений командами *Tooltip* с начала ввода перемещений командой OpenTooltipMotion.	
GetAxisValue	Функция возвращает координату текущего положения инструмента для заданной оси (вещественное число). Входной параметр: Index (целое число) - индекс оси.	Функция используется для получения текущего положения инструмента, например, при задании центра отверстия при использовании CreateDrillingCycleFormer .
WasAxisMovement	Функция возвращает признак наличия первого перемещения для заданной оси (логическое). Истина - первое перемещение уже было осуществлено, текущее положение инструмента можно получить используя GetAxisValue. Ложь - перемещений по заданной оси ещё не производилось. Входной параметр: Index (целое число) - индекс оси.	См. примечание к GetAxisValue.
AddSinglePassThread	Метод добавляет в траекторию движения инструмента команду активации режима непрерывного нарезания резьбы с постоянным шагом	Для интерпретации команд, аналогичных G33 в ISO-кодах.

Метод	Описание метода	Описание использования
	токарным резцом по заданным параметрам. Входные параметры: Orient (TNCThreadOrientation) - ориентация резьбы; LeadType (TNCThreadLeadType) - способ задания шага резьбы; Step (вещественное число) - значение шага резьбы; StartAngle (вещественное число) - начальное угловое положение шпинделя в градусах (используется для многозаходной резьбы). TNCThreadLeadType: ltDistance - расстоянием; ltThreadsPerUnit - количеством витков на единицу длины. TNCThreadOrientation: toOuter - наружная; toInner - внутренняя; toFace - торцевая.	
CreateLinearMotionFormer	Функция возвращает интерфейс формирователя траектории движения линейных перемещений (INCT_LinearMotionFormer). Использование формирователя в интерпретаторе упрощает построение линейной траектории движения инструмента. Работа с формирователем описана в	Для интерпретации команд, аналогичных G00/G01 X Y Z A B C в ISO-кодах.

Метод	Описание метода	Описание использования
	статье Линейные перемещения. Входных параметров нет.	
CreateCircularMotionFormer	Функция возвращает интерфейс формирователя траектории перемещений по дуге (INCT_CircularMotionFormer). Использование формирователя в интерпретаторе упрощает построение траектории движения инструмента по дуге. Работа с формирователем описана в статье Перемещения по дуге. Входных параметров нет.	Для интерпретации команд, аналогичных G02/G03 X Y Z I J K R в ISO-кодах.
CreateDrillingCycleFormer	В зависимости от заданного параметра функция возвращает интерфейс формирователя траектории одного из циклов сверления. Работа с формирователями описана в статьях Цикл сверления, Цикл сверления со сдвигом, Цикл сверления с ломкой стружки. Входной параметр: CycleType (TNCDrillingCycleType) - тип цикла. dctSimple - простой цикл сверления, в этом случае возвращаемый функцией	Для интерпретации команд, аналогичных G73, G74, G76, G80-89 для стойки Fanuc.

Метод	Описание метода	Описание использования
	формирователь поддерживает только один интерфейс: INCT_DrillingCycleFormer . dctWithShift - цикл сверления со сдвигом, возвращаемый функцией формирователь поддерживает два интерфейса: INCT_DrillingCycleFormer, INCT_DrillingCycleWithShiftFormer . dctPeck - цикл сверления с ломкой стружки, возвращаемый функцией формирователь поддерживает два интерфейса: INCT_DrillingCycleFormer, INCT_PeckDrillingCycleFormer .	
CreateLatheCycleFormer	В зависимости от заданного параметра функция возвращает интерфейс формирователя траектории одного из токарных циклов. Работа с формирователями описана в статьях Токарный цикл точения канавок, Токарный цикл резьбонарезания, Токарный цикл обработки резанием. Входной параметр: CycleType (TNCLatheCycleType) - тип цикла. lctGrooving - токарный цикл точения канавок, в этом	Для интерпретации команд, аналогичных CYCLE93, CYCLE95, CYCLE97 для стойки Sinumerik.

Метод	Описание метода	Описание использования
	случае возвращаемый функцией формирователь поддерживает один интерфейс: INCT_LatheGroovingCycleFormer . lctThreadCutting - токарный цикл резьбонарезания, в этом случае возвращаемый функцией формирователь поддерживает один интерфейс: INCT_LatheThreadCuttingCycleFormer . lctStockRemoval - токарный цикл чернового и чистового точения контура, в этом случае возвращаемый функцией формирователь поддерживает один интерфейс: INCT_LatheStockRemovalCycleFormer .	
OutPolarInterpolation	Метод включает или выключает полярную интерполяцию. Входной параметр: Enable (логический) - признак включения полярной интерполяции. True - интерполяция включена, False - интерполяция выключена.	Для интерпретации команд, аналогичных G132/G133 токарной стойки Okuma.
OutCylindricalInterpolation	Метод включает или выключает цилиндрическую	Для интерпретации команд, аналогичных G119 токарной

Метод	Описание метода	Описание использования
	интерполяцию с заданным радиусом. Входные параметры: Enable (логический) - признак включения цилиндрической интерполяции. True - интерполяция включена, False - интерполяция выключена. Radius (вещественное число) - заданный радиус.	стойки Okuma.
AddOriginByShiftAndRotate	Метод включает смещение и поворот системы координат согласно заданным параметрам. Входные параметры: Shift (TNC3DPoint) - величина смещения по осям (X, Y, Z). Ang1, Ang2, Ang3 (вещественное число) - заданные значения углов поворота Convention (TNCEulerAnglesConvention) - очерёдность выполнения угловых перемещений. Описание см. SetTooltipAttitude. RotationsAroundMovableAxes (логическое) - с поворотом вокруг подвижных осей. Истина - с поворотом. Ложь - без поворота.	Для интерпретации команд, аналогичных G68.2, G53.1 стойки Fanuc.

Метод	Описание метода	Описание использования
	AnglesInDegrees (логическое) - Значение угловых перемещений задано в градусах. Истина - в градусах, Ложь - в радианах.	
ResetOrigin	Метод выключает смещение и поворот системы координат, заданный в AddOriginByShiftAndRotate.	
SetRTCPMode	Метод добавляет в траекторию движения инструмента команду включения или выключения режима RTCP - пятикоординатной интерполяции с управлением центром инструмента. Входной параметр: Enable (логический) - Истина - режим включён, Ложь - режим выключен.	Для интерпретации команд, аналогичных G43.4/G49 стойки Fanuc, TRAORI/TRAFOOF стойки Sinumerik, M128/M129 стойки Heidenhain.

INCT_AddOriginByShiftAndRotate - интерфейс CAM системы, который включает смещение и поворот системы координат с выбором решения для поворотных осей (параметр FlipMode)

Описание методов интерфейса:

Метод	Описание метода	Описание использования
AddOriginByShiftAndRotate	Метод включает смещение и поворот системы координат согласно заданным параметрам. Входные параметры: Shift (TNC3DPoint) - величина	Для интерпретации команд, аналогичных G68.2, G53.1 стойки Fanuc.

Метод	Описание метода	Описание использования
	смещения по осям (X, Y, Z). Ang1, Ang2, Ang3 (вещественное число) - заданные значения углов поворота Convention (TNCEulerAnglesConvention) - очерёдность выполнения угловых перемещений. Описание см. SetTooltipAttitude. RotationsAroundMovableAxes (логическое) - с поворотом вокруг подвижных осей. Истина - с поворотом. Ложь - без поворота. AnglesInDegrees (логическое) - Значение угловых перемещений задано в градусах. Истина - в градусах, Ложь - в радианах. PositioningMode (TNCSPositioningMode) - тип позиционирования осей станка локальной СК cspmStay - локальная СК не двигает оси станка cspmTurn - локальная СК позиционирует только поворотные оси станка cspmMove - локальная СК позиционирует все оси станка так, чтобы положение кончика инструмента по отношению к детали не изменилось FlipMode (TNCAddOriginFlipMode) - решение для положения	

Метод	Описание метода	Описание использования
	инструмента относительно заготовки, если станок позволяет обеспечить одинаковое положение несколькими способами: ofmNearest - ближайшее ofmDirect - прямое ofmAlternate - альтернативное	

INCT_CLData2 - интерфейс CAM системы, который реализует команды формирования траектории движения инструмента, используя оси станка

Описание методов интерфейса:

Метод	Описание метода	Описание использования
CreateMillingCycleFormer	Функция возвращает интерфейс формирователя траектории фрезерного цикла. Работа с формирователями описана в статье Цикл фрезерования резьбы. Входной параметр: CycleType (TNCMillingCycleType) - тип цикла. mctThreadMilling - фрезерования резьбы, в этом случае возвращаемый функцией формирователь поддерживает интерфейс: INCT_ThreadMillingCycleFormer.	Для интерпретации команд, аналогичных CYCLE90 для стойки Sinumerik.

INCT_CLData_CutTo5d - интерфейс CAM системы, который реализует команды формирования траектории движения инструмента по векторам нормали

Описание методов интерфейса:

Метод	Описание метода	Описание использования
CutTo5d	Используется для перемещений по векторам нормали Входные параметры: P (TNC3DPoint) - Координаты (X, Y, Z), Вещественные числа N (TNC3DPoint) - Вектора нормали от (-1 до 1), Вещественные числа	Для интерпретации команд, аналогичных TRAORI для стойки Sinumerik по векторам нормали A3 B3 C3.

INCT_CLData_AutoAxesBrakes - интерфейс CAM системы, который реализует команды включения/отключения автоматического вывода тормоза осей

Описание методов интерфейса:

Метод	Описание метода	Описание использования
Get_AutoAxesBrakes	Получение информации о состоянии тормоза осей Выходные параметры: Value (логическое) - автоматический вывод тормоза оси. Ложь - автоматический вывод тормоза оси отключен. Истина - автоматический вывод тормоза оси включен.	Используется когда нет необходимости выводить в CLData информацию о включении или отключении тормоза осей.
Set_AutoAxesBrakes	Включение/отключение автоматического вывода тормоза осей Входной параметр: Value (логическое) - автоматический вывод тормоза оси. Ложь - отключить автоматический вывод тормоза оси. Истина -	

Метод	Описание метода	Описание использования
	включить автоматический вывод тормоза оси.	

INCT_CLData_Comment - Интерфейс CAM системы, позволяющий добавлять собственные комментарии к CLData.

Описание методов интерфейса:

Метод	Описание метода	Описание использования
AddComment	Метод добавляет строку комментария к CLData. Входной параметр: Value (строка) - текст комментария.	Добавление пояснений к отдельным командам или группам команд. Внесение служебной информации для технологов или операторов станков.

Интерфейсы, описанные ниже, запрашиваются у объекта INCT_CLData. Результат запроса нужно проверять: в более ранних версиях CAM системы и в отдельных режимах её работы объект может не поддерживать интерфейс.

INCT_CLData_BreakCommand - интерфейс CAM системы, который реализует команду включения/отключения тормозов осей станка.

Описание методов интерфейса:

Метод	Описание метода	Описание использования
Open	Метод определяет начало ввода команды тормозов осей. Входных и выходных параметров нет.	Для интерпретации команд зажима и разжима осей, например M10/M11. Порядок вызова: Open, затем SetBreak для каждой оси, затем Close.
SetBreak	Метод задаёт состояние тормоза оси в команде. Входные параметры: AxisIndex (целое число) -	Обычно используется вместе с отключением автоматического вывода тормозов осей (Set_AutoAxesBrakes

Метод	Описание метода	Описание использования
	индекс оси станка; Value (логическое) - Истина - тормоз включён, Ложь - тормоз выключен.	интерфейса INCT_CLData_AutoAxesBrakes).
Close	Метод определяет завершение ввода и добавляет в траекторию движения инструмента команду тормозов осей, заданных методом SetBreak с начала ввода командой Open. Если не задана ни одна ось, команда не добавляется. Входных и выходных параметров нет.	

INCT_CLData_Insert - интерфейс CAM системы, который позволяет добавлять в CLData команду INSERT с произвольным текстом.

Описание методов интерфейса:

Метод	Описание метода	Описание использования
AddInsert	Метод добавляет в траекторию движения инструмента команду INSERT с заданным текстом. Входной параметр: Value (строка) - текст команды.	Передача постпроцессору текста, который CAM система не моделирует, например вспомогательных команд станка. Как текст попадёт в управляющую программу, определяет постпроцессор.

INCT_CLData_Print - интерфейс CAM системы, который позволяет добавлять в CLData команду PPRINT с произвольным текстом. Интерфейс поддерживают объекты INCT_CLData и INCT_GeomCLData.

Описание методов интерфейса:

Метод	Описание метода	Описание использования
AddPrint	Метод добавляет в траекторию движения инструмента команду PPRINT с заданным текстом. Входной параметр: Value (строка) - текст команды.	Для интерпретации команд, аналогичных PPRINT языка APT.

INCT_CLData_Wait - интерфейс CAM системы, который реализует синхронизацию каналов многоканального станка.

Описание методов интерфейса:

Метод	Описание метода	Описание использования
AddWait	Метод добавляет в траекторию движения инструмента точку синхронизации каналов с заданным номером. Каналы, в которых встретилась точка синхронизации с одинаковым номером, ожидают друг друга. Входной параметр: Id (целое число) - номер точки синхронизации.	Для интерпретации команд ожидания в многоканальных программах, аналогичных WAITM стойки Sinumerik.

INCT_CLData_LngID - интерфейс CAM системы, который возвращает язык интерфейса CAM системы.

Описание методов интерфейса:

Метод	Описание метода	Описание использования
GetLngID	Функция возвращает код языка интерфейса CAM системы (целое число) - код языка Windows, например 1049 - русский, 1033 -	Для вывода сообщений интерпретатора на языке интерфейса CAM системы.

Метод	Описание метода	Описание использования
	английский. Входных параметров нет.	

INCT_AddOriginByPhysicAxes - интерфейс CAM системы, который реализует задание системы координат по положению осей станка.

Описание методов интерфейса:

Метод	Описание метода	Описание использования
AddOriginByPhysicAxes	Метод включает смещение и поворот системы координат, заданный положением осей станка. Оси с индексами AxisIndex1-AxisIndex3 в расчёте ставятся в значения Ang1-Ang3: ориентация новой системы координат совпадает с ориентацией инструмента в этом положении станка, а её начало смещено на Shift относительно текущей системы координат детали. Перемещение осей станка определяет параметр PositioningMode. Входные параметры: Shift (TNC3DPoint) - смещение начала системы координат по осям (X, Y, Z); AxisIndex1, AxisIndex2, AxisIndex3 (целое число) - индексы осей станка, -1 - ось не задана; Ang1, Ang2, Ang3 (вещественное число) - значения осей AxisIndex1-AxisIndex3 в тех же единицах, что значение координаты в	Для интерпретации команд, аналогичных PLANE AXIAL стойки Heidenhain.

Метод	Описание метода	Описание использования
	AddMotion; PositioningMode (TNCSPositioningMode) - тип позиционирования осей станка локальной СК, описание см. INCT_AddOriginByShiftAndRotate . Если не задана ни одна ось, команда не добавляется.	

INCT_OriginSolver - интерфейс CAM системы, который позволяет заранее рассчитать положения осей станка для поворота системы координат, не меняя состояния станка и не добавляя команд в траекторию движения инструмента.

Описание методов интерфейса:

Метод	Описание метода	Описание использования
CalculateOriginSolutions	Метод рассчитывает положения осей станка для системы координат, заданной смещением и поворотом. Решения рассчитываются сразу для трёх режимов TNCAddOriginFlipMode: ofmDirect - прямое, ofmAlternate - альтернативное, ofmNearest - ближайшее из возможных. Результаты возвращают функции GetOriginSolutionState и GetOriginSolutionAxisValue. Входные параметры: Shift, Ang1, Ang2, Ang3, Convention, RotationsAroundMovableAxes, AnglesInDegrees - как у	Для интерпретации команд, аналогичных PLANE SPATIAL ... SEQ+/SEQ- стойки Heidenhain: позволяет заранее выбрать решение по знаку главной поворотной оси и проверить, достижимо ли оно, а затем передать выбранный режим в INCT_AddOriginByShiftAndRotate .AddOriginByShiftAndRotate (параметр FlipMode).

Метод	Описание метода	Описание использования
	метода AddOriginByShiftAndRotate.	
GetOriginSolutionState	Функция возвращает состояние решения для заданного режима (TNCOriginSolutionState): ossNotCalculated - решение не рассчитывалось; ossCalcError - решения нет; ossOutOfLimits - решение выходит за пределы осей станка; ossReachable - решение достижимо. Входной параметр: FlipMode (TNCAddOriginFlipMode) - режим.	
GetOriginSolutionAxisValue	Функция возвращает значение оси станка в решении для заданного режима (вещественное число). Если ось в решении не участвует, возвращается 0. Входные параметры: FlipMode (TNCAddOriginFlipMode) - режим; AxisIndex (целое число) - индекс оси станка.	
GetFirstRotaryAxisAfterCurrentTool	Функция возвращает индекс поворотной оси, ближайшей к текущему инструменту в кинематической цепочке	

Метод	Описание метода	Описание использования
	станка (целое число). Шпиндели не учитываются. Если такой оси нет, возвращается -1. Входных параметров нет.	
GetLastRotaryAxisFromCurrentWorkpiece	Функция возвращает индекс поворотной оси на стороне текущей заготовки, наиболее удалённой от неё в кинематической цепочке станка (целое число). Шпиндели не учитываются. Если такой оси нет, возвращается -1. Входных параметров нет.	

Формирование траектории движения инструмента в геометрических осях

INCT_GeomCLData - интерфейс CAM системы, который реализует команды формирования траектории движения инструмента, используя геометрические оси.

Описание методов интерфейса:

Метод	Описание метода	Описание использования
SetSystemUnits	Метод устанавливает единицы измерения проекта. Входной параметр: Units (TNCSystemMeasure) smMetrics - метрическая; smImperial - дюймовая.	Для интерпретации команд, аналогичных G20, G21 в ISO-кодах.
SetFeedrateUnits	Метод устанавливает единицы измерения подачи. Входной параметр: Units (TNCFeedrateMeasure) fmMMPerMinute - мм в минуту; fmInchPerMinute - дюймы в минуту.	Для интерпретации команд, аналогичных G700, G710 для стойки Sinumerik.
GetSystemUnits	Функция возвращает единицы измерения проекта (TNCSystemMeasure)	
GetFeedrateUnits	Функция возвращает текущие единицы измерения подачи (TNCFeedrateMeasure)	
OutStandardFeed	Метод добавляет в траекторию движения инструмента команду	Для интерпретации команд, аналогичных G00 (Rapid) / G01 (Working) в ISO-кодах.

Метод	Описание метода	Описание использования
	изменения подачи на стандартную. Входной параметр: Feed (TNCFeedType) - тип подачи: ffWorking - рабочая; ffRapid - ускоренная; ffFirst - первого прохода; ffEngage - подвода; ffRetract - отхода; ffPlunge - врезания; ffFinish - чистового прохода; ffNext - перехода; ffReturn - возврата; ffApproach - подхода; ffRapid5D - ускоренная по безопасной поверхности с контролем пределов осей; ffTransitionOnSafe - перехода по безопасной поверхности; ffApproachFromSafe - подвода с безопасной поверхности; ffReturnToSafe - выхода на безопасную поверхность; ffLongNext - длинного перехода; ffThreadPitch - равная шагу резьбы. Значение подачи для каждого типа задаётся на вкладке 'Подачи' операции в CAM системе.	
OutFeed	Метод добавляет в траекторию движения инструмента команду изменения подачи с	Для интерпретации команд, аналогичных F в ISO-кодах.

Метод	Описание метода	Описание использования
	заданным типом и значением. Входные параметры: Feed (TNCFeedType) - тип подачи (см. OutStandardFeed), Value (вещественное число) - значение подачи, MPM (логическое) - признак единиц измерения подачи: Истина - мм в мин. Ложь - мм на оборот.	
SetCurrentPlane	Метод добавляет в траекторию движения инструмента команду смены текущей плоскости на заданную. Входной параметр: Plane (TNCPlaneType) - тип плоскости: ptXY - плоскость XY; ptYZ - плоскость YZ; ptZX - плоскость ZX.	Для интерпретации команд, аналогичных G17-G19 в ISO-кодах.
AddSpindleSpeedOnRPM	Метод добавляет в траекторию движения инструмента команду включения активного шпинделя с заданной частотой в оборотах в минуту. Если шпиндель уже включен, происходит изменение частоты вращения шпинделя на заданную.	Для интерпретации команд, аналогичных S в ISO-кодах.

Метод	Описание метода	Описание использования
	Входные параметры: RPM (вещественное число) - скорость вращения шпинделя в оборотах в минуту; Range (целое число) - номер диапазона вращения шпинделя, Direction (логическое) - направление вращения (Истина - по часовой стрелке CW, Ложь - против часовой стрелки CCW).	

Метод	Описание метода	Описание использования
AddSpindleSpeedOnCSS	Метод добавляет в траекторию движения инструмента команду включения активного шпинделя с постоянной скоростью резания. Если шпиндель уже включен, происходит изменение режима и частоты вращения на заданную. Входные параметры: CSS (вещественное число) - значение постоянной скорости резания; MaxRPM (вещественное число) - максимальная частота вращения шпинделя; Range (целое число) - номер диапазона вращения шпинделя, Direction (логическое) - направление вращения (Истина - по часовой стрелке CW, Ложь - против часовой стрелки CCW).	Для интерпретации команд, аналогичных G96 для стойки Sinumerik
AddSpindleOff	Метод добавляет в траекторию движения инструмента команду выключения активного шпинделя. Входных и выходных параметров нет.	Для интерпретации команд, аналогичных M05 в ISO-кодах.
AddSpindleOrient	Метод добавляет в траекторию движения	

Метод	Описание метода	Описание использования
	инструмента команду ориентированного останова активного шпинделя. Входной параметр: OrientationAngle (вещественное число) - угол поворота шпинделя.	
AddOpStop	Метод добавляет в траекторию движения инструмента команду опционального прерывания выполнения управляющей программы. При достижении кадра опционального останова исполнение программы прерывается только в том случае, если в стойке ЧПУ включен соответствующий режим. При выключении данного режима исполнение программы не прерывается. Входных и выходных параметров нет.	Для интерпретации команд, аналогичных M01 в ISO-кодах.
AddCoolant	Метод добавляет в траекторию движения инструмента команду включение или выключения систем охлаждения станка. Входные параметры: OnOff (логический) - Истина - включение, Ложь - отключение; PipeNumber (целое число) - Номер	Для интерпретации команд, аналогичных M08/M09 в ISO-кодах.

Метод	Описание метода	Описание использования
	включаемого трубопровода охлаждения (1 – жидкость, 2 – туман, 3 – инструмент).	
AddDelay	Метод добавляет в траекторию движения инструмента команду приостановки выполнения программы на заданное время. Входной параметр: Value (вещественное число) - Время выстоя в секундах.	Для интерпретации команд, аналогичных G04 в ISO-кодах.
AddLengthCompensation	Метод добавляет в траекторию движения инструмента команду компенсации на длину инструмента с заданными параметрами. Входные параметры: Mode (TNCLengthCompensationMode) - режим: lcOff - выключить компенсацию на длину инструмента; lcOn - включить положительную компенсацию на длину инструмента; lcOnNegative - включить отрицательную компенсацию на длину инструмента. OffsetNumber (целое число) - номер корректора на длину.	Для интерпретации команд, аналогичных G43, G44, G49, H в ISO-кодах.

Метод	Описание метода	Описание использования
AddLengthCompensationValue	Метод добавляет в траекторию движения инструмента команду компенсации на длину инструмента с заданными параметрами. Входные параметры: Mode, OffsetNumber (см. AddLengthCompensation); Value (вещественное число) - значение коррекции длины инструмента. Внимание! В текущей версии функция работает аналогично AddLengthCompensation, без возможности задания значения.	
AddLengthCompensationWithSpecifiedAxis	Метод добавляет в траекторию движения инструмента команду компенсации на длину инструмента с заданными параметрами. Входные параметры: Mode (см. AddLengthCompensation); SpecifiedAxis (целое число) - номер оси, вдоль которой производится компенсация на длину; OffsetNumber (см. AddLengthCompensation). Внимание! В текущей версии функция работает аналогично	Для интерпретации команд, аналогичных G43 (вместе с G17-G19) в ISO-кодах.

Метод	Описание метода	Описание использования
	AddLengthCompensation, без возможности выбора оси.	
AddRadiusCompensation	Метод добавляет в траекторию движения инструмента команду коррекции на радиус инструмента с заданными параметрами. Входные параметры: Mode (TNCRadiusCompensationMode) - режим: rcOff - выключить коррекцию на радиус инструмента; rcLeft - включить коррекцию на радиус инструмента слева; rcRight - включить коррекцию на радиус инструмента справа. OffsetNumber (целое число) - номер корректора на радиус.	Для интерпретации команд, аналогичных G41, G42, D в ISO-кодах.
AddRadiusCompensationValue	Метод добавляет в траекторию движения инструмента команду коррекции на радиус инструмента с заданными параметрами. Входные параметры: Mode, OffsetNumber (см. AddRadiusCompensation); Value, Value2 (вещественное число) - значения коррекции радиуса инструмента.	

Метод	Описание метода	Описание использования
	Внимание! В текущей версии функция работает аналогично AddRadiusCompensation, без возможности задания значения.	
CutTo3d	Метод добавляет в траекторию движения инструмента команду линейного перемещения инструмента в заданную точку. Входной параметр: P (TNC3DPoint) - координаты заданной точки	Для интерпретации команд, аналогичных G00/G01 X Y Z (CutTo3d) или G00/G01 X Y Z A B C (CutTo5d, CutTo6d) в ISO-кодах
CutTo5d	Метод добавляет в траекторию движения инструмента команду линейного перемещения инструмента в заданную точку с заданным направлением положения инструмента. Входные параметры: P (TNC3DPoint) - координаты заданной точки; N (TNC3DPoint) - вектор нормали инструмента в системе координат детали.	
CutTo6d	Метод добавляет в траекторию движения инструмента команду линейного перемещения	

Метод	Описание метода	Описание использования
	инструмента в заданное положение с заданной ориентацией инструмента. Входной параметр: P (TNC3DMatrix) - положение и ориентация инструмента в системе координат детали. TNC3DMatrix - однородная матрица 4×4 с полями vX, A, vY, B, vZ, C, vT, D: vX, vY, vZ (TNC3DPoint) - направления осей системы координат инструмента (vZ - ось инструмента); vT (TNC3DPoint) - положение инструмента (X, Y, Z); A, B, C (вещественное число) - 0; D (вещественное число) - 1.	
ArcTo2d	Метод добавляет в траекторию движения инструмента команду перемещения инструмента по дуге окружности с заданными параметрами. Входные параметры: Pe (TNC3DPoint) - конечная точка дуги (X, Y, Z); Pc (TNC3DPoint) - координата центра дуги (X, Y, Z); Plane (TNCPlaneType, см. SetCurrentPlane) - плоскость дуги; R (вещественное число) - радиус дуги (радиус больше нуля, если вращение против часовой стрелки, и радиус меньше нуля, если вращение	Для интерпретации команд, аналогичных G02/G03 X Y Z I J K R в ISO-кодах

Метод	Описание метода	Описание использования
	по часовой стрелке); CanBeFull (логическое) - может быть замкнутой.	
ArcTo3d	Метод добавляет в траекторию движения инструмента команду перемещения инструмента по дуге окружности с заданными параметрами. Входные параметры: Pe (TNC3DPoint) - конечная точка дуги (X, Y, Z); Pc (TNC3DPoint) - координата центра дуги (X, Y, Z); Nc (TNC3DPoint) - вектор нормали плоскости, в которой лежит дуга; R (вещественное число) - радиус дуги (радиус больше нуля, если вращение против часовой стрелки, и радиус меньше нуля, если вращение по часовой стрелке); CanBeFull (логическое) - может быть замкнутой.	
ArcTo5d	Метод добавляет в траекторию движения инструмента команду перемещения инструмента по дуге окружности с заданными параметрами. Входные параметры: Pe (TNC3DPoint) - конечная точка дуги (X, Y, Z); Ne (TNC3DPoint) - вектор нормали инструмента в конечной	

Метод	Описание метода	Описание использования
	точке в системе координат детали; Pc (TNC3DPoint) - координата центра дуги (X, Y, Z); Nc (TNC3DPoint) - вектор нормали плоскости, в которой лежит дуга; R (вещественное число) - радиус дуги (радиус больше нуля, если вращение против часовой стрелки, и радиус меньше нуля, если вращение по часовой стрелке); CanBeFull (логическое) - может быть замкнутой.	
ArcTo6d	Метод добавляет в траекторию движения инструмента команду перемещения инструмента по дуге окружности с заданными параметрами. Входные параметры: Pe (TNC3DMatrix) - положение и ориентация инструмента в конечной точке дуги в системе координат детали (описание типа см. CutTo6d); Pc (TNC3DPoint) - координата центра дуги (X, Y, Z); Nc (TNC3DPoint) - вектор нормали плоскости, в которой лежит дуга; R (вещественное число) - радиус дуги (радиус больше нуля, если вращение против часовой стрелки, и радиус меньше нуля, если вращение по часовой стрелке);	

Метод	Описание метода	Описание использования
	CanBeFull (логическое) - может быть замкнутой.	
GetLastPosition	Функция возвращает координаты текущего положения инструмента и вектор нормали инструмента относительно системы координат детали. Входных параметров нет. Выходные параметры: P (TNC3DPoint) - текущее положение инструмента; N (TNC3DPoint) - вектор нормали инструмента. Примечание: до первого перемещения положение инструмента не определено, поэтому перед вызовом нужно проверить WasMovement.	Функция используется для получения текущего положения инструмента.
WasMovement	Функция возвращает признак наличия первого перемещения (логическое). Истина - первое перемещение уже было осуществлено, текущее положение инструмента можно получить используя GetLastPosition. Ложь - перемещений ещё не производилось. При моделировании по управляющей программе (в	См. примечание к GetLastPosition.

Метод	Описание метода	Описание использования
	том числе в NC Тюнере) функция всегда возвращает Истина.	
CreateLinearMotionFormer	Функция возвращает интерфейс формирователя траектории движения линейных перемещений (INCT_GeomLinearMotionFormer). Использование формирователя в интерпретаторе упрощает построение линейной траектории движения инструмента. Работа с формирователем описана в статье Линейные перемещения. Входных параметров нет.	Для интерпретации команд, аналогичных G00/G01 X Y Z в ISO-кодах.
CreateCircularMotionFormer	Функция возвращает интерфейс формирователя траектории перемещений по дуге (INCT_CircularMotionFormer). Использование формирователя в интерпретаторе упрощает построение траектории движения инструмента по дуге. Работа с формирователем описана в статье Перемещения по дуге. Входных параметров нет.	Для интерпретации команд, аналогичных G02/G03 X Y Z I J K R в ISO-кодах.

Метод	Описание метода	Описание использования
CreateDrillingCycleFormer	В зависимости от заданного параметра функция возвращает интерфейс формирователя траектории одного из циклов сверления. Работа с формирователями описана в статьях Цикл сверления, Цикл сверления со сдвигом, Цикл сверления с ломкой стружки. Входной параметр: CycleType (TNCDrillingCycleType) - тип цикла. dctSimple - простой цикл сверления, в этом случае возвращаемый функцией формирователь поддерживает только один интерфейс: INCT_DrillingCycleFormer . dctWithShift - цикл сверления со сдвигом, возвращаемый функцией формирователь поддерживает два интерфейса: INCT_DrillingCycleFormer, INCT_DrillingCycleWithShiftFormer. dctPeck - цикл сверления с ломкой стружки, возвращаемый функцией формирователь поддерживает два интерфейса: INCT_DrillingCycleFormer, INCT_PeckDrillingCycleFormer.	Для интерпретации команд, аналогичных G73, G74, G76, G80-89 для стойки Fanuc.

Метод	Описание метода	Описание использования
CreateLatheCycleFormer	В зависимости от заданного параметра функция возвращает интерфейс формирователя траектории одного из токарных циклов. Работа с формирователями описана в статьях Токарный цикл точения канавок, Токарный цикл резьбонарезания, Токарный цикл обработки резанием. Входной параметр: CycleType (TNCLatheCycleType) - тип цикла. lctGrooving - токарный цикл точения канавок, в этом случае возвращаемый функцией формирователь поддерживает один интерфейс: INCT_LatheGroovingCycleFormer . lctThreadCutting - токарный цикл резьбонарезания, в этом случае возвращаемый функцией формирователь поддерживает один интерфейс: INCT_LatheThreadCuttingCycleFormer . lctStockRemoval - токарный цикл чернового и чистового точения контура, в этом случае возвращаемый функцией формирователь поддерживает один интерфейс:	Для интерпретации команд, аналогичных CYCLE93, CYCLE95, CYCLE97 для стойки Sinumerik.

Метод	Описание метода	Описание использования
	INCT_LatheStockRemovalCycleFormer .	

Параметры оборудования

INCT_Equipment - интерфейс CAM системы, который реализует доступ к некоторым параметрам оборудования.

Описание методов интерфейса:

Метод	Описание метода	Описание использования
GetAxisIndex	Функция возвращает индекс оси станка по её адресу (целое число). Входной параметр: Addr (строка) - адрес оси ('X', 'Y', 'Z' и т.д. так, как указано в кинематической схеме станка).	Для управления перемещением инструмента (при помощи INCT_CLData.AddMotion или CreateLinearMotionFormer) необходимо задавать индексы осей, которые возвращает функция GetAxisIndex.
GetAxisIndexByNum	Функция возвращает индекс основной или дополнительной оси робота по её номеру (целое число). Входной параметр: AxisNum (целое число) - номер оси робота (1-6). Ext (логическое): Ложь — ось робота, Истина — дополнительная ось.	Если ось робота задаётся номером (не адресом), то для получения индекса оси можно воспользоваться функцией GetAxisIndexByNum. Как правило, оси роботов Motoman можно получить таким способом.

Метод	Описание метода	Описание использования
GetXYZindex	Функция возвращает индекс оси, связанной с перемещением по координатам X или Y или Z по заданному параметру (см. INCT_CLData.AddCircle). Входной параметр: Ort (целое число) 1 - X; 2 - Y; 3 - Z.	Для получения индексов осей, связанных с X, Y, Z текущего инструмента.
CurrentLatheToolWidth	Функция возвращает ширину текущего токарного инструмента (вещественное число).	Используется для задания шага ширины канавки в INCT_LatheGroovingCycleFormer .
GetSpindlesCount	Функция возвращает количество шпинделей текущего станка (целое число).	Для выполнения команд управляющих шпинделем станка по номеру. Например, SPOS[n] = Value для стойки Sinumerik: номера шпинделей в УП идут с 1, а индексы шпинделей - с 0, поэтому ось шпинделя n - GetAxisIndexBySpindleIndex(n - 1).
GetAxisIndexBySpindleIndex	Функция возвращает номер оси шпинделя по индексу шпинделя (целое число). Входной параметр: SpindleIndex (целое число) - индекс шпинделя, возможные значения от 0 до GetSpindlesCount -1.	
GetOperationGroup	Функция возвращает тип операции в CAM системе (TNCOperationGroup). Тип TNCOperationGroup: ogUnknown - неизвестная ogMill - фрезерная	Следует использовать при разработке интерпретаторов только в крайних случаях. Функция введена для компенсации положения токарного инструмента в

Метод	Описание метода	Описание использования
	ogLathe - токарная ogAuxiliary - вспомогательная ogWireEDM - электроэрозионная ogWelding - разного рода сварка ogAdditive - различные аддитивные операции типа наплавки ogWaterJet - гидроабразивная резка ogSawing - распиловка	токарной операции CAM системы, без возможности задания значения корректора.

INCT_Equipment2 - интерфейс CAM системы.

Описание методов интерфейса:

Метод	Описание метода	Описание использования
GetCurrentToolRadius	Функция возвращает радиус текущего инструмента (вещественное число)	Может использоваться для расчёта перебегов для CYCLE90 в Sinumerik

INCT_Equipment_Channels - интерфейс CAM системы, который реализует доступ к каналам станка. Интерфейс запрашивается у объекта INCT_Equipment; результат запроса нужно проверять: в более ранних версиях CAM системы объект может его не поддерживать.

Описание методов интерфейса:

Метод	Описание метода	Описание использования
Get_Count	Функция возвращает количество каналов станка (целое число).	
Get_CurrentIndex	Функция возвращает индекс канала, к которому относится текущая операция (целое число). Нумерация начинается с 0.	Для интерпретации многоканальных программ: позволяет определить, программу какого канала

Метод	Описание метода	Описание использования
		интерпретирует интерпретатор.
Get_Channel	Функция возвращает интерфейс канала (INCT_Channel) по индексу. Входной параметр: Index (целое число) - индекс канала, возможные значения от 0 до Get_Count - 1.	

INCT_Channel - интерфейс CAM системы, который реализует доступ к параметрам канала станка.

Описание методов интерфейса:

Метод	Описание метода	Описание использования
Get_Name	Функция возвращает имя канала, заданное в кинематической схеме станка (строка).	

Управление анализом текста управляющей программы

INCT_Analyzer - интерфейс CAM системы, который реализует доступ к управлению настройками анализа текста управляющей программы. Настройки анализатора влияют на алгоритм определения регистров и значений в тексте управляющей программы.

Описание методов интерфейса:

Метод	Описание метода	Описание использования
BeginUpdate	Сигнализирует о начале операции обновления INCT_Analyzer. Входных и выходных параметров нет.	Вызывайте BeginUpdate для запуска операции изменений настроек INCT_Analyzer. После завершения изменений настроек вызовите EndUpdate для применения всех произведённых изменений. Каждый вызов BeginUpdate должен сопровождаться соответствующим вызовом метода EndUpdate. Если за один раз производится изменение только одной из настроек INCT_Analyzer, использование BeginUpdate, EndUpdate необязательно.
EndUpdate	Сигнализирует об окончании операции обновления. Все изменения, произошедшие между началом и окончанием обновления INCT_Analyzer будут применены. Входных и выходных параметров нет.	
Get_CanUseEqualToAssignRegister	Функция возвращает значение настройки 'Можно использовать символ = для задания значения регистра' (логическое). Истина: символ '=' используется анализатором текста управляющей программы для задания значения регистра. Ложь: при	Если настройка включена (значение Истина), следующая комбинация регистров и значений в тексте управляющей программы будет считаться корректной: X=10.0 Y10.0 Z-1 A=-1 B+3.5 C=-3.5

Метод	Описание метода	Описание использования
	определении значения регистра встреченный символ '=' считается некорректной лексемой. Входных параметров нет.	Если настройка выключена (значение Ложь), корректной будет считаться такая комбинация задания значений регистрам: Y10.0 Z-1 B+3.5
Set_CanUseEqualToAssignRegister	Метод устанавливает значение настройки 'Можно использовать символ = для задания значения регистра'. Входной параметр: Value (логическое), описание параметра см. в Get_CanUseEqualToAssignRegister. Выходных параметров нет.	
Get_RegistersContainSpaces	Устарел Функция возвращает значение настройки 'Регистр может содержать пробелы' (логическое). Истина: при определении регистра анализатор текста управляющей программы учитывает возможность использования в адресе регистра пробелов. Ложь: при определении регистра пробел считается только разделителем лексем. Входных параметров нет.	Устарел Если настройка включена (значение Истина), следующая комбинация регистров и значений в тексте управляющей программы будет считаться корректной: TOOL CALL 1 CYCL CALL при условии, что в списке регистры определены с адресами 'TOOL CALL', 'CYCL CALL'. Регистру с адресом 'TOOL CALL' будет присвоено значение 1. Регистр с
Set_RegistersContainSpaces	Устарел	

Метод	Описание метода	Описание использования
	Метод устанавливает значение настройки 'Регистр может содержать пробелы'. Входной параметр: Value (логическое), описание параметра см. в Get_RegistersContainSpaces. Выходных параметров нет.	адресом 'CYCL CALL' должен быть определён как регистр без значения (vatOnlyAddr). Если настройка выключена анализатор будет считать 'TOOL', 'CALL', 'CYCL', 'CALL' отдельными лексемами.
Get_RegistersCanEndWithPlusOrMinus	Устарел Функция возвращает значение настройки 'Регистр может заканчиваться плюсом или минусом' (логическое). Истина: при определении регистра анализатор текста управляющей программы учитывает возможность использования в окончании адреса регистра символов плюс или минус. Ложь: при определении регистра символы плюс и минус считаются отдельными лексемами, не входящими в состав адреса регистра. Входных параметров нет.	Устарел Если настройка включена (значение Истина), следующая комбинация регистров и значений в тексте управляющей программы: DL+0 DR+0 будут распознаны как регистр 'DL+' и значение 0, регистр 'DR+' и значение 0. При условии, что в списке адресов регистры определены с адресами 'DL+', 'DR+'. Если настройка выключена, анализатор будет считать 'DL', '+0', 'DR', '+0' отдельными лексемами.
Set_RegistersCanEndWithPlusOrMinus	Устарел Метод устанавливает значение настройки 'Регистр может заканчиваться плюсом или минусом'.	

Метод	Описание метода	Описание использования
	Входной параметр: Value (логическое), описание параметра см. в Get_RegistersCanEndWithPlusOrMinus. Выходных параметров нет.	
Get_RegistersCanEndWithNumber	Устарел Функция возвращает значение настройки 'Регистр может заканчиваться числом' (логическое). Истина: при определении регистра анализатор текста управляющей программы учитывает возможность использования в окончании адреса регистра целого числа. Ложь: при определении регистра символы числа считаются отдельными лексемами, не входящими в состав адреса регистра. Входных параметров нет.	Устарел Если настройка включена (значение Истина), следующая комбинация регистров и значений в тексте управляющей программы: DR2+0 R0 будут распознаны как регистр 'DR2' и значение 0, регистр 'R0' без значения. При условии, что в списке адресов регистры определены с адресами 'DR2', 'R0'. А регистр с адресом 'R0' должен быть определён как регистр без значения (vatOnlyAddr).
Set_RegistersCanEndWithNumber	Устарел Метод устанавливает значение настройки 'Регистр может заканчиваться числом'. Входной параметр: Value (логическое), описание параметра см. в	Если настройка выключена, анализатор будет считать 'DR', '2', '+0', 'R', '0' отдельными лексемами.

Метод	Описание метода	Описание использования
	Get_RegistersCanEndWithNumber. Выходных параметров нет.	
Get_LineBeginWithNumber	Функция возвращает значение настройки 'Строка управляющей программы может начинаться с числа' (логическое). Истина: при анализе строки текста управляющей программы первая лексема - число игнорируется. Ложь: при анализе строки текста управляющей программы первое число считается лексемой. Входных параметров нет.	Если настройка включена (значение Истина), первая лексема - число в строке управляющей программы будет пропущена.
Set_LineBeginWithNumber	Метод устанавливает значение настройки 'Строка управляющей программы может начинаться с числа'. Входной параметр: Value (логическое), описание параметра см. в Get_LineBeginWithNumber. Выходных параметров нет.	
Get_Comments	Функция возвращает интерфейс для управления настройками определения комментариев в тексте управляющей программы (интерфейс INCT_Comments).	Если в настройку комментариев добавлены следующие комбинации открывающих и закрывающих символов:

Метод	Описание метода	Описание использования
	Входных параметров нет.	'!', '' '(*', '*)' то строки текста управляющей программы: ! X10 Y20 Z30.5 skip A+1 (* test G01 X100 F100 *) будут считаться комментариями (будут пропущены).
Get_Brackets	Функция возвращает интерфейс для управления настройками определения скобок в тексте управляющей программы (интерфейс INCT_Brackets). Входных параметров нет.	Добавлены для определения скобок в выражениях. Скобки распознаются как отдельные лексемы, если включена настройка CanUseEqualToAssignRegister, но при определении значения регистра не используются.

Метод	Описание метода	Описание использования
Get_Quotes	Функция возвращает интерфейс для управления настройками определения строк в тексте управляющей программы (интерфейс INCT_Quotes). Входных параметров нет.	Если в настройку определения строк добавлен символ: '"', и включена настройка 'Можно использовать символ = для задания значения регистра', то строка текста управляющей программы: T="1" будет распознана анализатором как регистр T и значение 1. Строковое значение "1" будет сконвертировано в число 1.
Get_UnionLineChars	Функция возвращает интерфейс для управления настройками объединения строк текста управляющей программы (интерфейс INCT_UnionLineChars). Входных параметров нет.	Если в настройку объединения строк добавлен символ: "~", то строки: 1 CYCL DEF 200 ~ Q200=+1~ Q201=-30.645~ Q206=+200~ будут оттранслированы и сынтерпретированы как одна строка: 1 CYCL DEF 200 Q200=+1 Q201=-30.645 Q206=+200
Get_EndOfLineChars	Функция возвращает интерфейс для управления настройками разделения строк текста управляющей программы (интерфейс INCT_EndOfLineChars).	Если в настройку разделения строк добавлен символ: ";", то строки управляющей программы: MoveL [10,10,10,10];MoveL [20,20,20,20];

Метод	Описание метода	Описание использования
	Входных параметров нет.	MoveC [10,10,10,10], [10,10,10,10]; будут переданы интерпретатору для трансляции согласно заданному разделителю: MoveL [10,10,10,10]; MoveL [20,20,20,20]; MoveC [10,10,10,10], [10,10,10,10];
Get_ExcludeEOLChars	Функция возвращает значение настройки 'Исключать символы конца строки из текста УП' (логическое). Истина: при разделении строк согласно настройкам, заданным в INCT_EndOfLineChars, символ конца строки будет исключён из текста и не будет передан интерпретатору. Ложь: символ конца строки будет оставлен в тексте и передан интерпретатору. Входных параметров нет.	Если настройка включена (значение Истина), и заданным разделителем является ';' (INCT_EndOfLineChars) то следующий текст УП: MoveL [10,10,10,10];MoveL [20,20,20,20]; MoveC [10,10,10,10], [10,10,10,10]; будет передан интерпретатору в следующем виде: MoveL [10,10,10,10] MoveL [20,20,20,20] MoveC [10,10,10,10], [10,10,10,10]
Set_ExcludeEOLChars	Метод устанавливает значение настройки	

Метод	Описание метода	Описание использования
	'Исключать символы конца строки из текста УП'. Входной параметр: Value (логическое), описание параметра см. в Get_ExcludeEOLChars. Выходных параметров нет.	т.е. символ разделителя строк будет исключен; если настройка выключена, то текст будет передан так: MoveL [10,10,10,10]; MoveL [20,20,20,20]; MoveC [10,10,10,10], [10,10,10,10];

INCT_Analyzer2 - интерфейс CAM системы, который реализует доступ к управлению настройками анализа текста управляющей программы.

Описание методов интерфейса:

Метод	Описание метода	Описание использования
Get_RegistersCaseSensitivity	Функция возвращает значение настройки "Чувствительность к регистру символов". Истина: при анализе текста управляющей программы определение адреса регистров будет производиться с учётом регистра символов (регистры с адресом "A" и "a" - разные). Ложь: при анализе текста управляющей программы определение адреса регистров будет производиться без учёта регистра символов (регистры с адресом "A" и "a" - одинаковые). Входных параметров нет.	Если настройка "Чувствительность к регистру символов" включена (значение Истина), и все регистры интерпретатора заданы заглавными буквами, то регистры заданные строчными буквами будут пропущены. Для следующего текста УП; G0 X0 Y0 Z0 x10 y10 z10 Все регистры первой строки будут распознаны и добавлены в список объектов для интерпретации INCT_BlockExecArray. Все регистры второй строки будут пропущены (не будут распознаны).
Set_RegistersCaseSensitivity	Метод устанавливает значение настройки "Чувствительность к регистру символов". Входной параметр: Value (логическое), описание параметра см. в Get_RegistersCaseSensitivity. Выходных параметров нет.	

INCT_Comments - интерфейс CAM системы, который реализует список настроек для управления комментариями. Настройки состоят из пар строк: открывающий набор символов и

закрывающий набор символов. При анализе кадра управляющей программы символы между любой парой открывающего из закрывающего набора символов считаются комментарием и пропускаются. Закрывающий набор символов может быть пустой строкой, в этом случае комментарием считается остаток строки от открывающего набора символов.

Описание методов интерфейса:

Метод	Описание метода	Описание использования
Add	Функция добавляет пару настроек в список, возвращает индекс добавленной пары в списке (целое число). Входные параметры: Open (строка) - открывающий набор комментариев; Close (строка) - закрывающий набор комментариев.	Для добавления настроек комментариев при настройке анализатора.
Get	Метод возвращает настройки комментариев по индексу. Входной параметр: Idx (целое число) - индекс пары настроек. Выходные параметры: Open (строка) - открывающий набор комментариев; Close (строка) - закрывающий набор комментариев.	Для получения настроек.
Delete	Метод удаляет пару настроек по заданному параметру. Входной параметр: Idx (целое число) - индекс пары настроек.	Для удаления настроек комментариев при перенастройке анализатора.
Get_Count	Функция возвращает количество пар настроек (целое число).	Для получения количества настроек.

INCT_Brackets - интерфейс CAM системы, который реализует список настроек для управления определением скобок в тексте управляющей программы. Настройки состоят из пары символов: открывающий и закрывающий. Скобки распознаются как отдельные лексемы, если включена настройка CanUseEqualToAssignRegister, но при определении значения регистра не используются; настройки добавлены для использования в будущем.

Описание методов интерфейса:

Метод	Описание метода	Описание использования
Add	Функция добавляет пару настроек в список, возвращает индекс добавленной пары в списке (целое число). Входные параметры: Open (символ) - открывающий символ скобки; Close (символ) - закрывающий символ скобки.	Для добавления настроек скобок при настройке анализатора.
Get	Метод возвращает настройки скобок по индексу. Входной параметр: Idx (целое число) - индекс пары настроек. Выходные параметры: Open (символ) - открывающий символ скобки; Close (символ) - закрывающий символ скобки.	Для получения настроек.
Delete	Метод удаляет пару настроек по заданному параметру. Входной параметр: Idx (целое число) - индекс пары настроек.	Для удаления настроек скобок при перенастройке анализатора.
Get_Count	Функция возвращает количество пар настроек (целое число).	Для получения количества настроек.

INCT_Quotes - интерфейс CAM системы, который реализует список настроек для определения строк. Каждая настройка состоит из символа с которого начинается и заканчивается строковое значение. При анализе кадра управляющей программы символы, находящиеся между любой парой символов определения строк считаются строковым значением.

Описание методов интерфейса:

Метод	Описание метода	Описание использования
Add	Функция добавляет настройку в список, возвращает индекс добавленной настройки в списке (целое число). Входные параметры: Sym (символ) - символ с которого начинается и заканчивается строковое значение.	Для добавления настроек определения строковых значений при настройке анализатора.
Get	Метод возвращает настройку по индексу. Входной параметр: Idx (целое число) - индекс настройки. Выходные параметры: Sym (символ) - символ с которого начинается и заканчивается строковое значение.	Для получения настроек.
Delete	Метод удаляет настройку по заданному параметру. Входной параметр: Idx (целое число) - индекс настройки.	Для удаления настроек определения строковых значений при перенастройке анализатора.
Get_Count	Функция возвращает количество настроек (целое число).	Для получения количества настроек.

INCT_UnionLineChars - интерфейс CAM системы, который реализует список настроек для объединения строк текста управляющей программы. Каждая настройка состоит из символа объединения строк. Если кадр управляющей программы заканчивается любым из символов списка настройки, то такая строка объединяется со следующей, без учёта символа объединения.

Описание методов интерфейса:

Метод	Описание метода	Описание использования
Add	Функция добавляет настройку в список, возвращает индекс добавленной настройки в списке (целое число). Входные параметры: Sym (символ) - символ объединения строки.	Для добавления настроек символов объединения строки при настройке анализатора.
Get	Метод возвращает настройку по индексу. Входной параметр: Idx (целое число) - индекс настройки. Выходные параметры: Sym (символ) - символ объединения строки.	Для получения настроек.
Delete	Метод удаляет настройку по заданному параметру. Входной параметр: Idx (целое число) - индекс настройки.	Для удаления настроек символов объединения строки при перенастройке анализатора.
Get_Count	Функция возвращает количество настроек (целое число).	Для получения количества настроек.

INCT_EndOfLineChars - интерфейс CAM системы, который реализует список настроек для разделения строк текста управляющей программы. Каждая настройка состоит из строки - разделителя. Текст управляющей программы передаётся интерпретатору построчно, где каждая строка, это содержимое между разделителями,

Описание методов интерфейса:

Метод	Описание метода	Описание использования
Add	Функция добавляет настройку в список, возвращает индекс	Для добавления настроек разделителя строк при настройке анализатора.

Метод	Описание метода	Описание использования
	добавленной настройки в списке (целое число). Входные параметры: Str (строка) - разделитель.	
Get	Метод возвращает настройку по индексу. Входной параметр: Idx (целое число) - индекс настройки. Выходные параметры: Str (строка) - разделитель.	Для получения настроек.
Delete	Метод удаляет настройку по заданному параметру. Входной параметр: Idx (целое число) - индекс настройки.	Для удаления настроек разделителей строк при перенастройке анализатора.
Get_Count	Функция возвращает количество настроек (целое число).	Для получения количества настроек.

Упрощение построения траектории движения инструмента

В разделе описаны интерфейсы CAM системы, которые упрощают построение траектории движения инструмента.

Линейные перемещения

INCT_LinearMotionFormer - интерфейс CAM системы, который реализует функциональность по упрощению формирования линейных перемещений. Интерфейс возвращает функция INCT_CLData.CreateLinearMotionFormer.

Предназначение:

Процесс формирования линейных перемещений по кадру управляющей программы происходит следующим образом. Текст управляющей программы транслируется, т.е. разбирается на регистры со значениями. Регистры со значениями добавляются в список объектов для интерпретации. Объекты списка интерпретируются. При интерпретации объектов, по изменившимся значениям регистров, которые связаны с осями станка кинематической схемы, в траекторию движения инструмента (INCT_CLData) добавляется команда многокоординатных перемещений (INCT_CLData.OpenMultiMotion, AddMotion, CloseMultiMotion).

Если рассмотреть этот процесс детальнее, получим следующий набор действий.

До интерпретации управляющей программы:

1. *Получить список регистров, связанных с осями станка в кинематической схеме.*
2. *Организовать хранение списка регистров с индексами осей.*

Перед интерпретацией кадра:

1. *Получить текущие значения координат положения инструмента для всех осей, связанных с осями станка в кинематической схеме.*

При интерпретации регистров:

1. *Получить новые значения регистров кадра, связанных с осями станка в кинематической схеме.*
2. *Установить новые значения для всех изменённых регистров кадра, связанных с осями станка в кинематической схеме.*

После интерпретации кадра или в процессе интерпретации, но после интерпретации всех регистров:

1. *Выполнить проверку новых значений координат осей со старыми на предмет изменений.*
2. *Если изменение было, то:*
3. *Выполнить OpenMultiMotion.*
4. *Проверить значение каждой оси на предмет изменения, и, если изменение положения инструмента по оси производилось, выполнить AddMotion.*

5. *Выполнить CloseMultiMotion.*

А для исполнения команд, требующих перемещения в физических координатах станка, дополнительно необходимо переключение типа перемещений в gtmPhysic.

Действия из списка выше, *выделенные наклонным шрифтом*, реализованы в формирователе линейных перемещений (INCT_LinearMotionFormer). Использование формирователя позволяет упростить написание интерпретатора.

Рекомендуем создавать один формирователь линейных перемещений на весь цикл интерпретации управляющей программы. Создание производить при инициализации интерпретатора (INCT_Interpreter.Initialize).

Описание методов интерфейса:

Метод	Описание метода	Описание использования
SetGotoMode	Метод устанавливает тип формируемых перемещений. Входной параметр: Mode (TNCGotoMode) - тип перемещения, см. описание OpenMultiMotion. Если тип перемещения не задан методом, то имеет значение gtmTooltip. Выходных параметров нет.	Функция может использоваться при интерпретации G53 в ISO-кодах (устанавливать тип перемещения в gtmPhysic, затем возвращать в gtmTooltip).
Get_ClearGotoMode	Функция возвращает значение параметра ClearGotoMode (логический). Входных параметров нет.	Параметр ClearGotoMode управляет автоматическим сбросом значения тип перемещения (см. SetGotoMode) в значение gtmTooltip. Сброс типа перемещения производится в InitBlock. Как правило, действие команды G53

Метод	Описание метода	Описание использования
Set_ClearGotoMode	Метод устанавливает значение параметра ClearGotoMode. Входной параметр: Value (логический) - значение параметра ClearGotoMode. Выходных параметров нет.	распространяется только на текущий кадр, поэтому целесообразно установить значение параметра ClearGotoMode в Истина. Таким образом, после интерпретации кадра с G53 (где тип перемещения будет установлен в gtmPhysic), тип перемещения сам будет возвращён в значение gtmTooltip при очередном выполнении InitBlock.
InitAxisIndex	Метод добавляет индекс оси в список. Входные параметры: Index (целое число) - индекс добавляемой в список оси. Выходных параметров нет.	Используется для добавления осей, связанных с осями станка в кинематической схеме. Как правило выполняется сразу после получения интерфейса (функция INCT_CLData.CreateLinearMotionFormer) в реализации метода INCT_Interpreter.Initialize интерпретатора. Получить индексы регистров, связанных с осями станка, можно при помощи использования интерфейсов INCT_SysState, INCT_Equipment.
ClearAxesIndexes	Очистить список индексов осей	Используется при смене инструмента, если названия

Метод	Описание метода	Описание использования
		осей, управляющих инструментом совпадают.
InitBlock	Метод запоминает во внутренний список текущие значения координат инструмента для всех осей из списка, сформированного при помощи InitAxisIndex. Если значение параметра ClearGotoMode Истина, то тип перемещения, задаваемый в SetGotoMode устанавливается в значение gtmTooltip. Входных и выходных параметров нет.	Необходимо использовать перед интерпретацией объектов, связанных с регистрами. Рекомендуется использовать в реализации метода INCT_Interpreter.BeforeInterprete.
SetAxisValue	Метод запоминает во внутренний список новые значения координат инструмента для заданной оси. Входные параметры: Abs (логическое) - тип нового значения координаты, Истина — абсолютное значение, Ложь — относительное значение (относительно текущего значения координаты, см. InitBlock); Index (целое) - индекс оси; Value - (числовое) - новое значение. Выходных параметров нет.	Используется для сохранения нового значения во внутреннем списке с целью последующего формирования команды многокоординатного перемещения. Используется при интерпретации с INCT_Interpreter.Interprete.

Метод	Описание метода	Описание использования
MakeWorkpath	Метод сравнивает список текущих координат (сохранённых в InitBlock) с новыми координатами движения инструмента (сохранённых в SetAxisValue). Если изменение координат присутствовало, в траектории движения инструмента формируется команда многокоординатных перемещений по изменённым координатам осей (INCT_CLData.OpenMultiMotion, AddMotion, CloseMultiMotion).	Для формирования команды многокоординатного перемещения. Используется при интерпретации в INCT_Interpreter.Interprete, после интерпретации всех регистров (добавить в конец списка объектов для интерпретации). Либо в INCT_Interpreter.AfterInterprete.

INCT_GeomLinearMotionFormer - интерфейс CAM системы, который реализует функциональность по упрощению формирования линейных перемещений. Интерфейс возвращает функция INCT_GeomCLData.CreateLinearMotionFormer.

Предназначение соответствует интерфейсу INCT_LinearMotionFormer, только для перемещений в геометрических осях. Перемещения формируются командой CutTo3d. Поэтому для формирования первого перемещения необходимо задать значения координат всех трёх осей.

Описание методов интерфейса:

Метод	Описание метода	Описание использования
ClearSpecifiedAxes	Метод очищает список заданных осей для первого перемещения. Входных и выходных параметров нет.	Необходимо использовать в начале каждой операции. Рекомендуется использовать в реализации метода INCT_Interpreter.NextOperation.
InitBlock	Метод запоминает во внутренний список текущие значения 3-х координат инструмента.	Необходимо использовать перед интерпретацией объектов, связанных с регистрами. Рекомендуется использовать в реализации

Метод	Описание метода	Описание использования
	Входных и выходных параметров нет.	метода INCT_Interpreter.BeforeInterprete.
SetAxisValue	Метод запоминает во внутренний список новые значения координат инструмента для заданной оси. Входные параметры: Abs (логическое) - тип нового значения координаты, Истина — абсолютное значение, Ложь — относительное значение (относительно текущего значения координаты, см. InitBlock); Index (целое число) - индекс оси (1-3); Value - (вещественное значение) - новое значение. Выходных параметров нет.	Используется для сохранения нового значения во внутреннем списке с целью последующего формирования команды перемещения. Используется при интерпретации с INCT_Interpreter.Interprete.
MakeWorkpath	Метод сравнивает список текущих координат (сохранённых в InitBlock) с новыми координатами движения инструмента (сохранённых в SetAxisValue). Если изменение координат присутствовало, в траектории движения инструмента и заданы значения координат для 3-х осей, формируется команда перемещений (INCT_GeomCLData.CutTo3d).	Для формирования команды перемещения. Используется при интерпретации в INCT_Interpreter.Interprete, после интерпретации всех регистров (добавить в конец списка объектов для интерпретации). Либо в INCT_Interpreter.AfterInterprete.

Перемещения по дуге

INCT_CircularMotionFormer - интерфейс CAM системы, который реализует функциональность, позволяющую упростить формирования перемещений по дуге. Интерфейс возвращают функции INCT_CLData.CreateCircularMotionFormer, INCT_GeomCLData.CreateCircularMotionFormer.

Предназначение:

Процесс формирования перемещений по дуге по кадру управляющей программы происходит следующим образом. Текст управляющей программы транслируется, т.е. разбирается на регистры со значениями. Регистры со значениями добавляются в список объектов для интерпретации. Объекты списка интерпретируются. При интерпретации объектов, по изменившимся значениям регистров, которые связаны с осями станка кинематической схемы, в траекторию движения инструмента (INCT_CLData/INCT_GeomCLData) добавляется команда перемещений по дуге (INCT_CLData.AddCircle/INCT_GeomCLData.ArcTo2d).

Если рассмотреть этот процесс детальнее, получим следующий набор действий.

Перед интерпретацией кадра:

1. *Получить индексы осей X, Y, Z, для текущего инструмента или рабочего объекта.*
2. *Организовать хранение индексов осей.*
3. *Получить текущие значения координат положения инструмента для сохранённых в списке осей, отдельно для точки центра дуги.*

При интерпретации регистров:

1. Получить новые значения регистров кадра, связанных с точкой центра, конечной точкой, радиусом или углом, направлением, дуги.
2. *Установить новые значения для всех изменённых параметров дуги.*

После интерпретации кадра или в процессе интерпретации, но после интерпретации всех регистров:

1. *Выполнить расчёт недостающих параметров, для использования в INCT_CLData.AddCircle/INCT_GeomCLData.ArcTo2d.*
2. *Выполнить INCT_CLData.AddCircle/INCT_GeomCLData.ArcTo2d.*

Действия из списка выше, *выделенные наклонным шрифтом*, реализованы в формирователе перемещений по дуге (INCT_CircularMotionFormer). Использование формирователя позволяет упростить написание интерпретатора.

Рекомендуем создавать один формирователь перемещений по дуге на весь цикл интерпретации управляющей программы. Создание производить при инициализации

интерпретатора (INCT_Interpreter.Initialize).

Описание методов интерфейса:

Метод	Описание метода	Описание использования
InitBlock	Метод запоминает во внутренний список индексы осей и текущие значения для координат X, Y, Z текущего инструмента или рабочего объекта. Входных и выходных параметров нет.	Необходимо использовать перед интерпретацией объектов, связанных с регистрами. Рекомендуется использовать в реализации метода INCT_Interpreter.BeforeInterprete.
InitCenterPoint	Метод запоминает текущие значения координат X, Y, Z текущего инструмента или рабочего объекта как точку центра. Входных и выходных параметров нет.	Необходимо использовать перед интерпретацией объектов, связанных с регистрами. Рекомендуется использовать в реализации метода INCT_Interpreter.BeforeInterprete или **INCT_Interpreter.Interprete** (если точка центра окружности задаётся не в текущем кадре).
SetConstructMode	Метод задаёт режим построения дуги. Входной параметр: Mode (TNCConstructArcMode) - режим построения дуги: caByCircleCenter - по направлению, конечной точке и центру дуги caByRadius - по направлению, конечной точке и радиусу дуги	Необходимо использовать при интерпретации регистров в реализации метода **INCT_Interpreter.Interprete**.

Метод	Описание метода	Описание использования
	Выходных параметров нет.	
SetMode	Метод задаёт режим направления дуги. Входной параметр: Mode (TNCCircularMode) - направление дуги: cmCW - по часовой стрелке; cmCCW - против часовой стрелки. Выходных параметров нет.	
SetEndPoint	Метод задаёт значение координаты одной из осей конечной точки. Входные параметры: Abs (логическое) - тип значения конечной точки, Истина — абсолютное значение, Ложь — относительное значение (относительно текущего значения координаты, см. InitBlock); AxisType (TNCAxisType) - ось axX - ось X, axY - ось Y, axZ - ось Z; Value - (числовое) - значение координаты. Выходных параметров нет.	
SetEndPointByAngle	Метод задаёт значение конечной точки дуги по заданному углу относительно текущей точки.	

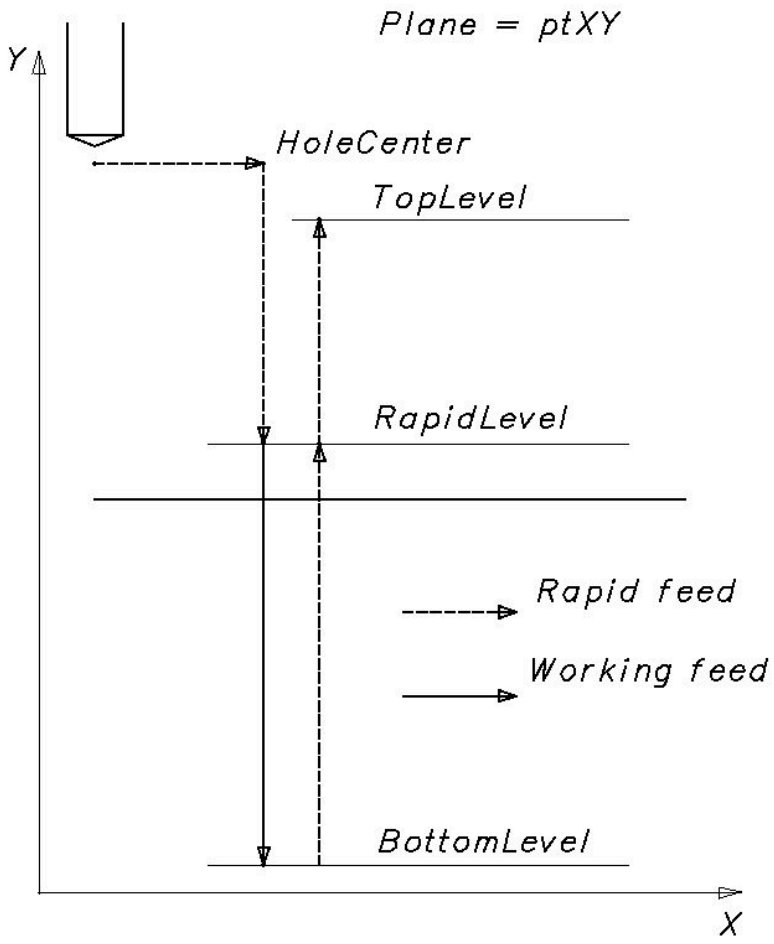
Метод	Описание метода	Описание использования
	Входные параметры: Plane (TNCPlaneType) - текущая плоскость ptXY - плоскость XY, ptYZ - плоскость YZ, ptZX - плоскость ZX; Angle - (числовое) - заданный угол дуги. Выходных параметров нет. Важно! Если выполняется построение полной окружности (угол дуги окружности равен 360 градусов), необходимо конечную точку траектории принудительно приравнять значению начальной (текущей) точке. Иначе, может возникнуть ситуация, в результате которой из-за погрешности вычислений координат конечной точки в одной сотой или тысячной части числа, следующая команда перемещения окажется некорректной. Это особенно актуально для тех команд, в которых начальная точка используется при расчете вспомогательных параметров, участвующих в построении траектории (например, при перемещении инструмента по дуге окружности, заданной координатами ее	

Метод	Описание метода	Описание использования
	центра, конечной точки и направлением движения).	
SetCenterPoint	Метод задаёт значение координаты одной из осей точки центра дуги. Входные параметры: Abs (логическое) - тип значения точки центра, Истина — абсолютное значение, Ложь — относительное значение (относительно текущего значения координаты, см. InitCenterPoint); AxisType (TNCAxisType) - ось axX - ось X, axY - ось Y, axZ - ось Z: ; Value - (числовое) - значение координаты. Выходных параметров нет.	
SetRadius	Метод задаёт значение радиуса дуги. Входной параметр: Value - (числовое) - значение радиуса. Выходных параметров нет.	
MakeWorkpath	Метод рассчитывает конечную точку дуги, если был введён угол (SetEndPointByAngle). В зависимости от режима построения дуги (SetConstructMode)	Для формирования команды перемещения по дуге. Используется при интерпретации в INCT_Interpreter.Interprete, после интерпретации всех регистров (добавить в конец списка объектов для

Метод	Описание метода	Описание использования
	рассчитывает недостающие параметры для AddCircle. В траектории движения инструмента добавляет команду перемещения по дуге (INCT_CLData.AddCircle/**INCT_GeomCLData.ArcTo2d**) с рассчитанными ранее параметрами. Входной параметр: LastPlane (TNCPlaneType) - плоскость дуги. Выходных параметров нет.	интерпретации). Либо в INCT_Interpreter.AfterInterprete.

Цикл сверления

INCT_DrillingCycleFormer - интерфейс CAM системы, который реализует функциональность по упрощению формирования перемещений в циклах сверления. Объект, реализующий этот интерфейс возвращает функция INCT_CLData.CreateDrillingCycleFormer, INCT_GeomCLData.CreateDrillingCycleFormer при её выполнении с любым из параметров: dctSimple, dctWithShift, dctPeck.



Предназначение:

Перемещения в различных циклах сверления, для разных стоек в общем случае повторяют одну и ту же траекторию (Fanuc: G74, G81, G82, G84, G85, G86, G88, G89; Sinumerik: CYCLE81, CYCLE82, CYCLE85, CYCLE87, CYCLE88, CYCLE89; Heidenhain: CYCL DEF 201, CYCL DEF 206, CYCL DEF 207 и др.):

1. Позиционирование инструмента над центром отверстия/следующего отверстия на ускоренной подаче.
2. Перемещение инструмента к уровню ускоренного перемещения на ускоренной подаче.
3. Перемещение инструмента к уровню дна отверстия на рабочей подаче.
4. Возврат к уровню ускоренного перемещения на ускоренной подаче.
5. Возврат к уровню отхода/возврата на ускоренной подаче.

6. Если количество повторений не закончилось, вернуться к пункту 1.

Использование формирователя позволяет по заданным параметрам: первоначальное положение, уровни ускоренного перемещения, дна отверстия, количества повторов и пр. сформировать траекторию движения инструмента. А также добавить в траекторию движения на любом формируемом участке необходимые для конкретного цикла особенности: выстой, ориентированный останов, нарезание резьбы и прочее, изменить тип или значение подачи, либо пропустить перемещение инструмента, подготовленное формирователем.

Использование формирователя:

Рекомендуем создавать новый формирователь цикла сверления для каждого нового цикла в кадре управляющей программы, кроме случаев повторения работы цикла в новом кадре (аналог CYCL CALL для Heidenhain). Таким образом, интерфейс старого формирователя должен освобождаться при создании нового или завершении работы интерпретатора.

Для формирования команд движения инструмента по траектории цикла, необходимо выполнить следующие шаги:

1. Получить интерфейс формирователя (INCT_CLData.CreateDrillingCycleFormer, INCT_GeomCLData.CreateDrillingCycleFormer).
2. Задать индексы осей X, Y, Z по которым будут производиться перемещения (Set_AxisIndexes).
3. Задать текущую плоскость (Set_Plane).
4. Задать значение точки центра отверстия (Set_HoleCenter).
5. Задать режим точки центра (Set_HoleCenterMode).
6. Задать уровень ускоренного перемещения (Set_RapidLevel).
7. Задать уровень дна отверстия (Set_BottomLevel).
8. Задать уровень возврата (Set_TopLevel).
9. Задать подачу (Set_Feedrate), единицы измерения подачи (Set_MPM).
10. Задать количество повторений работы цикла (Set_Count).
11. Назначить обработчик событий цикла (Set_Events), если в траекторию цикла нужно вносить изменения.
12. Сформировать команды движения инструмента по траектории цикла (MakeWorkpath).

Описание методов интерфейса:

Метод	Описание метода
Set_AxisIndexes	Метод устанавливает индексы осей X, Y, Z в которых будут производиться перемещения при формировании траектории цикла.

Метод	Описание метода
	Входные параметры: Value (TNCDCAxisIndexes) - X (целое число) - индекс оси X, Y (целое число) - индекс оси Y, Z (целое число) - индекс оси Z. Выходных параметров нет.
Get_AxisIndexes	Функция возвращает индексы осей X, Y, Z в которых будут производиться перемещения при формировании траектории цикла (TNCDCAxisIndexes). Входных параметров нет.
Set_HoleCenter	Метод устанавливает координаты точки центра отверстия. Входные параметры: Value (TNC3DPoint), X, Y, Z - значение координат для соответствующих осей. Выходных параметров нет.
Get_HoleCenter	Функция возвращает координаты центра точки отверстия (TNC3DPoint). Входных параметров нет.

Метод	Описание метода
Set_HoleCenterMode	Метод устанавливает режим точки центра. Входной параметр: Value (логическое) - режим точки центра. Истина - координаты точки центра отверстия (Set_HoleCenter) заданы в абсолютных значениях. Ложь - координаты точки центра отверстия заданы в относительных значениях от текущего положения инструмента. Выходных параметров нет.
Get_HoleCenterMode	Функция возвращает значение режима точки центра (логическое). Входных параметров нет.
Set_Plane	Метод устанавливает текущую плоскость. Входной параметр: Value (TNCPlaneType) - текущая плоскость. ptXY - плоскость XY, сверление отверстия производится вдоль оси Z. ptYZ - плоскость YZ, сверление отверстия производится вдоль оси X. ptZX - плоскость ZX, сверление отверстия производится вдоль оси Y. Выходных параметров нет.
Get_Plane	Функция возвращает текущую плоскость (TNCPlaneType). Входных параметров нет.
Set_RapidLevel	Метод устанавливает уровень ускоренного перемещения по оси сверления.

Метод	Описание метода
	Входной параметр: Value (вещественное число) - уровень ускоренного перемещения. Выходных параметров нет.
Get_RapidLevel	Функция возвращает уровень ускоренного перемещения по оси сверления (вещественное число). Входных параметров нет.
Set_BottomLevel	Метод устанавливает уровень дна отверстия по оси сверления. Входной параметр: Value (вещественное число) - уровень дна отверстия. Выходных параметров нет.
Get_BottomLevel	Функция возвращает уровень дна отверстия по оси сверления (вещественное число). Входных параметров нет.
Set_TopLevel	Метод устанавливает уровень отхода/возврата по оси сверления. Входной параметр: Value (вещественное число) - уровень отхода/возврата. Выходных параметров нет.
Get_TopLevel	Функция возвращает уровень отхода/возврата по оси сверления (вещественное число). Входных параметров нет.

Метод	Описание метода
Set_Feedrate	Метод устанавливает значение рабочей подачи. Входной параметр: Value (вещественное число) - значение подачи. Выходных параметров нет.
Get_Feedrate	Функция возвращает значение рабочей подачи (вещественное число). Входных параметров нет.
Set_MPM	Метод устанавливает признак единиц измерения подачи. Входной параметр: Value (логическое) - признак единиц измерения подачи. Истина - мм в мин. Ложь - мм на оборот. Выходных параметров нет.
Get_MPM	Функция возвращает признак единиц измерения подачи (логическое). Входных параметров нет.
Set_Count	Метод устанавливает количество повторений цикла. Входной параметр: Value (целое число) - количество повторений цикла. Выходных параметров нет. Работа цикла различается в зависимости от режима точки центра отверстия (Set_HoleCenterMode):

Метод	Описание метода
	• для количества повторений = 0 будет произведено только позиционирование инструмента над центром отверстия; • для количества повторений = 1 будет выполнена одна итерация цикла; • для количества повторений > 1 цикл повторится заданное количество раз: если координаты точки центра отверстия заданы в абсолютных значениях - на одном месте; если координаты точки центра отверстия заданы в относительных значениях - на каждой итерации будет производиться сдвиг на значение, заданное в Set_HoleCenter, относительно точки, в которой завершилась предыдущая итерация.
Get_Count	Функция возвращает количество повторений цикла (целое число). Входных параметров нет.

Метод	Описание метода
Set_Events	Метод назначает события для управления поведением цикла. При использовании событий можно добавить в траекторию движения на любом формируемом участке необходимые для конкретного цикла особенности: выстой, ориентированный останов, нарезание резьбы и прочее, изменить тип или значение подачи, либо пропустить перемещение инструмента подготовленное формирователем. Входной параметр: Value (INCT_DrillingCycleEvents) - объект, реализующий события. Подробное описание ниже. Выходных параметров нет.
Get_Events	Функция возвращает объект, реализующий события, связанный с формирователем цикла (INCT_DrillingCycleEvents). Входных параметров нет.
MakeWorkpath	Метод формирует траекторию цикла по заданным параметрам.
Get_NextHole	Функция возвращает рассчитанные для текущей итерации координаты точки текущего отверстия (TNC3DPoint). Значение доступно для использования только внутри событий (Set_Events).
Get_RapidPoint	Функция возвращает рассчитанные для текущей итерации координаты точки ускоренного перемещения (TNC3DPoint). Значение доступно для использования только внутри событий (Set_Events).

Метод	Описание метода
Get_TopPoint	Функция возвращает рассчитанные для текущей итерации координаты точки отхода/возврата (TNC3DPoint). Значение доступно для использования только внутри событий (Set_Events).
Get_BottomPoint	Функция возвращает рассчитанные для текущей итерации координаты точки дна отверстия (TNC3DPoint). Значение доступно для использования только внутри событий (Set_Events).

INCT_DrillingCycleEvents - интерфейс реализуемый в интерпретаторе, который позволяет управлять поведением цикла сверления. Интерфейс представляет из себя набор методов-событий, которые выполняются при перемещении инструмента внутри формирователя траектории цикла. Используя методы интерфейса можно добавить в траекторию движения на любом формируемом участке необходимые для конкретного цикла особенности: выстой, ориентированный останов, нарезание резьбы и прочее, изменить тип или значение подачи, либо пропустить перемещение инструмента подготовленное формирователем.

Если для формирователя траектории цикла назначены события, то любое перемещение инструмента внутри формирователя состоит из 3-х этапов:

1. Выполняется метод INCT_DrillingCycleEvents.BeforeMove. В качестве параметров методу передаются заданные циклом: тип перемещения, координаты точки перемещения, тип подачи.
2. Выполняется метод INCT_DrillingCycleEvents.OnMove. В качестве параметров методу передаются: тип перемещения, координаты точки перемещения, тип подачи, признак выполнения перемещения. Все параметры, кроме типа перемещения в реализации метода могут быть изменены. В зависимости от заданного в OnMove признака, производится выполнение перемещения инструмента в заданную точку с заданной подачей, или перемещение не производится.
3. Выполняется метод INCT_DrillingCycleEvents.AfterMove. В качестве параметров методу передаются: тип перемещения, координаты точки перемещения, тип подачи, возвращённые методом OnMove.

Описание методов интерфейса:

Метод	Описание метода	Описание использования
BeforeMove	Метод будет выполнен перед каждым перемещением инструмента при формировании траектории цикла. Входные параметры: MoveType (TNCDrillingCycleMoveType) - тип перемещения, описание см. ниже; Point (TNC3DPoint) - координаты точки перемещения; Feed (TNCFeedType) - тип подачи. mtMoveToNextHole - перемещение к первому или следующему отверстию; mtMoveToRapidLevel - перемещение к уровню ускоренного перемещения; mtMoveToBottomLevel - перемещение к дну отверстия; mtReturnToRapidLevel - возврат к уровню ускоренного перемещения; mtReturnToTopLevel - возврат к уровню отхода/возврата; mtPeck* - используются при формировании цикла с ломкой стружки. Выходных параметров нет.	При необходимости добавления тех или иных команд перед выполнением перемещения, их нужно добавить в реализацию метода. Например, если есть необходимость установить направление вращения шпинделя по часовой стрелке перед высверливанием отверстия, сделать это можно добавив в реализацию метода BeforeMove вызов INCT_CLData.AddSpindleSpeedOnRPM с соответствующими параметрами по условию MoveType = mtMoveToBottomLevel.
OnMove	Метод будет выполнен перед каждым перемещением инструмента при формировании траектории цикла. Входные параметры: MoveType	При необходимости переопределения значений точки перемещения или подачи, либо для отказа от перемещения (пропуска). Например, если нужно изменить тип подачи с

Метод	Описание метода	Описание использования
	(TNCDrillingCycleMoveType) - тип перемещения, описание см. выше; Входные и выходные параметры: Point (TNC3DPoint) - координаты точки перемещения; Feed (TNCFeedType) - тип подачи; Handled (логическое) - признак выполнения перемещения внутри реализации (Истина - перемещение выполнено, Ложь - перемещение не выполнено). Если после вызова OnMove параметр Handled принимает значение Истина, это значит, что перемещение было выполнено внутри реализации метода OnMove и формирователь цикла не добавляет команду перемещения в траекторию движения инструмента. Если после вызова OnMove параметр Handled принимает значение Ложь, формирователь считает, что перемещение не было выполнено внутри реализации метода OnMove и добавляет в траекторию движения инструмента команду перемещения в точку Point с типом подачи Feed.	ускоренной на рабочую, при возврате после высверливания, то в реализации метода OnMove необходимо изменить значение возвращаемого методом параметра Feed на ffWorking по условию MoveType = mtReturnToRapidLevel (типы подачи см. в описании OutStandardFeed). Если нужно выполнить перемещение инструмента в точку, отличную от рассчитанной циклом, необходимо изменить значение возвращаемого методом OnMove параметра Point. Если перемещение в рассчитанную циклом точку выполнять не нужно, тогда внутри реализации метода OnMove нужно изменить значение параметра Handled на Истина.

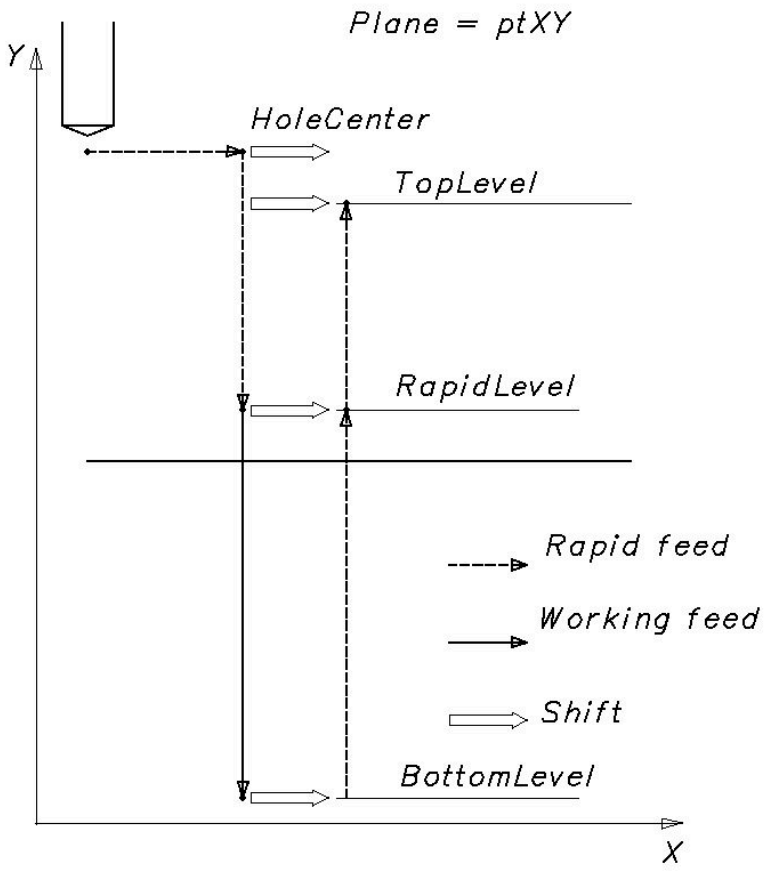
Метод	Описание метода	Описание использования
AfterMove	Метод будет выполнен после каждого перемещения инструмента при формировании траектории цикла, даже если перемещение было отменено в OnMove. Входные параметры: MoveType (TNCDrillingCycleMoveType) - тип перемещения, описание см. выше; Point (TNC3DPoint) - координаты точки перемещения; Feed (TNCFeedType) - тип подачи. Выходных параметров нет.	При необходимости добавления тех или иных команд после выполнением перемещения, их нужно добавить в реализацию метода. Например, если есть необходимость добавить выстой после высверливания отверстия, сделать это можно добавив в реализацию метода AfterMove вызов INCT_CLData.AddDelay с соответствующими параметрами по условию MoveType = mtMoveToBottomLevel.

Цикл сверления со сдвигом

INCT_DrillingCycleWithShiftFormer - интерфейс CAM системы, который реализует функциональность по упрощению формирования перемещений в циклах сверления со сдвигом. При вызове с параметром dctWithShift функция INCT_CLData.CreateDrillingCycleFormer, INCT_GeomCLData.CreateDrillingCycleFormer в качестве результата возвращает объект, реализующий два интерфейса: INCT_DrillingCycleFormer, INCT_DrillingCycleWithShiftFormer.

Посредством интерфейса INCT_DrillingCycleFormer для цикла сверления со сдвигом задаются основные параметры. Описание использования приведено в этой статье Цикл сверления.

Интерфейс INCT_DrillingCycleWithShiftFormer добавляет в реализацию формирователя возможность управления параметрами построения траектории со сдвигом.



Предназначение:

Перемещения в различных циклах сверления со сдвигом, для разных стоек в общем случае повторяют одинаковую траекторию:

1. Позиционирование инструмента над центром отверстия/следующего отверстия на ускоренной подаче.
2. Перемещение инструмента к уровню ускоренного перемещения на ускоренной подаче.
3. Перемещение инструмента к уровню дна отверстия на рабочей подаче.

4. Возврат к уровню ускоренного перемещения на ускоренной подаче.
5. Возврат к уровню отхода/возврата на ускоренной подаче.
6. Если количество повторений не закончилось, вернуться к пункту 1.

На одном или нескольких участках таких циклов траектория движения инструмента смещена вдоль одной или нескольких осей. Примеры таких циклов, Fanuc: G76, G87; Sinumerik: CYCLE86; Heidenhain: CYCL DEF 202 и др.

Использование формирователя:

Рекомендуем создавать новый формирователь цикла сверления для каждого нового цикла в кадре управляющей программы, кроме случаев повторения работы цикла в новом кадре (аналог CYCL CALL для Heidenhain). Таким образом, интерфейс старого формирователя должен освобождаться при создании нового или завершении работы интерпретатора.

Использование интерфейса INCT_DrillingCycleFormer позволяет по заданным параметрам: первоначальное положение, уровни ускоренного перемещения, дна отверстия, количества повторов и пр. сформировать траекторию движения инструмента. А также добавить в траекторию движения на любом формируемом участке необходимые для конкретного цикла особенности: выстой, ориентированный останов, нарезание резьбы и прочее, изменить тип или значение подачи, либо пропустить перемещение инструмента, подготовленное формирователем.

Использование интерфейса INCT_DrillingCycleWithShiftFormer позволяет задать смещение координатам точек перемещения инструмента, на любом из этапов цикла, кроме первого. На каждом из заданных этапов перемещение инструмента из текущей точки в новую точку производится следующим образом:

- перемещение из текущей точки в текущую точку со смещением;
- перемещение из текущей точки со смещением в новую точку со смещением;
- перемещение из новой точки со смещением в новую точку.

Для формирования команд движения инструмента по траектории цикла, необходимо выполнить следующие шаги:

1. Получить интерфейсы формирователя (INCT_CLData.CreateDrillingCycleFormer, INCT_GeomCLData.CreateDrillingCycleFormer с параметром dctWithShift).
2. Задать индексы осей X, Y, Z, по которым будут производиться перемещения (Set_AxisIndexes).
3. Задать текущую плоскость (Set_Plane).
4. Задать значение точки центра отверстия (Set_HoleCenter).
5. Задать режим точки центра (Set_HoleCenterMode).
6. Задать уровень ускоренного перемещения (Set_RapidLevel).

7. Задать уровень дна отверстия (Set_BottomLevel).
8. Задать уровень возврата (Set_TopLevel).
9. Задать подачу (Set_Feedrate), единицы измерения подачи (Set_MPM).
10. Задать количество повторений работы цикла (Set_Count).
11. Назначить обработчик событий цикла (Set_Events), если в траекторию цикла нужно вносить изменения.
12. Задать тип смещения точек перемещения инструмента (SetShiftByOneAxis или SetShiftBySomeAxis).
13. Задать участки траектории движения инструмента, на которых будет действовать смещение (Set_ShiftPathSet).
14. Сформировать команды движения инструмента по траектории цикла (MakeWorkpath).

Описание методов интерфейса INCT_DrillingCycleWithShiftFormer:

Метод	Описание метода
SetShiftByOneAxis	Метод переключает тип расчёта смещений в значение "Расчёт по одной оси" и размер смещения. В этом случае для всех этапов, смещение будет производиться вдоль одной из осей в зависимости от заданной в Set_Plane плоскости: для ptXY по оси X, для ptYZ по оси Y, для ptZX по оси Z. Входной параметр: Value (вещественное число) - значение смещения. Выходных параметров нет.
SetShiftBySomeAxis	Метод переключает тип расчёта смещений в значение "Расчёт по выбранным осям" и задаёт размер смещений для каждой оси. В этом случае смещение точки уровня дна отверстия будет производиться вдоль выбранных осей с заданными значениями. Для остальных уровней смещение будет производиться вдоль выбранных осей, кроме оси сверления, так же с заданными значениями, по оси сверления без смещения. Входной параметр: Axis (TNCShiftAxis.X, Y, Z - логическое) оси, по которым выполняются смещения; Values(TNCShiftValues.X, Y, Z -

Метод	Описание метода
	вещественное число) - значения смещений по осям. Выходных параметров нет.
Set_ShiftPathSet	Метод задаёт этапы в траектории движения инструмента на которых формирователь цикла будет делать смещение. Входной параметр: Value (TNCShiftPathSet) - множество этапов траектории (TNCShiftPath), на которых формирователь цикла будет делать смещение. Можно выбрать несколько этапов траектории. spMoveToRapidLevel - при перемещении к уровню ускоренного перемещения; spMoveToBottomLevel - при перемещении к уровню дна отверстия; spReturnToRapidLevel - при возврате к уровню ускоренного перемещения; spReturnToTopLevel - при возврате к уровню отхода/возврата. В библиотеке типов тип называется SetOfTNCShiftPath и передаётся как байт; TNCShiftPathSet - его имя в Delphi (множество TNCShiftPath). Каждый выбранный этап устанавливает бит с номером, равным его значению TNCShiftPath (битовые маски: spMoveToRapidLevel = 1, spMoveToBottomLevel = 2, spReturnToRapidLevel = 4, spReturnToTopLevel = 8). Выходных параметров нет.
Get_ShiftPathSet	Функция возвращает заданные этапы в траектории движения инструмента на которых формирователь цикла будет делать смещение (TNCShiftPathSet). Входных параметров нет.
Get_ShiftNextHole	Функция возвращает рассчитанные для текущей итерации координаты точки

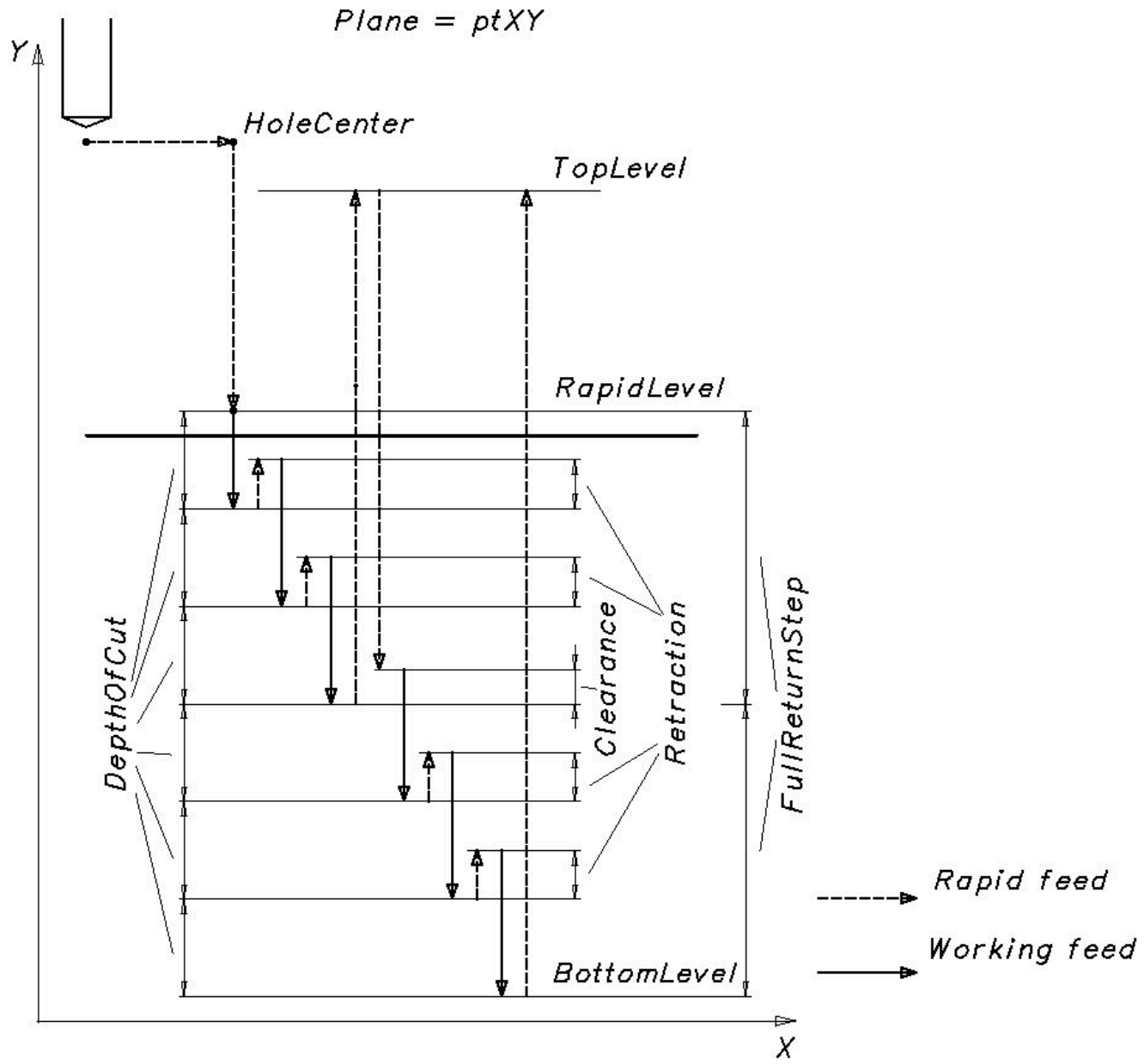
Метод	Описание метода
	текущего отверстия со смещением (TNC3DPoint). Значение доступно для использования только внутри событий (Set_Events). Входных параметров нет.
Get_ShiftRapidPoint	Функция возвращает рассчитанные для текущей итерации координаты точки ускоренного перемещения со смещением (TNC3DPoint). Значение доступно для использования только внутри событий (Set_Events). Входных параметров нет.
Get_ShiftTopPoint	Функция возвращает рассчитанные для текущей итерации координаты точки отхода/возврата со смещением (TNC3DPoint). Значение доступно для использования только внутри событий (Set_Events). Входных параметров нет.
Get_ShiftBottomPoint	Функция возвращает рассчитанные для текущей итерации координаты точки дна отверстия со смещением (TNC3DPoint). Значение доступно для использования только внутри событий (Set_Events). Входных параметров нет.

Цикл сверления с ломкой стружки

INCT_PeckDrillingCycleFormer - интерфейс CAM системы, который реализует функциональность по упрощению формирования перемещений в циклах с ломкой стружки. При вызове с параметром dctPeck функция INCT_CLData.CreateDrillingCycleFormer, INCT_GeomCLData.CreateDrillingCycleFormer в качестве результата возвращает объект, реализующий два интерфейса: INCT_DrillingCycleFormer, INCT_PeckDrillingCycleFormer.

Посредством интерфейса INCT_DrillingCycleFormer для цикла сверления с ломкой стружки задаются основные параметры. Описание использования приведено в этой статье Цикл сверления.

Интерфейс INCT_PeckDrillingCycleFormer добавляет в реализацию формирователя возможность управления параметрами построения траектории с ломкой стружки.



Предназначение:

Перемещения в различных циклах сверления с ломкой стружки, для разных стоек в общем случае повторяют одинаковую траекторию:

1. Позиционирование инструмента над центром отверстия/следующего отверстия на ускоренной подаче.
2. Перемещение инструмента к уровню ускоренного перемещения на ускоренной подаче.
3. Перемещение инструмента к уровню дна отверстия на рабочей подаче.
4. Возврат к уровню ускоренного перемещения на ускоренной подаче.
5. Возврат к уровню отхода/возврата на ускоренной подаче.
6. Если количество повторений не закончилось, вернуться к пункту 1.

Выполнение сверления отверстия для таких циклов (этап перемещения к уровню дна отверстия) происходит не за один проход, а в несколько итераций. Примеры таких циклов, Fanuc: G73, G83; Sinumerik: CYCLE83; Heidenhain: CYCL DEF 200, 203, 205 и др.

Использование формирователя:

Рекомендуем создавать новый формирователь цикла сверления для каждого нового цикла в кадре управляющей программы, кроме случаев повторения работы цикла в новом кадре (аналог CYCL CALL для Heidenhain). Таким образом, интерфейс старого формирователя должен освобождаться при создании нового или завершении работы интерпретатора.

Использование интерфейса INCT_DrillingCycleFormer позволяет по заданным параметрам: первоначальное положение, уровни ускоренного перемещения, дна отверстия, количества повторов и пр. сформировать траекторию движения инструмента. А так же добавить в траекторию движения на любом формируемом участке необходимые для конкретного цикла особенности: выстой, ориентированный останов, нарезание резьбы и прочее, изменить тип или значение подачи, либо пропустить перемещение инструмента подготовленное формирователем.

Использование интерфейса INCT_PeckDrillingCycleFormer позволяет задать различные параметры, которые управляют количеством итераций, глубиной реза на итерации, типом отвода инструмента и пр.

Для формирования команд движения инструмента по траектории цикла, необходимо выполнить следующие шаги:

1. Получить интерфейсы формирователя (INCT_CLData.CreateDrillingCycleFormer, INCT_GeomCLData.CreateDrillingCycleFormer с параметром dctPeck).
2. Задать индексы осей X, Y, Z по которым будут производиться перемещения (Set_AxisIndexes).
3. Задать текущую плоскость (Set_Plane).
4. Задать значение точки центра отверстия (Set_HoleCenter).
5. Задать режим точки центра (Set_HoleCenterMode).
6. Задать уровень ускоренного перемещения (Set_RapidLevel).
7. Задать уровень дна отверстия (Set_BottomLevel).
8. Задать уровень возврата (Set_TopLevel).
9. Задать подачу (Set_Feedrate), единицы измерения подачи (Set_MPM).
10. Задать количество повторений работы цикла (Set_Count).
11. Назначить обработчик событий цикла (Set_Events), если в траекторию цикла нужно вносить изменения.
12. Установить параметры итераций для ломки стружки (Set_StartLevel, Set_DepthOfCut, Set_DegressionType, Set_Degression, Set_FullReturn и пр. описание см. ниже).

13. Сформировать команды движения инструмента по траектории цикла (MakeWorkpath).

Метод INCT_DrillingCycleFormer.Set_Events для цикла с ломкой стружки в качестве входного параметра принимает на вход объект, реализующий два интерфейса: INCT_DrillingCycleEvents и INCT_PeckDrillingCycleEvents. Для событий интерфейса INCT_DrillingCycleEvents: BeforeMove, OnMove, AfterMove в качестве типа перемещения, наряду с ранее описанными в статье Цикл сверления значениями, будут передаваться следующие:

mtPeckMoveToNextDepthLevel - перемещение к следующей глубине реза (очередная итерация);
mtPeckReturnToRapidLevel - возврат к уровню ускоренного перемещения на текущей итерации (вывод стружки);
mtPeckMoveToClearanceLevel - перемещение к текущему уровню реза за вычетом зазора, после возврата с уровня ускоренного перемещения;
mtPeckMoveToRetractionLevel - отвод и возврат к текущему уровню реза (ломка стружки).

Описание методов интерфейса INCT_PeckDrillingCycleFormer:

Метод	Описание метода
Set_DepthOfCut	Метод устанавливает глубину реза для итерации. Глубина реза это значение, на которое перемещается инструмент при высверливании отверстия, на каждой итерации. Входной параметр: Value (вещественное число) - глубина реза. Выходных параметров нет.
Get_DepthOfCut	Функция возвращает глубину реза (вещественное число). Входных параметров нет.
Set_StartLevelType	Метод устанавливает тип значения 'Начальный уровень'. Входной параметр: Value (TNCStartLevelType) - тип значения: slFromLevel - от заданного в Set_StartLevel значения начинается отсчёт первой глубины реза. slFirstLevel - заданное в Set_StartLevel значение это первая глубина реза. Выходных параметров нет.

Метод	Описание метода
Get_StartLevelType	Функция возвращает тип значения 'Начальный уровень' (TNCStartLevelType). Входных параметров нет.
Set_StartLevel	Метод устанавливает уровень, относительно которого отсчитывается первая глубина реза или первую глубину реза в зависимости от установленного в Set_StartLevelType значения. Входной параметр: Value (вещественное число) - заданное значение. Выходных параметров нет.
Get_StartLevel	Функция возвращает уровень, относительно которого отсчитывается первая глубина реза или первую глубину реза в зависимости от установленного в Set_StartLevelType значения (вещественное число). Входных параметров нет.
Set_Degression	Метод устанавливает значение дегрессии - изменения глубины реза на каждой итерации. Входной параметр: Value (вещественное число) - значение дегрессии. Расчётное значение дегрессии будет зависеть от типа дегрессии, заданного в Set_DegressionType. Выходных параметров нет.
Get_Degression	Функция возвращает текущее значение дегрессии (вещественное число). Входных параметров нет.
Set_DegressionType	Метод устанавливает тип дегрессии. Входной параметр: Value (TNCDegressionType) - тип дегрессии. dtNone - нет дегрессии, глубина реза не изменяется; dtValue - глубина реза уменьшается на

Метод	Описание метода
	постоянное значение при каждой итерации (постоянное значение задаётся в Set_Degression); dtCoef - глубина реза умножается на коэффициент при каждой итерации (значение коэффициента задаётся в Set_Degression). Выходных параметров нет.
Get_DegressionType	Функция возвращает текущий тип дегрессии (TNCDegressionType). Входных параметров нет.
Set_MinDepthOfCut	Метод устанавливает значение минимальной глубины реза. Как правило, значение используется для циклов с дегрессией. Входной параметр: Value (вещественное число) - значение минимальной глубины реза. Выходных параметров нет.
Get_MinDepthOfCut	Функция возвращает текущее значение минимальной глубины реза (вещественное число). Входных параметров нет.

Метод	Описание метода
Set_Retraction	Метод устанавливает текущее значение отвода. Отвод - значение, на которое поднимается инструмент для ломки стружки, после достижения текущей глубины реза на итерации, если для этой итерации не производится возврат на уровень ускоренного перемещения (т.е. признак возврата в Set_FullReturn задан в значении - Ложь). Входной параметр: Value (вещественное число) - значение отвода. Расчётное значение отвода будет зависеть от типа отвода, заданного в Set_RetractionType. Выходных параметров нет.
Get_Retraction	Функция возвращает текущее значение отвода (вещественное число). Входных параметров нет.
Set_RetractionType	Метод устанавливает значение типа отвода. Входной параметр: Value (TNCRetractionType) - значение типа отвода. rtNone - отвод не производится: rtValue - значение отвода (задаётся в Set_Retraction). Выходных параметров нет.
Get_RetractionType	Функция возвращает текущее значение типа отвода (TNCRetractionType). Входных параметров нет.
Set_RetractionCount	Метод устанавливает количество отходов для ломки стружки. Входной параметр: Value (целое число) - значение количества отходов для ломки стружки. Выходных параметров нет.
Get_RetractionCount	Функция возвращает количество отходов для ломки стружки (целое число).

Метод	Описание метода
	Входных параметров нет.
Set_ClearanceType	Метод устанавливает тип зазора. Зазор - значение, на которое инструмент не достигает уровня глубины реза текущей итерации, после возврата из уровня ускоренного перемещения (т.е. признак возврата в Set_FullReturn задан в значении - Истина). Входной параметр: Value (TNCClearanceType) - тип зазора. ctNone - без зазора; ctValue - значение (задаётся в Set_Clearance). Выходных параметров нет.
Get_ClearanceType	Функция возвращает текущий тип зазора (TNCClearanceType). Входных параметров нет.
Set_Clearance	Метод устанавливает значение зазора. Входной параметр: Value (вещественное число) - значение зазора. Расчётное значение зазора будет зависеть от типа зазора, заданного в Set_ClearanceType. Выходных параметров нет.
Get_Clearance	Функция возвращает текущее значение зазора (вещественное число). Входных параметров нет.
Set_FullReturn	Метод устанавливает признак возврата к уровню ускоренного перемещения. Если признак установлен, то при достижении глубины реза на текущей итерации, производится возврат к уровню ускоренного перемещения, а затем перемещение к уровню глубины реза на итерации с учётом зазора (см. Set_ClearanceType). Входной параметр: Value (логическое) - признак возврата.

Метод	Описание метода
	Истина - с возвратом к уровню ускоренного перемещения; Ложь - без возврата. Выходных параметров нет.
Get_FullReturn	Функция возвращает признак возврата к уровню ускоренного перемещения (логическое). Входных параметров нет.
Set_FullReturnStep	Метод устанавливает шаг итераций реза, на котором осуществляется возврат к уровню ускоренного перемещения. Пример. В качестве шага итераций реза установлено значение 3. В этом случае на каждой третьей итерации производится возврат к уровню ускоренного перемещения (итерации 3, 6, 9 и т.д.). На остальных итерациях возврат не производится (итерации 1, 2, 4, 5, 7, 8 и т.д.). Входной параметр: Value (целое число) - шаг итераций реза. Только, если в Set_FullReturn задано значение Истина (с возвратом). Выходных параметров нет.
Get_FullReturnStep	Функция возвращает шаг итераций реза, на котором осуществляется возврат к уровню ускоренного перемещения (целое число). Входных параметров нет.

Метод	Описание метода
Set_SameLastSteps	Метод устанавливает количество последних одинаковых шагов реза. Если значение отлично от 0 или 1, то заданное методом количество последних итераций будет выполнено с одинаковой глубиной реза. Входной параметр: Value (целое число) - количество последних одинаковых шагов реза. Выходных параметров нет.
Get_SameLastSteps	Функция возвращает текущее количество последних одинаковых шагов реза (целое число). Входных параметров нет.

INCT_PeckDrillingCycleEvents - интерфейс реализуемый в интерпретаторе, который позволяет получать и управлять рассчитываемыми значениями внутри цикла сверления с ломкой стружки. Интерфейс представляет собой набор методов-событий, которые выполняются при расчёте тех или иных значений внутри формирователя траектории цикла. Используя методы интерфейса можно получить и изменить в процессе формирования траектории цикла уровни реза для итераций, признак возврата к уровню ускоренного перемещения на итерации, значения зазора и отвода.

Если для формирователя траектории цикла назначены события, то для каждого рассчитанного значения выполняется вызов соответствующего метода:

1. Метод INCT_PeckDrillingCycleEvents.OnCalcNextDepth выполняется при добавлении во внутренний список формирователя очередного рассчитанного уровня реза. Количество уровней реза (размер списка) будет являться количеством итераций за которые инструмент переместится на дно отверстия. В качестве параметра методу передаётся рассчитанный уровень глубины реза.
2. Метод INCT_PeckDrillingCycleEvents.OnGetNextFullReturn выполняется после расчёта признака возврата к уровню ускоренного перемещения по параметрам цикла. В качестве параметра методу передаётся номер итерации и рассчитанный признак.
3. Метод INCT_PeckDrillingCycleEvents.OnGetNextClearance выполняется после расчёта значения зазора. В качестве параметра методу передаётся номер итерации и рассчитанное значение зазора.
4. Метод INCT_PeckDrillingCycleEvents.OnGetNextRetraction выполняется после расчёта значения отвода. В качестве параметра методу передаётся номер итерации и рассчитанное значение отвода.

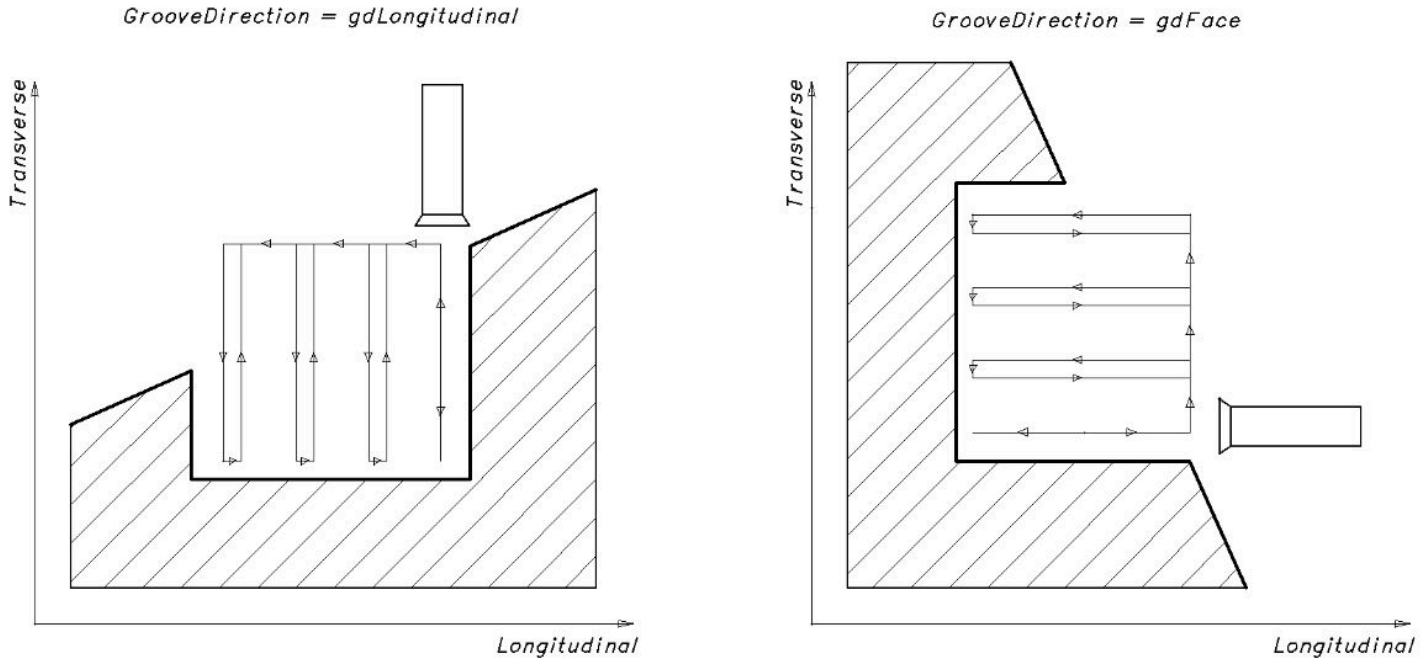
Описание методов интерфейса:

Метод	Описание метода	Описание использования
OnCalcNextDepth	Метод выполняется при каждом добавлении во внутренний список формирователя очередного рассчитанного уровня реза. Размер сформированного списка (количество элементов) будет являться количеством итераций за которые инструмент переместится на дно отверстия. Входной и выходной параметр: Depth (вещественное число) - значение уровня реза для итерации (абсолютное значение).	1. Для получения общего количества итераций, для пересчёта параметров зазора, отвода или возврата к уровню ускоренного перемещения при реализации алгоритмов расчёта, не заложенных в функциональность цикла. 2. Для изменения уровней глубин реза при реализации алгоритмов расчёта, не заложенных в функциональность цикла.
OnGetNextFullReturn	Метод выполняется после расчёта признака возврата к уровню ускоренного перемещения по параметрам цикла для текущей итерации. Входные параметры: Iteration (целое) - номер итерации. Входной и выходной параметр: FullReturn (логическое) - признак возврата к уровню ускоренного перемещения на заданной итерации.	Для изменения признака возврата к уровню ускоренного перемещения для текущей итерации при реализации алгоритмов расчёта признака, не заложенных в функциональность цикла.

Метод	Описание метода	Описание использования
OnGetNextClearance	Метод выполняется после расчёта значения зазора по параметрам цикла для текущей итерации. Вызывается, только если признак возврата к уровню ускоренного перемещения для итерации установлен (FullReturn = Истина). Входные параметры: Iteration (целое) - номер итерации. Входной и выходной параметр: Clearance (вещественное число) - значение зазора на заданной итерации.	Для изменения значения зазора для текущей итерации при реализации алгоритмов расчёта значения, не заложенных в функциональность цикла.
OnGetNextRetraction	Метод выполняется после расчёта значения отвода по параметрам цикла для текущей итерации. Вызывается, только если признак возврата к уровню ускоренного перемещения для итерации не установлен (FullReturn = Ложь). Входные параметры: Iteration (целое) - номер итерации. Входной и выходной параметр: Retraction (вещественное число) - значение отвода на заданной итерации.	Для изменения значения отвода для текущей итерации при реализации алгоритмов расчёта значения, не заложенных в функциональность цикла.

Токарный цикл точения канавок

INCT_LatheGroovingCycleFormer - интерфейс CAM системы, который реализует функциональность по упрощению формирования перемещений токарного цикла точения канавок. Объект, реализующий этот интерфейс, возвращает функция INCT_CLData.CreateLatheCycleFormer, INCT_GeomCLData.CreateLatheCycleFormer при её выполнении с параметром lctGrooving.



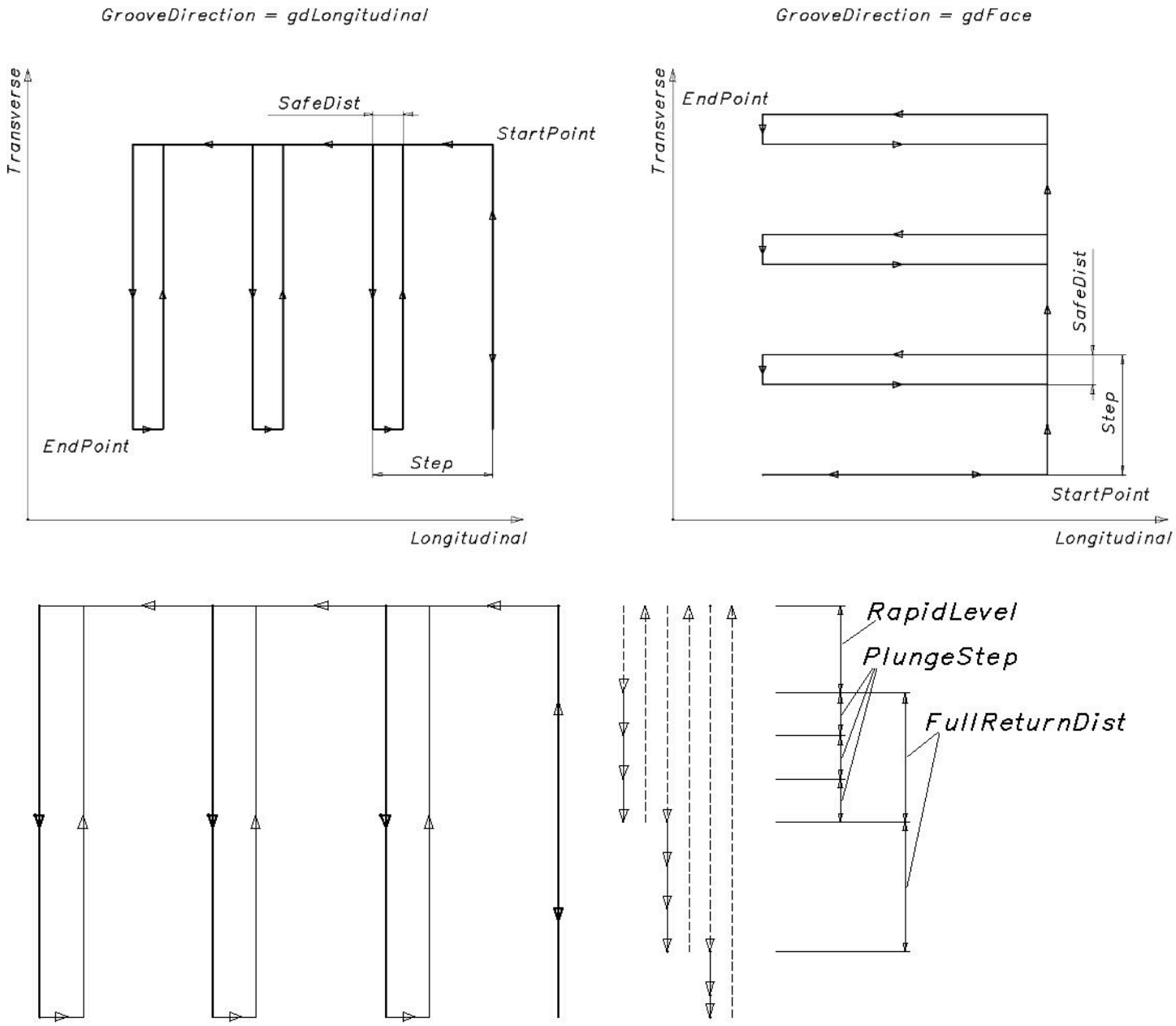
Предназначение:

Выполнение перемещений в различных типах токарных циклов точения канавок, для разных стоек в общем случае повторяют одинаковую траекторию:

1. Подвод инструмента к стартовой или следующей точке врезания на ускоренной подаче.
2. Врезание на глубину канавки за один проход или в несколько итераций на величину шага резания на рабочей подаче, с отскоком для ломки стружки.
3. Возврат в стартовую позицию с отводом или без него, с подачей перехода.
4. Если ширина канавки не достигнута, вернуться к пункту 1.

Примеры таких циклов, Sinumerik: CYCLE93.

Использование формирователя позволяет сформировать траекторию движения инструмента по заданным параметрам: начальная и конечная точки канавки, шаг в ширину канавки, шаг резания, расстояние отвода и пр. А также добавить в траекторию движения на любом формируемом участке необходимые для конкретного цикла особенности, изменить тип или значение подачи, либо пропустить перемещение инструмента, подготовленное формирователем.



Использование формирователя:

Рекомендуем создавать новый формирователь токарного цикла для каждого нового цикла в кадре управляющей программы, кроме случаев повторения работы цикла в новом кадре. Таким образом, интерфейс старого формирователя должен освобождаться при создании нового или в случае завершения работы интерпретатора.

Для формирования команд движения инструмента по траектории цикла, необходимо выполнить следующие шаги:

1. Получить интерфейс формирователя (INCT_CLData.CreateLatheCycleFormer, INCT_GeomCLData.CreateLatheCycleFormer).
2. Задать индексы продольной и поперечной осей, по которым будут производиться перемещения (Set_AxisIndexes).

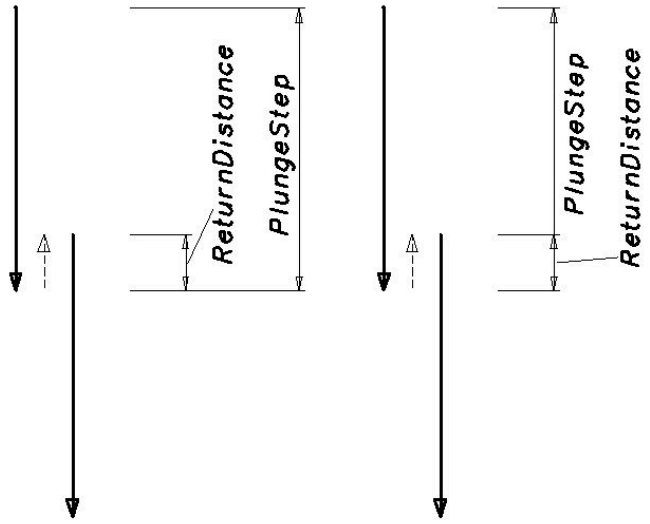
3. Задать начальную точку канавки (Set_StartPoint).
4. Задать конечную точку канавки (Set_EndPoint).
5. Задать направление канавки (Set_GrooveDirection).
6. Задать шаг смещения в ширину канавки (Set_Step).
7. Задать расстояние отвода для возврата (Set_SafeDist).
8. Задать шаг резания в осевом направлении (Set_PlungeStep).
9. Задать значение отскока для ломки стружки (Set_ReturnDistance).
10. Назначить обработчик событий цикла (Set_Events), если в траекторию цикла нужно вносить изменения.
11. Сформировать команды движения инструмента по траектории цикла (MakeWorkpath).

Описание методов интерфейса:

Метод	Описание метода
Set_AxisIndexes	Метод устанавливает индексы продольной и поперечной осей, по которым будут производиться перемещения при формировании траектории цикла. Входные параметры: Value (TNCLCAxisIndexes) - Transverse (целое число) - индекс поперечной оси, Longitudinal (целое число) - индекс продольной оси. Выходных параметров нет.
Get_AxisIndexes	Функция возвращает индексы продольной и поперечной осей, в которых будут производиться перемещения при формировании траектории цикла (TNCLCAxisIndexes). Входных параметров нет.
Set_StartPoint	Метод устанавливает значения координат начальной точки канавки. Входной параметр: Value (TNC2DPoint) - значение координат начальной точки канавки (Transverse - поперечная, Longitudinal - продольная). Выходных параметров нет.
Get_StartPoint	Функция возвращает значение координат начальной точки канавки (TNC2DPoint).

Метод	Описание метода
	Входных параметров нет.
Set_EndPoint	Метод устанавливает значение координат конечной точки канавки. Входной параметр: Value (TNC2DPoint) - значение координат конечной точки канавки. Выходных параметров нет.
Get_EndPoint	Функция возвращает значение координат конечной точки канавки (TNC2DPoint). Входных параметров нет.
Set_GrooveDirection	Метод устанавливает значение направления канавки. Входной параметр: Value (TNCGrooveDirection) - значение направления канавки. gdLongitudinal - продольная; gdFace - торцевая. Выходных параметров нет.
Get_GrooveDirection	Функция возвращает значение направления канавки (TNCGrooveDirection). Входных параметров нет.
Set_Step	Метод устанавливает значение шага в ширину канавки в радиальном направлении. Входной параметр: Value (вещественное число) - значение шага. Выходных параметров нет.
Get_Step	Функция возвращает значение шага в ширину канавки в радиальном направлении (вещественное число). Входных параметров нет.

Метод	Описание метода
Set_SafeDist	Метод устанавливает значение отскока в бок перед выводом инструмента. Входной параметр: Value (вещественное число) - значение отскока в бок. Выходных параметров нет.
Get_SafeDist	Функция возвращает значение отскока в бок перед выводом инструмента (вещественное число). Входных параметров нет.
Set_PlungeStep	Метод устанавливает значение шага при выполнении врезания на глубину канавки. Входной параметр: Value (вещественное число) - значение шага. Выходных параметров нет.
Get_PlungeStep	Функция возвращает значение шага при выполнении врезания на глубину канавки (вещественное число). Входных параметров нет.
Set_ReturnDistance	Метод устанавливает значение отскока вверх для ломки стружки. Входной параметр: Value (вещественное число) - значение отскока вверх. Выходных параметров нет.
Get_ReturnDistance	Функция возвращает значение отскока вверх для ломки стружки (вещественное число). Входных параметров нет.
Set_ReturnDistanceInsidePlungeStep	Метод устанавливает значение признака 'Отскок внутри шага врезания'. Истина - отскок включён в шаг врезания, Ложь - отскок складывается с шагом врезания. Входной параметр: Value (логическое) - значение признака.

Метод	Описание метода
	Выходных параметров нет. 
Get_ReturnDistanceInsidePlungeStep	Функция возвращает значение признака 'Отскок внутри шага врезания'. Входных параметров нет.
Set_RapidLevel	Метод устанавливает значение уровня ускоренного перемещения. Входной параметр: Value (вещественное число) - значение уровня ускоренного перемещения. Выходных параметров нет.
Get_RapidLevel	Функция возвращает значение уровня ускоренного перемещения. Входных параметров нет.
Set_FullReturnDist	Метод устанавливает расстояние возврата к уровню ускоренного перемещения. Входной параметр: Value (вещественное число) - расстояние возврата к уровню ускоренного перемещения. Выходных параметров нет.
Get_FullReturnDist	Функция возвращает расстояние возврата к уровню ускоренного перемещения. Входных параметров нет.

Метод	Описание метода
Set_Feedrate	Метод устанавливает значение рабочей подачи. Входной параметр: Value (вещественное число) - значение рабочей подачи. Выходных параметров нет.
Get_Feedrate	Функция возвращает значение рабочей подачи. Входных параметров нет.
Set_MPM	Метод устанавливает тип рабочей подачи. Входной параметр: Value (логическое) - тип рабочей подачи. Истина - мм/мин, Ложь - мм/об. Выходных параметров нет.
Get_MPM	Функция возвращает тип рабочей подачи. Входных параметров нет.
Set_Events	Метод назначает события для управления поведением цикла. При использовании событий можно добавить в траекторию движения на любом формируемом участке необходимые для конкретного цикла особенности, изменить тип или значение подачи, либо пропустить перемещение инструмента, подготовленное формирователем. Входной параметр: Value (INCT_LatheCycleEvents) - объект, реализующий события. Подробное описание ниже. Выходных параметров нет.
Get_Events	Функция возвращает объект, реализующий события, связанный с формирователем цикла (INCT_LatheCycleEvents). Входных параметров нет.

Метод	Описание метода
MakeWorkpath	Метод формирует траекторию цикла по заданным параметрам.

INCT_LatheCycleEvents - интерфейс, реализуемый в интерпретаторе, который позволяет управлять поведением токарного цикла. Интерфейс представляет из себя набор методов-событий, которые выполняются при перемещении инструмента внутри формирователя траектории цикла. Используя методы интерфейса можно добавить в траекторию движения на любом формируемом участке необходимые для конкретного цикла особенности: выстой, ориентированный останов и прочее, изменить тип подачи, либо пропустить перемещение инструмента, подготовленное формирователем.

Если для формирователя траектории цикла назначены события, то любое перемещение инструмента внутри формирователя состоит из 3-х этапов:

1. Выполняется метод INCT_LatheCycleEvents.BeforeMove. В качестве параметров методу передаются заданные циклом: тип перемещения, координаты точки перемещения, тип подачи.
2. Выполняется метод INCT_LatheCycleEvents.OnMove. В качестве параметров методу передаются: тип перемещения, координаты точки перемещения, тип подачи, признак выполнения перемещения. Все параметры, кроме типа перемещения, в реализации метода могут быть изменены. В зависимости от заданного в OnMove признака, производится выполнение перемещения инструмента в заданную точку с указанной подачей, или перемещение не производится.
3. Выполняется метод INCT_LatheCycleEvents.AfterMove. В качестве параметров методу передаются: тип перемещения, координаты точки перемещения, тип подачи, возвращённые методом OnMove.

Описание методов интерфейса:

Метод	Описание метода	Описание использования
BeforeMove	Метод будет выполнен перед каждым перемещением инструмента при формировании траектории цикла. Входные параметры: MoveType (TNCLatheCycleMoveType) - тип перемещения, описание см.	При необходимости добавления тех или иных команд перед выполнением перемещения, их нужно добавить в реализацию метода. Например, если есть необходимость включить систему охлаждения станка

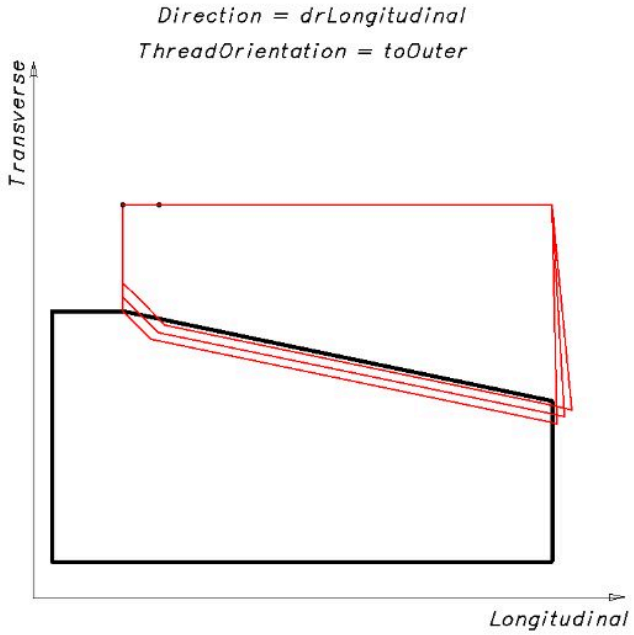
Метод	Описание метода	Описание использования
	ниже; Point (TNC2DPoint) - координаты точки перемещения (Transverse - поперечная, Longitudinal - продольная); Feed (TNCFeedType) - тип подачи. ImMoveToStartPoint - перемещение к стартовой точке точения канавки; ImMoveToNextGroove - перемещение к следующему шагу точения канавки; ImMoveToNextLevel - перемещение на очередную глубину врезания канавки; ImMoveToBottomLevel - перемещение на дно канавки; ImChipBreaking - ломка стружки; ImRetractSideways - отскок в бок, перед выводом инструмента; ImReturnToTopLevel - возврат на начальный уровень; ImReturnToStartPoint - возврат в начальную точку; ImCutTheThread - нарезание резьбы (для цикла Токарный цикл резьбонарезания); ImApproach - подвод (для цикла Токарный цикл обработки резанием); ImRetract - отвод (для цикла обработки резанием); ImWorkingMove - рабочее перемещение (для цикла обработки резанием); ImOverlapContour - проход по	перед точением канавки, сделать это можно добавив в реализацию метода BeforeMove вызов INCT_CLData.AddCoolant с соответствующими параметрами по условию MoveType = ImMoveToStartPoint.

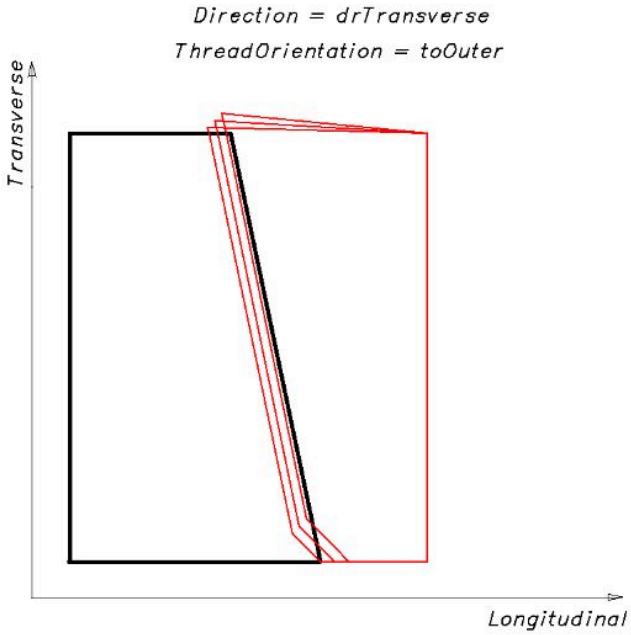
Метод	Описание метода	Описание использования
	контуру (для цикла обработки резанием); lmReturnToSafeDistance - возврат на безопасное расстояние (для цикла обработки резанием); lmTtransition - переход (для цикла обработки резанием); lmMoveToRapidLevel - возврат на уровень ускоренного перемещения (вывод инструмента). Выходных параметров нет.	
OnMove	Метод будет выполнен непосредственно перед каждым перемещением инструмента при формировании траектории цикла. Входные параметры: MoveType (TNCLatheCycleMoveType) - тип перемещения, описание см. выше; Входные и выходные параметры: Point (TNC2DPoint) - координаты точки перемещения; Feed (TNCFeedType) - тип подачи; Handled (логическое) - признак выполнения перемещения внутри реализации (Истина - перемещение выполнено, Ложь - перемещение не выполнено). Если после вызова OnMove параметр Handled принимает	При необходимости переопределения значений точки перемещения или подачи, либо для отказа от перемещения (пропуска). Например, если нужно изменить тип подачи на ускоренную, при возврате на начальный уровень, после точения канавки, то в реализации метода OnMove необходимо изменить значение возвращаемого методом параметра Feed на ffRapid по условию MoveType = lmReturnToTopLevel (типы подачи см. в описании OutStandardFeed). Если нужно выполнить перемещение инструмента в точку, отличную от рассчитанной циклом, необходимо изменить значение возвращаемого методом OnMove параметра

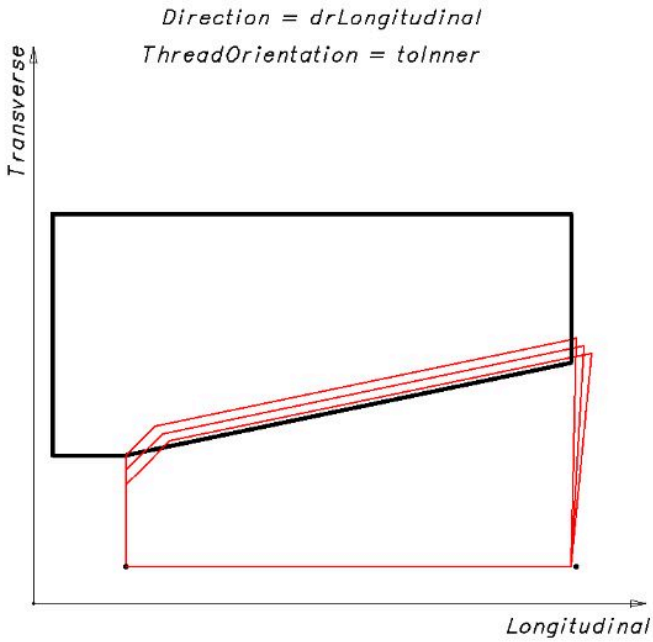
Метод	Описание метода	Описание использования
	значение Истина, это значит, что перемещение было выполнено внутри реализации метода OnMove и формирователь цикла не добавляет команду перемещения в траекторию движения инструмента. Если после вызова OnMove параметр Handled принимает значение Ложь, формирователь считает, что перемещение не было выполнено внутри реализации метода OnMove и добавляет в траекторию движения инструмента команду перемещения в точку Point с типом подачи Feed.	Point. Если перемещение в рассчитанную циклом точку выполнять не нужно, тогда внутри реализации метода OnMove нужно изменить значение параметра Handled на Истина.
AfterMove	Метод будет выполнен после каждого перемещения инструмента при формировании траектории цикла, даже если перемещение было отменено в OnMove. Входные параметры: MoveType (TNCLatheCycleMoveType) - тип перемещения, описание см. выше; Point (TNC2DPoint) - координаты точки перемещения; Feed (TNCFeedType) - тип подачи. Выходных параметров нет.	При необходимости добавления тех или иных команд после выполнения перемещения, их нужно добавить в реализацию метода. Например, если есть необходимость добавить выстой на дне канавки, сделать это можно добавив в реализацию метода AfterMove вызов INCT_CLData.AddDelay с соответствующими параметрами по условию MoveType = lmMoveToBottomLevel.

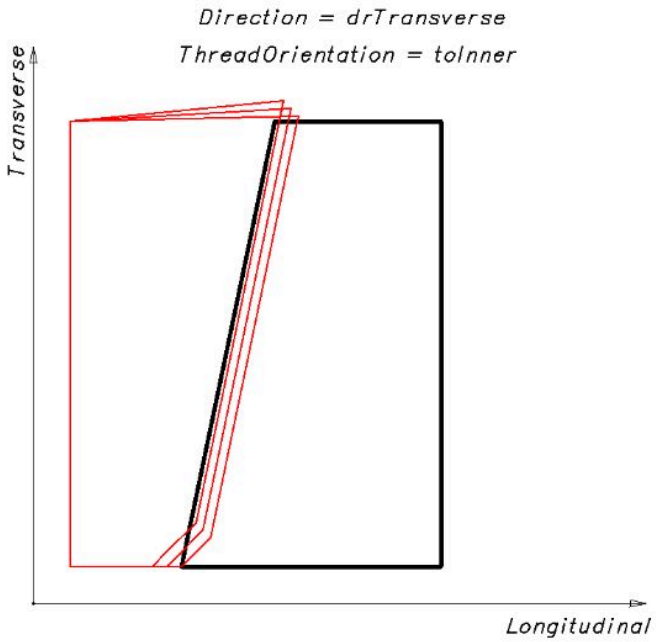
Токарный цикл резьбонарезания

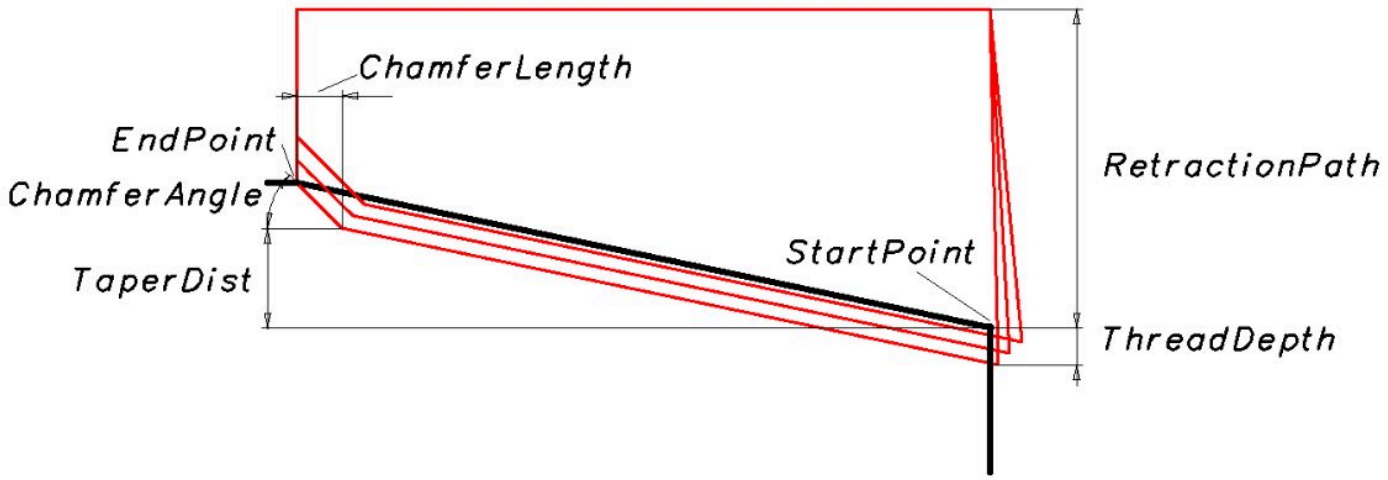
INCT_LatheThreadCuttingCycleFormer - интерфейс CAM системы, который реализует функциональность по упрощению формирования перемещений токарного цикла нарезания резьбы. Объект, реализующий этот интерфейс возвращает функция INCT_CLData.CreateLatheCycleFormer, INCT_GeomCLData.CreateLatheCycleFormer при её выполнении с параметром lctThreadCutting.











Предназначение:

Выполнение перемещений в различных типах токарных циклов нарезания резьбы, для разных стоек в общем случае повторяют одинаковую траекторию:

1. Подвод инструмента к стартовой точке или точке следующего захода в плоскости отвода на ускоренной подаче.
2. Перемещение к начальной точке резьбы на рабочей подаче.
3. Нарезание резьбы до конечной точки на рабочей подаче.
4. Возврат к плоскости отвода на подаче с ускоренным перемещением.
5. Если количество заходов (черновых и чистовых) не закончилось, вернуться к пункту 1.

Примеры таких циклов, Sinumerik: CYCLE97.

Использование формирователя позволяет по заданным параметрам: начальная и конечная точки резьбы, плоскость отвода, количество заходов резьбы, способ врезания, угол врезания и пр. сформировать траекторию движения инструмента. А так же добавить в траекторию движения на любом формируемом участке необходимые для конкретного цикла особенности, изменить тип или значение подачи, либо пропустить перемещение инструмента подготовленное формирователем.

Использование формирователя:

Рекомендуем создавать новый формирователь токарного цикла для каждого нового цикла в кадре управляющей программы, кроме случаев повторения работы цикла в новом кадре. Таким образом, интерфейс старого формирователя должен освобождаться при создании нового или завершении работы интерпретатора.

Для формирования команд движения инструмента по траектории цикла, необходимо выполнить следующие шаги:

1. Получить интерфейс формирователя (INCT_CLData.CreateLatheCycleFormer, INCT_GeomCLData.CreateLatheCycleFormer).
2. Задать индексы продольной и поперечной осей по которым будут производиться перемещения (Set_AxisIndexes).
3. Установить параметры цикла (Set_ThreadOrientation, Set_ThreadDepth, Set_ThreadStep, Set_ThreadStartAngle, Set_StartPoint и пр. описание см. ниже).
4. Назначить обработчик событий цикла (Set_Events), если в траекторию цикла нужно вносить изменения.
5. Сформировать команды движения инструмента по траектории цикла (MakeWorkpath).

Описание методов интерфейса:

Метод	Описание метода
Set_AxisIndexes	Метод устанавливает индексы продольной и поперечной осей по которым будут производиться перемещения при формировании траектории цикла. Входные параметры: Value (TNCLCAxisIndexes) - Transverse (целое число) - индекс поперечной оси, Longitudinal (целое число) - индекс продольной оси. Выходных параметров нет.
Get_AxisIndexes	Функция возвращает индексы продольной и поперечной осей в которых будут производиться перемещения при формировании траектории цикла (TNCLCAxisIndexes). Входных параметров нет.
Set_Direction	Метод устанавливает направление цикла нарезания резьбы. Входной параметр: Value (TNCThreadCuttingDirection) - направление цикла: drLongitudinal - продольное, drTransverse - поперечное. Выходных параметров нет.

Метод	Описание метода
Get_Direction	Функция возвращает направление цикла (TNCThreadCuttingDirection). Входных параметров нет.
Set_ThreadOrientation	Метод устанавливает тип ориентации резьбы. Входной параметр: Value (TNCThreadOrientation см. INCT_CLData.AddSinglePassThread) - тип ориентации резьбы. Выходных параметров нет.
Get_ThreadOrientation	Функция возвращает тип ориентации резьбы (TNCThreadOrientation). Входных параметров нет.
Set_ThreadDepth	Метод устанавливает значение глубины резьбы. Входной параметр: Value (вещественное число) - значение глубины резьбы. Выходных параметров нет.
Get_ThreadDepth	Функция возвращает значение глубины резьбы (вещественное число). Входных параметров нет.
Set_ThreadStep	Метод устанавливает значение шага резьбы. Входной параметр: Value (вещественное число) - значение шага резьбы. Выходных параметров нет.
Get_ThreadStep	Функция возвращает значение шага резьбы (вещественное число). Входных параметров нет.
Set_ThreadStartAngle	Метод устанавливает значение начального угла резьбы. Входной параметр: Value (вещественное число) - значение начального угла резьбы. Выходных параметров нет.

Метод	Описание метода
Get_ThreadStartAngle	Функция возвращает значение начального угла резьбы (вещественное число). Входных параметров нет.
Set_ThreadStartCount	Метод устанавливает количество заходов резьбы. Входной параметр: Value (целое число) - количество заходов резьбы. Выходных параметров нет.
Get_ThreadStartCount	Функция возвращает количество заходов резьбы (целое число). Входных параметров нет.
Set_StartPoint	Метод устанавливает значения координат начальной точки резьбы. Входной параметр: Value (TNC2DPoint) - значения координат начальной точки резьбы. Выходных параметров нет.
Get_StartPoint	Функция возвращает значения координат начальной точки резьбы (TNC2DPoint). Входных параметров нет.
Set_EndPoint	Метод устанавливает значения координат конечной точки резьбы. Входной параметр: Value (TNC2DPoint) - значения координат конечной точки резьбы. Выходных параметров нет.
Get_EndPoint	Функция возвращает значения координат конечной точки резьбы (TNC2DPoint). Входных параметров нет.
Set_TaperDist	Метод устанавливает значение конусности между начальной и конечной точкой резьбы. Входной параметр: Value (вещественное число) - значение конусности между

Метод	Описание метода
	начальной и конечной точкой резьбы. Выходных параметров нет.
Get_TaperDist	Функция возвращает значение значение конусности между начальной и конечной точкой резьбы (вещественное число). Входных параметров нет.
Set_ChamferLength	Метод устанавливает длину фаски для выхода инструмента в конце резьбы. Входной параметр: Value (вещественное число) - длина фаски для выхода инструмента в конце резьбы. Выходных параметров нет.
Get_ChamferLength	Функция возвращает длину фаски для выхода инструмента в конце резьбы (вещественное число). Входных параметров нет.
Set_ChamferAngle	Метод устанавливает значение угла фаски в градусах. Входной параметр: Value (вещественное число) - значение угла фаски в градусах. Выходных параметров нет.
Get_ChamferAngle	Функция возвращает значение угла фаски в градусах (вещественное число). Входных параметров нет.
Set_RetractionPath	Метод устанавливает уровень плоскости отвода. Входной параметр: Value (вещественное число) - уровень плоскости отвода. Выходных параметров нет.
Get_RetractionPath	Функция возвращает заданный уровень плоскости отвода (вещественное число). Входных параметров нет.

Метод	Описание метода
Set_RunInPath	Метод устанавливает значение входного участка. Входной параметр: Value (вещественное число) - значение входного участка. Выходных параметров нет.
Get_RunInPath	Функция возвращает значение входного участка (вещественное число). Входных параметров нет.
Set_RunOutPath	Метод устанавливает значение выходного участка. Входной параметр: Value (вещественное число) - значение выходного участка. Выходных параметров нет.
Get_RunOutPath	Функция возвращает значение выходного участка (вещественное число). Входных параметров нет.
Set_ThreadAngle	Метод устанавливает значение угла профиля резьбы. Входной параметр: Value (вещественное число) - значение угла профиля резьбы. Выходных параметров нет.
Get_ThreadAngle	Функция возвращает значение угла профиля резьбы (вещественное число). Входных параметров нет.
Set_InclinationAngle	Метод устанавливает значение угла одной из стенок профиля резьбы относительно вертикали. Входной параметр: Value (вещественное число) - значение. Выходных параметров нет.
Get_InclinationAngle	Функция возвращает значение угла одной из стенок профиля резьбы относительно

Метод	Описание метода
	вертикали (вещественное число). Входных параметров нет.
Set_SequenceType	Метод устанавливает способ врезания на черновых проходах. Входной параметр: Value (TNCThreadSequenceType) - способ врезания на черновых проходах. stRadial - радиально; stFlank - гранью; stModifiedFlank - под углом; stAlternateFlank - попеременно гранями. Выходных параметров нет.
Get_SequenceType	Функция возвращает значение способа врезания на черновых проходах (TNCThreadSequenceType). Входных параметров нет.
Set_PlungeAngle	Метод устанавливает значение угла врезания для стратегии stModifiedFlank. Входной параметр: Value (вещественное число) - значение угла врезания в градусах. Выходных параметров нет.
Get_PlungeAngle	Функция возвращает значение угла врезания для стратегии stModifiedFlank (вещественное число). Входных параметров нет.
Set_DepthIsEqual	Метод устанавливает способ определения глубины резания. Входной параметр: Value (логическое) - способ определения глубины резания. Истина - постоянная глубина, Ложь - постоянная площадь резания. Выходных параметров нет.
Get_DepthIsEqual	Функция возвращает способ определения глубины резания (логическое).

Метод	Описание метода
	Входных параметров нет.
Set_IsPassCountDefined	Метод устанавливает метод определения глубины резания. Входной параметр: Value (логическое) - метод определения глубины резания. Истина - по количеству проходов, Ложь - по начальной глубине. Выходных параметров нет.
Get_IsPassCountDefined	Функция возвращает метод определения глубины резания (логическое). Входных параметров нет.
Set_MinCutDepth	Метод устанавливает значение минимально возможной глубины чернового прохода. Входной параметр: Value (вещественное число) - значение минимально возможной глубины чернового прохода. Выходных параметров нет.
Get_MinCutDepth	Функция возвращает значение минимально возможной глубины чернового прохода (вещественное число). Входных параметров нет.
Set_FirstCutDepth	Метод устанавливает значение глубины первого чернового прохода. Входной параметр: Value (вещественное число) - значение глубины первого чернового прохода. Выходных параметров нет.
Get_FirstCutDepth	Функция возвращает значение глубины первого чернового прохода (вещественное число). Входных параметров нет.
Set_RoughPassCount	Метод устанавливает количество черновых проходов. Входной параметр: Value (целое число) -

Метод	Описание метода
	количество черновых проходов. Выходных параметров нет.
Get_RoughPassCount	Функция возвращает количество черновых проходов (целое число). Входных параметров нет.
Set_FinishPassCount	Метод устанавливает количество чистовых проходов. Входной параметр: Value (целое число) - количество чистовых проходов. Выходных параметров нет.
Get_FinishPassCount	Функция возвращает количество чистовых проходов (целое число). Входных параметров нет.
Set_FinishPassDepth	Метод устанавливает значение глубины чистовых проходов. Входной параметр: Value (вещественное число) - значение глубины чистовых проходов. Выходных параметров нет.
Get_FinishPassDepth	Функция возвращает значение глубины чистовых проходов (вещественное число). Входных параметров нет.
Set_Feedrate	Метод устанавливает значение рабочей подачи. Входной параметр: Value (вещественное число) - значение рабочей подачи. Выходных параметров нет.
Get_Feedrate	Функция возвращает значение рабочей подачи. Входных параметров нет.
Set_MPM	Метод устанавливает тип рабочей подачи. Входной параметр: Value (логическое) - тип рабочей подачи. Истина - мм/мин, Ложь -

Метод	Описание метода
	мм/об. Выходных параметров нет.
Get_MPM	Функция возвращает тип рабочей подачи. Входных параметров нет.
Set_Events	Метод назначает события для управления поведением цикла. При использовании событий можно добавить в траекторию движения на любом формируемом участке необходимые для конкретного цикла особенности: выстой, ориентированный останов и прочее, изменить тип или значение подачи, либо пропустить перемещение инструмента подготовленное формирователем. Входной параметр: Value (интерфейс) - объект, реализующий интерфейсы INCT_LatheCycleEvents, INCT_LatheThreadCuttingCycleEvents. Подробное описание ниже. Выходных параметров нет.
Get_Events	Функция возвращает объект, реализующий события, связанный с формирователем цикла (INCT_LatheCycleEvents, INCT_LatheThreadCuttingCycleEvents). Входных параметров нет.
MakeWorkpath	Метод формирует траекторию цикла по заданным параметрам.

INCT_LatheThreadCuttingCycleFormer2 - интерфейс CAM системы, который реализует функциональность по упрощению формирования перемещений токарного цикла нарезания резьбы.

Описание методов интерфейса:

Метод	Описание метода
Set_Approach	Метод устанавливает тип подвода. Входной параметр: Value (TNCThreadApproach) - тип подвода. taOutside - снаружи (по-умолчанию) taIndise - изнутри Выходных параметров нет.
Get_Approach	Функция возвращает тип подвода (TNCThreadApproach). Входных параметров нет.
Get_SignLongitudinal	Функция возвращает знак перемещения вдоль продольной оси (целое число).
Get_SignTransverse	Функция возвращает знак перемещения вдоль поперечной оси (целое число).

INCT_LatheCycleEvents - интерфейс реализуемый в интерпретаторе, который позволяет управлять поведением токарного цикла. Подробно описан в статье Токарный цикл точения канавок.

INCT_LatheThreadCuttingCycleEvents - интерфейс реализуемый в интерпретаторе, который позволяет управлять нарезанием резьбы. В интерфейсе описан метод-событие OnThread, который выполняется при выполнении нарезания резьбы внутри формирователя, и методы-события расчёта проходов: OnCalcBaseDepth, OnCalcCount, OnCalcNextDepth, OnCalcPoint.

Описание методов интерфейса:

Метод	Описание метода	Описание использования
OnThread	Метод будет выполнен перед каждым нарезанием резьбы при формировании траектории цикла. Реализация этого метода позволяет изменить параметры резьбы для выбранной итерации.	При необходимости замены параметров нарезания резьбы. Либо задания алгоритма изменения резьбы на различных заходах, не реализованных в формирователе.

Метод	Описание метода	Описание использования
	Входные параметры: Orient (TNCThreadOrientation) - ориентация резьбы; LeadType (TNCThreadLeadType) - способ задания шага резьбы; Step (вещественное число) - значение шага резьбы; StartAngle (вещественное число) - начальное угловое положение шпинделя в градусах (используется для многозаходной резьбы). См. INCT_CLData.AddSinglePassThread. Входные и выходные параметры: Handled (логическое) - признак выполнения перемещения внутри реализации (Истина - перемещение выполнено, Ложь - перемещение не выполнено). Если после вызова OnThread параметр Handled принимает значение Истина, это значит, что нарезание резьбы было выполнено внутри реализации метода OnThread и формирователь цикла не добавляет команду нарезания резьбы в траекторию движения инструмента. Если после вызова OnThread параметр Handled принимает значение Ложь, формирователь считает, что	

Метод	Описание метода	Описание использования
	нарезание резьбы не было выполнено внутри реализации метода OnThread и добавляет в траекторию движения инструмента команду нарезания резьбы.	
OnCalcBaseDepth	Метод будет выполнен для режима IsPassCountDefined = Истина после расчёта базовой (первой) глубины. Реализация этого метода позволяет изменить значение базовой глубины. Входной и выходной параметр: BaseDepth (вещественное число) - базовая глубина.	
OnCalcCount	Метод будет выполнен для режима IsPassCountDefined = Ложь после расчёта количества проходов. Реализация этого метода позволяет изменить значение количества проходов. Входной и выходной параметр: Count (целое число) - количество проходов.	
OnCalcNextDepth	Метод будет выполнен при расчёте очередной глубины (если количество проходов больше или равно одному).	

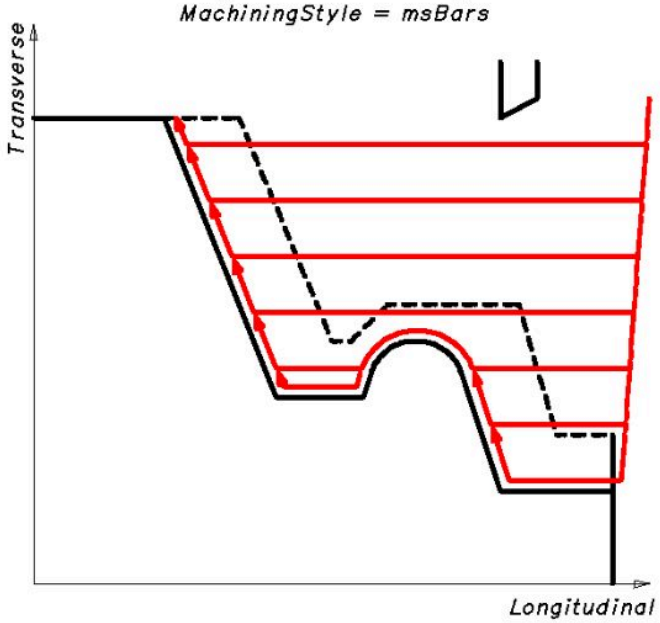
Метод	Описание метода	Описание использования
	Реализация этого метода позволяет изменить значение глубины для итерации. Входной параметр: Iteration (целое число) - номер итерации (прохода); Входной и выходной параметр: Depth (вещественное число) - глубина на итерации.	
OnCalcPoint	Метод будет выполнен при расчёте координат положения инструмента для очередного перемещения. Реализация этого метода позволяет изменить значение координат для итерации. Входной параметр: Iteration (целое число) - номер итерации (прохода); Входной и выходной параметр: Point (TNC2DPoint) - координаты положения инструмента на итерации.	

Токарный цикл обработки резанием

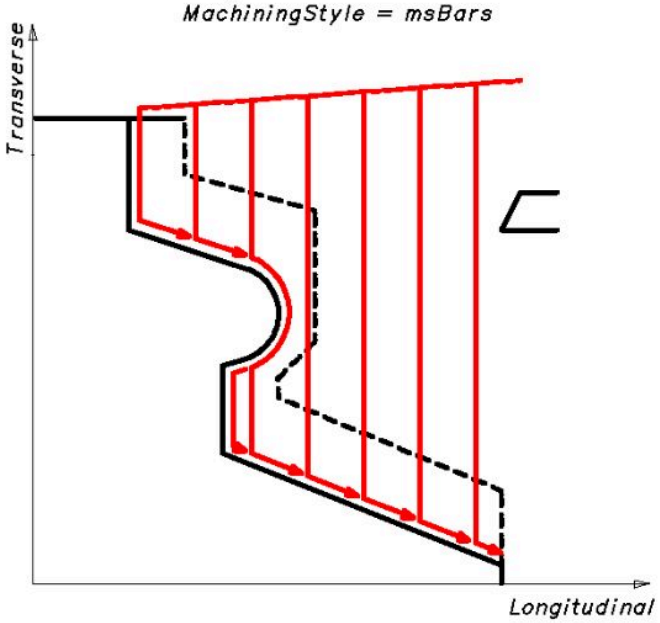
INCT_LatheStockRemovalCycleFormer - интерфейс CAM системы, который реализует функциональность по упрощению формирования перемещений токарного цикла точения по заданному контуру. Объект, реализующий этот интерфейс возвращает функция INCT_CLData.CreateLatheCycleFormer, INCT_GeomCLData.CreateLatheCycleFormer при её выполнении с параметром lctStockRemoval.

Интерфейс INCT_LatheStockRemovalCycleFormer2 расширяет возможности формирователя перемещений токарного цикла по заданному контуру. Описание новых свойств и методов представлено ниже. Объект, реализующий этот интерфейс возвращает функция CreateLatheCycleFormer (аналогично INCT_LatheStockRemovalCycleFormer).

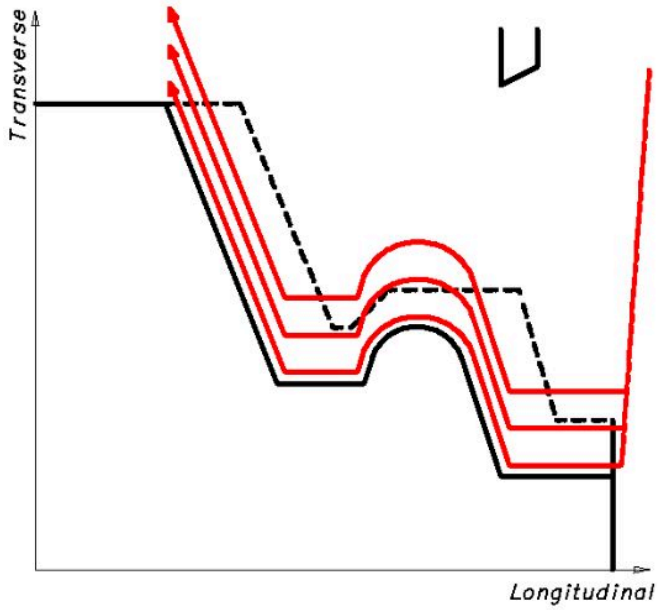
MachiningProcessing = mpRoughing
MachiningDirection = mdLongitudinal
MachiningStyle = msBars



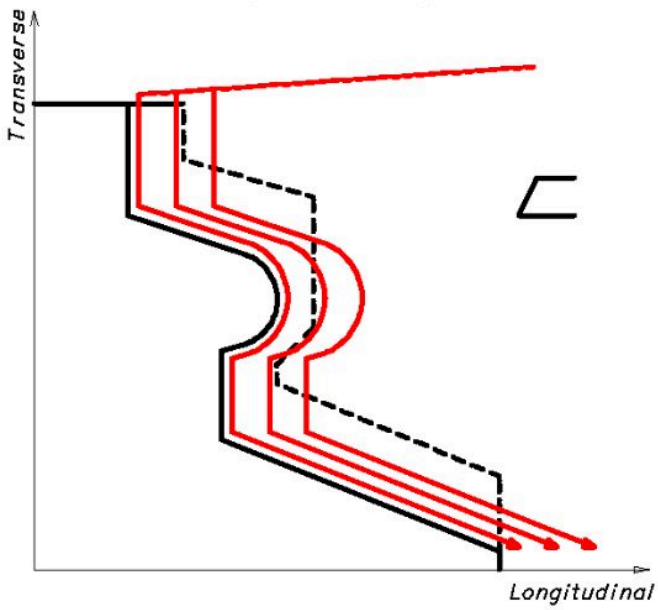
MachiningProcessing = mpRoughing
MachiningDirection = mdTransverse
MachiningStyle = msBars

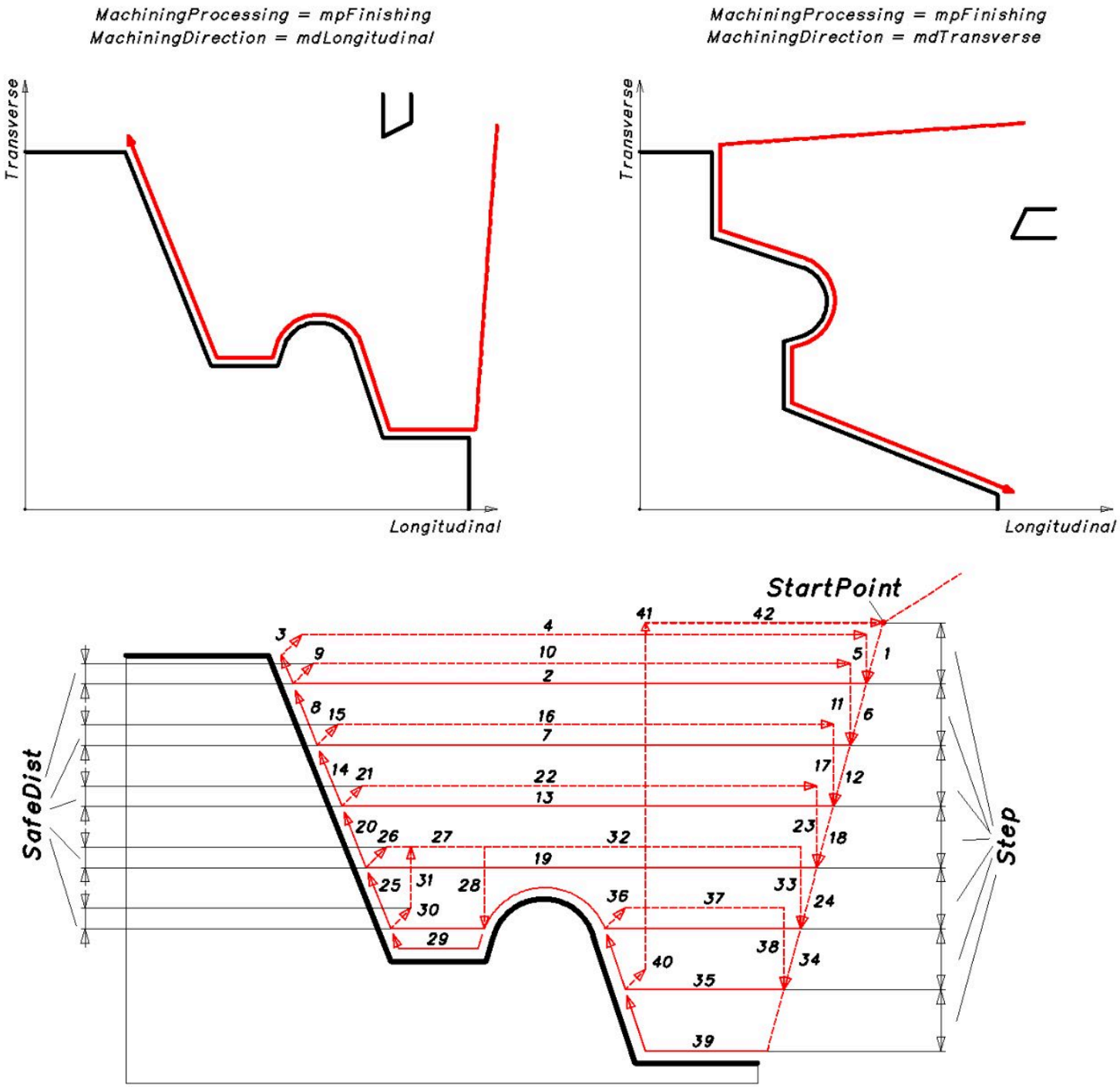


MachiningProcessing = mpRoughing
MachiningDirection = mdLongitudinal
MachiningStyle = msCopyContour



MachiningProcessing = mpRoughing
MachiningDirection = mdTransverse
MachiningStyle = msCopyContour





Предназначение:

Выполнение перемещений в различных типах токарных циклов точения по заданному контуру, для разных стоек в общем случае повторяют одинаковую траекторию:

1. Подвод инструмента к стартовой точке или точке следующего захода на ускоренной подаче.
2. Подвод к уровню чернового прохода на ускоренной подаче.
3. Подвод к точке чернового прохода параллельно продольной оси на рабочей подаче (черновой проход).

4. Возврат параллельно контуру вдоль контура с учётом чистового припуска до последней точки чернового прохода на рабочей подаче (завершение чернового прохода).
5. Возврат к плоскости отвода на подаче с ускоренным перемещением.
6. Возврат к пункту 1, если глубина обработки не достигнута.
7. Чистовая обработка контура (подвод к стартовой точке на ускоренной подаче, чистовая обработка вдоль контура на рабочей подаче, отвод на ускоренной подаче).

Примеры таких циклов, Sinumerik: CYCLE95.

Использование формирователя позволяет по заданным параметрам: траектория контура, плоскость отвода, шаг обработки, тип и направление обработки и пр. сформировать траекторию движения инструмента. А так же добавить в траекторию движения на любом формируемом участке необходимые для конкретного цикла особенности, изменить тип или значение подачи, либо пропустить перемещение инструмента подготовленное формирователем.

Использование формирователя:

Рекомендуем создавать новый формирователь токарного цикла для каждого нового цикла в кадре управляющей программы, кроме случаев повторения работы цикла в новом кадре. Таким образом, интерфейс старого формирователя должен освобождаться при создании нового или завершении работы интерпретатора.

Для формирования команд движения инструмента по траектории цикла, необходимо выполнить следующие шаги:

1. Получить интерфейс формирователя (INCT_CLData.CreateLatheCycleFormer, INCT_GeomCLData.CreateLatheCycleFormer).
2. Задать индексы продольной и поперечной осей по которым будут производиться перемещения (Set_AxisIndexes).
3. Установить параметры цикла (Set_NCProgram, Set_MachiningType, Set_MachiningDirection, Set_Overlap и пр., описание см. ниже).
4. Назначить обработчик событий цикла (Set_Events), если в траекторию цикла нужно вносить изменения.
5. Сформировать команды движения инструмента по траектории цикла (MakeWorkpath).

Описание методов интерфейса:

Метод	Описание метода
Set_AxisIndexes	Метод устанавливает индексы продольной и поперечной осей по которым будут

Метод	Описание метода
	производиться перемещения при формировании траектории цикла. Входные параметры: Value (TNCLCAxisIndexes) - Transverse (целое число) - индекс поперечной оси, Longitudinal (целое число) - индекс продольной оси. Выходных параметров нет.
Get_AxisIndexes	Функция возвращает индексы продольной и поперечной осей в которых будут производиться перемещения при формировании траектории цикла (TNCLCAxisIndexes). Входных параметров нет.
Set_Plane	Метод устанавливает текущую плоскость для управляющей программы контура, если плоскость не задана, то используется текущая плоскость на момент получения геометрии контура. Входные параметры: Value (TNCPlaneType) - текущая плоскость для управляющей программы контура. Выходных параметров нет.
Get_Plane	Функция возвращает текущую плоскость для управляющей программы контура. Входных параметров нет.
Set_NCProgram	Метод передаёт в формирователь интерфейс, реализующий текст контура управляющей программы.

Метод	Описание метода
	Входной параметр: Value (INCT_Program) - интерфейс, реализующий текст контура управляющей программы. Текст контура можно получить при помощи метода INCT_ProgramManager.GetPartByIndex, INCT_ProgramManager.GetPartByLabels. Выходных параметров нет.
Get_NCProgram	Функция возвращает интерфейс, реализующий текст контура управляющей программы (INCT_Program). Входных параметров нет.
Set_RecalcContourGeometry	Метод задаёт признак необходимости выполнения интерпретации управляющей программы контура. Входной параметр: Value (логический) - Ложь - не выполнять интерпретацию УП контура, если она уже выполнена, Истина - выполнить интерпретацию УП контура в любом случае.
Get_RecalcContourGeometry	Функция возвращает признак необходимости выполнения интерпретации управляющей программы контура. Входных параметров нет.
Set_MachiningProcessing	Метод устанавливает тип обработки. Входной параметр: Value (TNCMachiningProcessing) - тип обработки. mpRoughing - черновая (выполняется несколько проходов с учётом заданных параметров припуска, шага и пр.);

Метод	Описание метода
	mpFinishing - чистовая (выполняется только один проход по контуру без припуска). Выходных параметров нет.
Get_MachiningProcessing	Функция возвращает тип обработки. Входных параметров нет.
Set_MachiningType	Метод устанавливает тип обработки. Входной параметр: Value (TNCLatheMachiningType) - тип обработки. mtInside - внутри; mtOutside - снаружи. Выходных параметров нет.
Get_MachiningType	Функция возвращает тип обработки (TNCLatheMachiningType). Входных параметров нет.
Set_MachiningDirection	Метод устанавливает значение направления обработки. Входной параметр: Value (TNCLatheMachiningDirection) - значение направления обработки. mdLongitudinal - продольная; mdTransverse - поперечная. Выходных параметров нет.

Метод	Описание метода
Get_MachiningDirection	Функция возвращает значение направления обработки (TNCLatheMachiningDirection). Входных параметров нет.
Set_MachiningStyle	Метод устанавливает стиль обработки контура при черновой обработке. Входные параметры: Value (TNCLatheMachiningStyle) - стиль обработки. msBars - на каждой итерации обработка параллельно оси (продольной или поперечной в зависимости от MachiningDirection); msCopyContour - на каждой итерации обработка параллельно контуру (повтор контура). Выходных параметров нет.
Get_MachiningStyle	Функция возвращает стиль обработки контура при черновой обработке. Входных параметров нет.
Set_Overlap	Метод устанавливает значение типа перебега. Входной параметр: Value (логический) - значение типа перебега. Истина - с возвратом по контуру, Ложь - с прямым возвратом. Выходных параметров нет.
Get_Overlap	Функция возвращает значение типа перебега (логическое). Входных параметров нет.

Метод	Описание метода
Set_AxialStock	Метод устанавливает значение осевого припуска на контур. Входной параметр: Value (вещественное число) - значение осевого припуска на контур. Выходных параметров нет.
Get_AxialStock	Функция возвращает значение осевого припуска на контур (вещественное число). Входных параметров нет.
Set_RadialStock	Метод устанавливает значение радиального припуска на контур. Входной параметр: Value (вещественное число) - значение радиального припуска на контур. Выходных параметров нет.
Get_RadialStock	Функция возвращает значение радиального припуска на контур (вещественное число). Входных параметров нет.
Set_ProfileStock	Метод устанавливает значение эквидистантного припуска на контур. Входной параметр: Value (вещественное число) - значение эквидистантного припуска на контур. Выходных параметров нет.
Get_ProfileStock	Функция возвращает значение эквидистантного припуска на контур

Метод	Описание метода
	(вещественное число). Входных параметров нет.
Set_StartPointMode	Метод устанавливает режим задания начальной точки контура. Входные параметры: Value (TNCLatheStartPointMode) - режим задания начальной точки контура. spmCalcPoint - начальная точка рассчитывается циклом; spmSetPoint - начальная точка задаётся (см. StartPoint). Выходных параметров нет.
Get_StartPointMode	Функция возвращает режим задания начальной точки контура. Входных параметров нет.
Set_StartPoint	Метод устанавливает начальную точку цикла (для режима StartPointMode = spmSetPoint). Входные параметры: Value (TNC2DPoint) - начальная точка цикла. Выходных параметров нет.
Get_StartPoint	Функция возвращает начальную точку цикла. Входных параметров нет.
Set_EachReliefcutSeparately	Метод устанавливает значение признака обработки поднутрений. Входной параметр: Value (логическое) - Истина - каждое поднутрение обрабатывается отдельно, Ложь - все

Метод	Описание метода
	поднутрения обрабатываются за один проход. Выходных параметров нет.
Get_EachReliefcutSeparately	Функция возвращает значение признака обработки поднутрений. Входных параметров нет.
Set_DoReliefcutsInReverseOrder	Метод устанавливает признак порядка обработки поднутрений. Входные параметры: Value (логический) - признак порядка обработки поднутрений. Истина - обрабатывать поднутрения в обратном порядке. Ложь - обрабатывать поднутрения в прямом порядке. Выходных параметров нет.
Get_DoReliefcutsInReverseOrder	Функция возвращает признак порядка обработки поднутрений. Входных параметров нет.
Set_StepType	Метод устанавливает тип расчёта шага черновой обработки. Входной параметр: Value (TNCLatheStepType) - тип расчёта шага черновой обработки. stConst - при каждом проходе глубина обработки составляет значение шага обработки (Set_Step); stCalcEvenly - на каждом этапе (поднутрении) по значению шага обработки (Set_Step) формирователь вычисляет количество

Метод	Описание метода
	шагов, и выполняет обработку равными шагами (глубина реза / количество шагов). Выходных параметров нет.
Get_StepType	Функция возвращает тип шага черновой обработки (TNCLatheStepType). Входных параметров нет.
Set_Step	Метод устанавливает значение шага обработки. Входной параметр: Value (вещественное число) - значение шага обработки. Выходных параметров нет.
Get_Step	Функция возвращает значение шага обработки (вещественное число). Входных параметров нет.
Set_SafeDist	Метод устанавливает значение отвода для возврата (значение отскока в бок перед выводом инструмента). Входной параметр: Value (вещественное число) - значение отвода. Выходных параметров нет.
Get_SafeDist	Функция возвращает значение отвода для возврата (вещественное число). Входных параметров нет.
Set_LengthChipBreaking	Метод устанавливает значение длины хода для ломки стружки.

Метод	Описание метода
	Входной параметр: Value (вещественное число) - значение длины хода для ломки стружки. Выходных параметров нет.
Get_LengthChipBreaking	Функция возвращает значение длины хода для ломки стружки (вещественное число). Входных параметров нет.
Set_Feedrate	Метод устанавливает значение рабочей подачи. Входной параметр: Value (вещественное число) - значение рабочей подачи. Выходных параметров нет.
Get_Feedrate	Функция возвращает значение рабочей подачи. Входных параметров нет.
Set_MPM	Метод устанавливает тип рабочей подачи. Входной параметр: Value (логическое) - тип рабочей подачи. Истина - мм/мин, Ложь - мм/об. Выходных параметров нет.
Get_MPM	Функция возвращает тип рабочей подачи. Входных параметров нет.
Set_UseContourFeedrate	Метод устанавливает признак использования подачи, установленной в управляющей программе задания контура.

Метод	Описание метода
	Входные параметры: Value (логическое) - признак использования подачи. Истина - подача, установленная в контуре управляющей программы используется при обработке этих же участков контура. Ложь - подача, установленная в контуре игнорируется. Выходных параметров нет.
Get_UseContourFeedrate	Функция возвращает признак использования подачи, установленной в управляющей программе задания контура. Входных параметров нет.
Set_Events	Метод назначает события для управления поведением цикла. При использовании событий можно добавить в траекторию движения на любом формируемом участке необходимые для конкретного цикла особенности: выстой, ориентированный останов и прочее, изменить тип или значение подачи, либо пропустить перемещение инструмента подготовленное формирователем. Входной параметр: Value (интерфейс) - объект, реализующий интерфейсы INCT_LatheCycleEvents (подробно описан в статье Токарный цикл точения канавок), INCT_LatheCycleArcEvents (описан ниже), а также INCT_LatheStockRemovalCycleEvents, INCT_LatheStockRemovalCycleEvents2, INCT_LatheStockRemovalCycleEvents3 (описаны ниже). Формирователь запрашивает все эти интерфейсы у переданного объекта; объект может реализовать только нужные из них.

Метод	Описание метода
	Выходных параметров нет.
Get_Events	Функция возвращает объект, реализующий события, связанный с формирователем цикла (INCT_LatheCycleEvents, INCT_LatheCycleArcEvents). Входных параметров нет.
Get_SignLongitudinal	Функция возвращает знак продольной обработки (целое число). 1 - направление совпадает с продольной осью, 0 - направление не определено; -1 - направление обратное продольной оси. Входных параметров нет.
Get_SignTransverse	Функция возвращает знак поперечной обработки (целое число). 1 - направление совпадает с поперечной осью, 0 - направление не определено; -1 - направление обратное поперечной оси. Входных параметров нет.
MakeWorkpath	Метод формирует траекторию цикла по заданным параметрам. Процесс формирования траектории движения: 1. Получение текущего состояния интерпретатора (метод INCT_InterpreterState.Save), если интерпретатор поддерживает интерфейс INCT_InterpreterState (описан ниже). 2. Создание и инициализация нового экземпляра интерпретатора. 3. Восстановление текущего состояния для нового экземпляра интерпретатора

Метод	Описание метода
	(INCT_InterpreterState.Load). 4. Запуск нового интерпретатора по управляющей программе текста контура цикла, для формирования траектории контура. 5. После завершения работы нового интерпретатора, происходит формирование траектории движения инструмента по заданным параметрам и полученной траектории контура.

INCT_LatheStockRemovalCycleFormer2 расширяет возможности формирователя перемещений токарного цикла по заданному контуру.

Метод	Описание метода
Set_PassReturnMethod	Метод задаёт порядок выполнения возврата к точке захода. Входной параметр: Value (TNCPassReturnMethod) - порядок выполнения возврата к точке захода. prmStepwise - Для каждой итерации цикла возврат сначала выполняется по нормали к оси подачи, а затем по нормали к оси обработки; prmDirect - Для каждой итерации цикла возврат выполняется напрямую в точку следующего захода. Выходных параметров нет.
Get_PassReturnMethod	Функция возвращает порядок выполнения возврата к точке захода (TNCPassReturnMethod). Входных параметров нет.
Set_CutDepthCalcMethod	Метод задаёт способ определения глубины резания. Входной параметр: Value (TNCCutDepthCalcMethod) - способ определения глубины резания. cdcByStep -

Метод	Описание метода
	Глубина каждого реза равна или меньше (на последней итерации цикла) заданному шагу обработки; cdcByCount - Глубина каждого реза рассчитывается на основе заданного количества повторов цикла. Выходных параметров нет.
Get_CutDepthCalcMethod	Функция возвращает способ определения глубины резания (TNCCutDepthCalcMethod).
Set_Count	Метод задаёт количество повторений цикла (для способа определения глубины резания - по количеству повторов cdcByCount) Входной параметр: Value (целое число) - количество повторений. Выходных параметров нет.
Get_Count	Функция возвращает количество повторений цикла (для способа определения глубины резания - по количеству повторов cdcByCount). Входных параметров нет.
Set_Offset	Метод задаёт смещение точки начала обработки контура на каждом проходе. Входной параметр: Value (вещественное число) - смещение точки начала контура. Выходных параметров нет.
Get_Offset	Функция возвращает смещение точки начала обработки контура на каждом проходе. Входных параметров нет.
Get_ContourStartPoint	Функция возвращает координаты точки начала контура с учётом заданного в формирователе припуска (TNC2DPoint)
Get_ContourEndPoint	Функция возвращает координаты точки окончания контура с учётом заданного в формирователе припуска (TNC2DPoint)

Метод	Описание метода
Set_AxialStockDependsOnContour	Метод задаёт способ учёта осевого и радиального припусков (Set_AxialStock, Set_RadialStock). Входной параметр: Value (логическое). Истина (по умолчанию) - припуски берутся по модулю, а направление смещения определяется контуром и направлением обработки; Ложь - припуски считаются сдвигами вдоль осей с заданным знаком. Выходных параметров нет.
Get_AxialStockDependsOnContour	Функция возвращает способ учёта осевого и радиального припусков (логическое). Входных параметров нет.
Set_StartPointContainsSafeDist	Метод задаёт, учитывать ли значение отвода (Set_SafeDist) при расчёте начальной точки цикла (для режима StartPointMode = spmCalcPoint). Входной параметр: Value (логическое). Истина (по умолчанию) - начальная точка отодвигается от контура с учётом припусков и значения отвода; Ложь - только с учётом припусков. Выходных параметров нет.
Get_StartPointContainsSafeDist	Функция возвращает признак учёта значения отвода при расчёте начальной точки цикла (логическое). Входных параметров нет.

INCT_LatheCycleArcEvents - интерфейс, реализуемый в интерпретаторе, который позволяет управлять поведением токарного цикла при построении дуг. Интерфейс представляет из себя один метод-событие, которые выполняется при перемещении инструмента по дуге внутри формирователя траектории цикла. Используя метод интерфейса можно добавить в траекторию движения на любом формируемом участке необходимые для конкретного цикла особенности: выстой, ориентированный останов и прочее, изменить тип подачи, либо пропустить перемещение инструмента подготовленное формирователем.

Описание метода интерфейса:

Метод	Описание метода	Описание использования
OnArc	Метод будет выполнен перед каждым перемещением инструмента по дуге при формировании траектории цикла. Входные параметры: MoveType (TNCLatheCycleMoveType) - тип перемещения, описание см. Токарный цикл точения канавок; Входные и выходные параметры: Center (TNC2DPoint) - координаты центра дуги; Point (TNC2DPoint) - координаты конечной точки дуги; Radius (вещественное число) - радиус дуги; Feed (TNCFeedType) - тип подачи; Handled (логическое) - признак выполнения перемещения внутри реализации (Истина - перемещение выполнено, Ложь - перемещение не выполнено). Если после вызова OnArc параметр Handled принимает значение Истина, это значит, что перемещение было выполнено внутри реализации метода OnArc и формирователь цикла не добавляет команду перемещения по дуге в траекторию движения инструмента.	При необходимости переопределения перемещения по дуге или подачи, либо для отказа от перемещения (пропуска). Например, если нужно изменить тип подачи при проходе вдоль контура, то в реализации метода OnArc необходимо изменить значение возвращаемого методом параметра Feed по условию MoveType = lmOverlapContour (типы подачи см. в описании OutStandardFeed). Если нужно выполнить перемещение инструмента в точку, отличную от рассчитанной циклом, необходимо изменить значения возвращаемых методом OnArc параметров Center, Point, Radius. Если перемещение в рассчитанную циклом точку выполнять не нужно, тогда внутри реализации метода OnArc нужно изменить значение параметра Handled на Истина.

Метод	Описание метода	Описание использования
	Если после вызова OnArc параметр Handled принимает значение Ложь, формирователь считает, что перемещение по дуге не было выполнено внутри реализации метода OnArc и добавляет в траекторию движения инструмента команду перемещения по дуге в точку Point с типом подачи Feed.	

INCT_LatheStockRemovalCycleEvents - интерфейс, реализуемый в интерпретаторе, который позволяет управлять поведением токарного цикла обработки резанием. Интерфейс представляет из себя один метод-событие, которые выполняется после получения геометрии контура и перед расчётом траектории цикла. Используя метод интерфейса и интерфейс INCT_InterpreterState можно получить из текста управляющей программы параметры контура и передать их формирователю траектории цикла для корректного формирования (например, направление обработки, если оно задаётся командой в траектории контура).

Описание методов интерфейса:

Метод	Описание метода	Описание использования
OnCalcContourGeometry	Метод будет выполнен после получения геометрии контура и перед расчётом траектории цикла формирователем Параметров нет.	Используя метод интерфейса и интерфейс INCT_InterpreterState можно получить из текста управляющей программы параметры контура и передать их формирователю траектории цикла для корректного формирования (например, направление обработки, если оно задаётся командой в траектории контура).

INCT_LatheStockRemovalCycleEvents2 - интерфейс, реализуемый в интерпретаторе, который позволяет управлять поведением токарного цикла обработки резанием. Расширяет интерфейс INCT_LatheStockRemovalCycleEvents.

Описание методов интерфейса:

Метод	Описание метода	Описание использования
OnCalcNextStep	Метод выполняется при расчёте траектории движения инструмента для каждого прохода. Предназначен для изменения шага и смещения, используемых при расчёте следующей точки начала прохода по контуру и точки возврата после прохода по контуру для текущей итерации. Входные параметры: Iteration (Целое число) - номер прохода, начиная с 1; PointType (TNCLSRPointType) - тип точки контура: ptProfilePoint - точка начала прохода по контуру; ptReturnPoint - точка возврата после прохода по контуру. Входные и выходные параметры: Step (Вещественное число) - шаг для расчёта текущего прохода (Ось врезания); Offset (Вещественное число) - смещение для расчёта текущего прохода (вторая ось, перпендикулярная оси врезания).	Используя метод интерфейса можно задать собственный алгоритм расчёта точек начала прохода по контуру и возврата после прохода по контуру для каждого прохода, оперируя шагом и смещением на текущей итерации.

Метод	Описание метода	Описание использования
OnCalcNextPoint	Метод выполняется при расчёте траектории движения инструмента для каждого прохода. Предназначен для изменения координат точек начала прохода по контуру и возврата после прохода по контуру, рассчитанных для текущей итерации. Входные параметры: Iteration (Целое число) - номер прохода, начиная с 1; PointType (TNCLSRPointType) - тип точки контура: ptProfilePoint - точка начала прохода по контуру; ptReturnPoint - точка возврата после прохода по контуру. Входные и выходные параметры: Point (TNC2DPoint) - координаты точки.	Используя метод интерфейса можно задать собственный алгоритм расчёта точек начала прохода по контуру и возврата после прохода по контуру для каждого прохода, оперируя координатами точек на текущей итерации.

INCT_LatheStockRemovalCycleEvents3 - интерфейс, реализуемый в интерпретаторе, который позволяет управлять поведением токарного цикла обработки резанием. Расширяет интерфейс INCT_LatheStockRemovalCycleEvents2.

Описание методов интерфейса:

Метод	Описание метода	Описание использования
OnCalcSafeCut	Метод выполняется при расчёте траектории движения инструмента. Предназначен для изменения рассчитанного внутри цикла значений отвода для возврата (значений отскока в бок перед выводом	Перед вызовом метода цикл рассчитывает отвод по значению Set_SafeDist вдоль обеих осей в сторону от контура. Используя метод интерфейса можно задать собственное значение отвода, например отвод

Метод	Описание метода	Описание использования
	инструмента). Входные и выходные параметры: Point (TNC2DPoint) - координаты точки (относительные).	только вдоль одной оси. Новое значение используется при подводе к началу обработки, при возврате на безопасное расстояние после прохода и при расчёте уровней проходов.

INCT_InterpreterState - интерфейс, реализуемый интерпретатором (тем же объектом, который реализует интерфейс INCT_Interpreter), который позволяет сохранять текущее состояние интерпретатора и загружать его во вновь созданный экземпляр интерпретатора для корректного формирования траектории контура.

Например, для формирования траектории контура необходимо знать текущую плоскость. Для этого интерпретатор должен создать объект, сохранить в этот объект текущую плоскость (метод Save), а при создании нового экземпляра интерпретатора восстановить текущую плоскость из переданного ему объекта (метод Load).

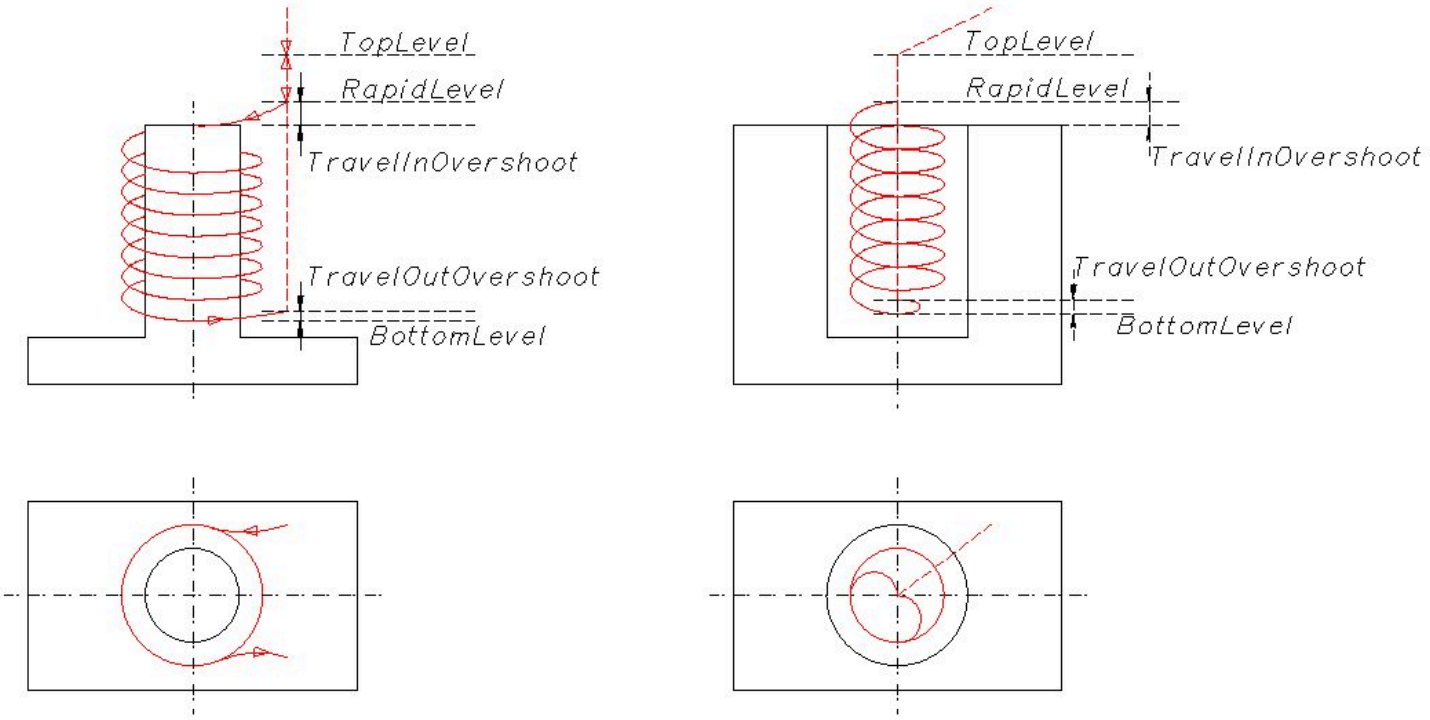
Описание методов интерфейса:

Метод	Описание метода	Описание использования
Save	Метод должен создать объект, реализующий свой собственный интерфейс, сохранить в объект данные текущего состояния, передать интерфейс на объект в выходном параметре. Входных параметров нет. Выходной параметр: State - интерфейс IUnknown.	Для формирования траектории контура необходимо знать текущую плоскость, заданную в тексте управляющей программы. При реализации метода Save необходимо сохранить текущую плоскость в созданный внутри метода объект. Ссылка на объект будет передана ядром CAM системы в новый экземпляр интерпретатора, путём вызова метода Load. При реализации которого необходимо получить
Load	Метод должен получить собственный интерфейс от переданного во входном	

Метод	Описание метода	Описание использования
	параметре объекта, получить текущее состояние интерпретатора, необходимое для формирования траектории контура. Входной параметр: State - интерфейс IUnknown. Выходных параметров нет.	значение текущей плоскости из переданного объекта.

Цикл фрезерования резьбы

INCT_ThreadMillingCycleFormer - интерфейс CAM системы, который реализует функциональность по упрощению формирования перемещений цикла фрезерования резьбы. Объект, реализующий этот интерфейс, возвращает функция INCT_CLData.CreateMillingCycleFormer при её выполнении с параметром mctThreadMilling.



Предназначение:

Выполнение перемещений в различных типах цикла фрезерования резьбы, для разных стоек в общем случае повторяют одинаковую траекторию:

1. Подвод инструмента на начальный уровень на ускоренной подаче.
2. Подвод инструмента на уровень перебега при подводе на ускоренной подаче.
3. Подвод по дуге или хелику к точке начала фрезерования на рабочей подаче (перебег).
4. Фрезерование резьбы с заданными параметрами (центр, диаметр, шаг) до уровня окончания фрезерования.
5. Отвод по дуге или хелику на рабочей подаче на уровне перебега при отводе (перебег).
6. Возврат на начальный уровень на ускоренной подаче.

Примеры таких циклов, Sinumerik: CYCLE90.

Использование формирователя позволяет сформировать траекторию движения инструмента по заданным параметрам. А также добавить в траекторию движения на любом формируемом

участке необходимые для конкретного цикла особенности, изменить тип или значение подачи, либо пропустить перемещение инструмента, подготовленное формирователем.

Использование формирователя:

Рекомендуем создавать новый формирователь для каждого нового цикла в кадре управляющей программы. Таким образом, интерфейс старого формирователя должен освобождаться при создании нового или в случае завершения работы интерпретатора.

Для формирования команд движения инструмента по траектории цикла, необходимо выполнить следующие шаги:

1. Получить интерфейс формирователя (INCT_CLData.CreateMillingCycleFormer).
2. Задать индексы осей, по которым будут производиться перемещения (Set_AxisIndexes).
3. Задать текущую плоскость (Set_Plane).
4. Задать рабочую подачу (Set_Feedrate).
5. Задать тип резьбы (Set_ThreadType).
6. Задать направление резьбы (Set_ThreadDirection).
7. Задать шаг резьбы (Set_ThreadStep).
8. Задать диаметр резьбы (Set_Diameter).
9. Задать точку центра резьбы (Set_CenterPoint).
10. Задать уровень подвода (Set_TopLevel).
11. Задать уровень безопасной плоскости (Set_RapidLevel).
12. Задать уровень окончания фрезерования (Set_BottomLevel).
13. Задать уровень перебега при подводе (Set_TravelInOvershoot).
14. Задать уровень перебега при отводе (Set_TravelOutOvershoot).
15. Назначить обработчик событий цикла (Set_Events), если в траекторию цикла нужно вносить изменения.
16. Сформировать команды движения инструмента по траектории цикла (MakeWorkpath).

Описание методов интерфейса:

Метод	Описание метода
Set_AxisIndexes	Метод устанавливает индексы осей X, Y, Z в которых будут производиться перемещения при формировании траектории цикла. Входные параметры: Value (TNCDCAxisIndexes) - X (целое число) - индекс оси X, Y (целое число) - индекс оси Y, Z (целое число) - индекс оси Z. Выходных параметров нет.

Метод	Описание метода
Get_AxisIndexes	Функция возвращает индексы осей X, Y, Z в которых будут производиться перемещения при формировании траектории цикла (TNCDCAxisIndexes). Входных параметров нет.
Set_Plane	Метод устанавливает текущую плоскость. Входной параметр: Value (TNCPlaneType) - текущая плоскость. ptXY - плоскость XY, фрезерование резьбы производится вдоль оси Z. ptYZ - плоскость YZ, фрезерование резьбы производится вдоль оси X. ptZX - плоскость ZX, фрезерование резьбы производится вдоль оси Y. Выходных параметров нет.
Get_Plane	Функция возвращает текущую плоскость (TNCPlaneType). Входных параметров нет.
Set_ThreadType	Метод устанавливает тип резьбы. Входной параметр: Value (TNCMillingThreadType) - тип резьбы. mttInternal - внутренняя mttExternal - наружная Выходных параметров нет.
Get_ThreadType	Функция возвращает тип резьбы (TNCMillingThreadType). Входных параметров нет.
Set_ThreadDirection	Метод устанавливает направление резьбы. Входной параметр: Value (TNCCircularMode) - направление резьбы. cmCW - по часовой стрелке cmCCW - против часовой стрелки Выходных параметров нет.

Метод	Описание метода
Get_ThreadDirection	Функция возвращает значение направления резьбы (TNCCircularMode). Входных параметров нет.
Set_ThreadStep	Метод устанавливает значение шага резьбы. Входной параметр: Value (вещественное число) - значение шага резьбы. Выходных параметров нет.
Get_ThreadStep	Функция возвращает значение шага резьбы (вещественное число). Входных параметров нет.
Set_Diameter	Метод устанавливает значение диаметра резьбы. Входной параметр: Value (вещественное число) - значение диаметра резьбы. Выходных параметров нет.
Get_Diameter	Функция возвращает значение диаметра резьбы (вещественное число). Входных параметров нет.
Set_CenterPoint	Метод устанавливает точку центра резьбы. Входной параметр: Value (TNC2DPoint) - точка центра резьбы. Выходных параметров нет.
Get_CenterPoint	Функция возвращает точку центра резьбы (TNC2DPoint). Входных параметров нет.
Set_TopLevel	Метод устанавливает уровень подвода. Входной параметр: Value (вещественное число) - значение уровня подвода. Выходных параметров нет.
Get_TopLevel	Функция возвращает значение уровня подвода (вещественное число). Входных параметров нет.

Метод	Описание метода
Set_RapidLevel	Метод устанавливает значение уровня безопасной плоскости. Входной параметр: Value (вещественное число) - значение уровня безопасной плоскости. Выходных параметров нет.
Get_RapidLevel	Функция возвращает значение уровня безопасной плоскости (вещественное число). Входных параметров нет.
Set_BottomLevel	Метод устанавливает значение уровня окончания фрезерования. Входной параметр: Value (вещественное число) - значение уровня окончания фрезерования. Выходных параметров нет.
Get_BottomLevel	Функция возвращает значение уровня окончания фрезерования (вещественное число). Входных параметров нет.
Set_TravelInOvershoot	Метод устанавливает уровень перебега при подводе. Входной параметр: Value (вещественное число) - уровень перебега при подводе. Выходных параметров нет.
Get_TravelInOvershoot	Функция возвращает уровень перебега при подводе (вещественное число). Входных параметров нет.
Set_TravelOutOvershoot	Метод устанавливает уровень перебега при отводе. Входной параметр: Value (вещественное число) - значение перебега при отводе. Выходных параметров нет.

Метод	Описание метода
Get_TravelOutOvershoot	Функция возвращает уровень перебега при отводе (вещественное число). Входных параметров нет.
Set_Feedrate	Метод устанавливает значение рабочей подачи. Входной параметр: Value (вещественное число) - значение рабочей подачи. Выходных параметров нет.
Get_Feedrate	Функция возвращает значение рабочей подачи (вещественное число). Входных параметров нет.
Set_MPM	Метод устанавливает тип рабочей подачи. Входной параметр: Value (логическое) - тип рабочей подачи. Истина - мм/мин, Ложь - мм/об. Выходных параметров нет.
Get_MPM	Функция возвращает тип рабочей подачи (логическое). Входных параметров нет.
Set_Events	Метод назначает события для управления поведением цикла. При использовании событий можно добавить в траекторию движения на любом формируемом участке необходимые для конкретного цикла особенности, изменить тип или значение подачи, либо пропустить перемещение инструмента, подготовленное формирователем. Входной параметр: Value (INCT_MillingCycleEvents) - объект, реализующий события. Подробное описание ниже. Выходных параметров нет.

Метод	Описание метода
Get_Events	Функция возвращает объект, реализующий события, связанный с формирователем цикла (INCT_MillingCycleEvents). Входных параметров нет.
MakeWorkpath	Метод формирует траекторию цикла по заданным параметрам.

INCT_MillingCycleEvents - интерфейс, реализуемый в интерпретаторе, который позволяет управлять поведением цикла фрезерования резьбы. Интерфейс представляет из себя набор методов-событий, которые выполняются при перемещении инструмента внутри формирователя траектории цикла. Используя методы интерфейса можно добавить в траекторию движения на любом формируемом участке необходимые для конкретного цикла особенности: выстой, ориентированный останов и прочее, изменить тип подачи, либо пропустить перемещение инструмента, подготовленное формирователем.

Если для формирователя траектории цикла назначены события, то любое перемещение инструмента внутри формирователя состоит из 3-х этапов:

1. Выполняется метод INCT_MillingCycleEvents.BeforeMove. В качестве параметров методу передаются заданные циклом: тип перемещения, координаты точки перемещения, тип подачи.
2. Выполняется метод INCT_MillingCycleEvents.OnMove/OnArc в зависимости от типа перемещения: по прямой или по дуге.
   Для OnMove в качестве параметров методу передаются: тип перемещения, координаты точки перемещения, тип подачи, признак выполнения перемещения. Все параметры, кроме типа перемещения, в реализации метода могут быть изменены. В зависимости от заданного в OnMove признака, производится выполнение перемещения инструмента в заданную точку с указанной подачей, или перемещение не производится.
   Для OnArc в качестве параметров методу передаются: тип перемещения, координаты центра дуги, точки перемещения, радиус, тип подачи, признак выполнения перемещения. Все параметры, кроме типа перемещения, в реализации метода могут быть изменены. В зависимости от заданного в OnArc признака, производится выполнение перемещения инструмента в заданную точку с указанной подачей, или перемещение не производится.
3. Выполняется метод INCT_MillingCycleEvents.AfterMove. В качестве параметров методу передаются: тип перемещения, координаты точки перемещения, тип подачи, возвращённые методом OnMove/OnArc.

Описание методов интерфейса:

Метод	Описание метода	Описание использования
BeforeMove	Метод будет выполнен перед каждым перемещением инструмента при формировании траектории цикла. Входные параметры: MoveType (TNCMillingCycleMoveType) - тип перемещения, описание см. ниже; Point (TNC3DPoint) - координаты точки перемещения; Feed (TNCFeedType) - тип подачи. mmMoveToTopLevel - перемещение к стартовой точке на уровне TopLevel mmMoveToRapidLevel - перемещение к уровню начала подвода mmTravelInMovement - подвод к резьбе по дуге mmThreadMilling - нарезание резьбы mmTravelOutMovement - отвод от резьбы к центру по дуге mmReturnToTopLevel - возврат на TopLevel Выходных параметров нет.	При необходимости добавления тех или иных команд перед выполнением перемещения, их нужно добавить в реализацию метода. Например, если есть необходимость включить систему охлаждения станка перед фрезерованием резьбы, сделать это можно добавив в реализацию метода BeforeMove вызов INCT_CLData.AddCoolant с соответствующими параметрами по условию MoveType = mmMoveToTopLevel.
OnMove	Метод будет выполнен непосредственно перед каждым прямолинейным перемещением инструмента при формировании траектории цикла. Входные параметры: MoveType (TNCMillingCycleMoveType) - тип перемещения, описание	При необходимости переопределения значений точки перемещения или подачи, либо для отказа от перемещения (пропуска). Например, если нужно изменить тип подачи на ускоренную, при возврате на начальный уровень, после фрезерования, то в

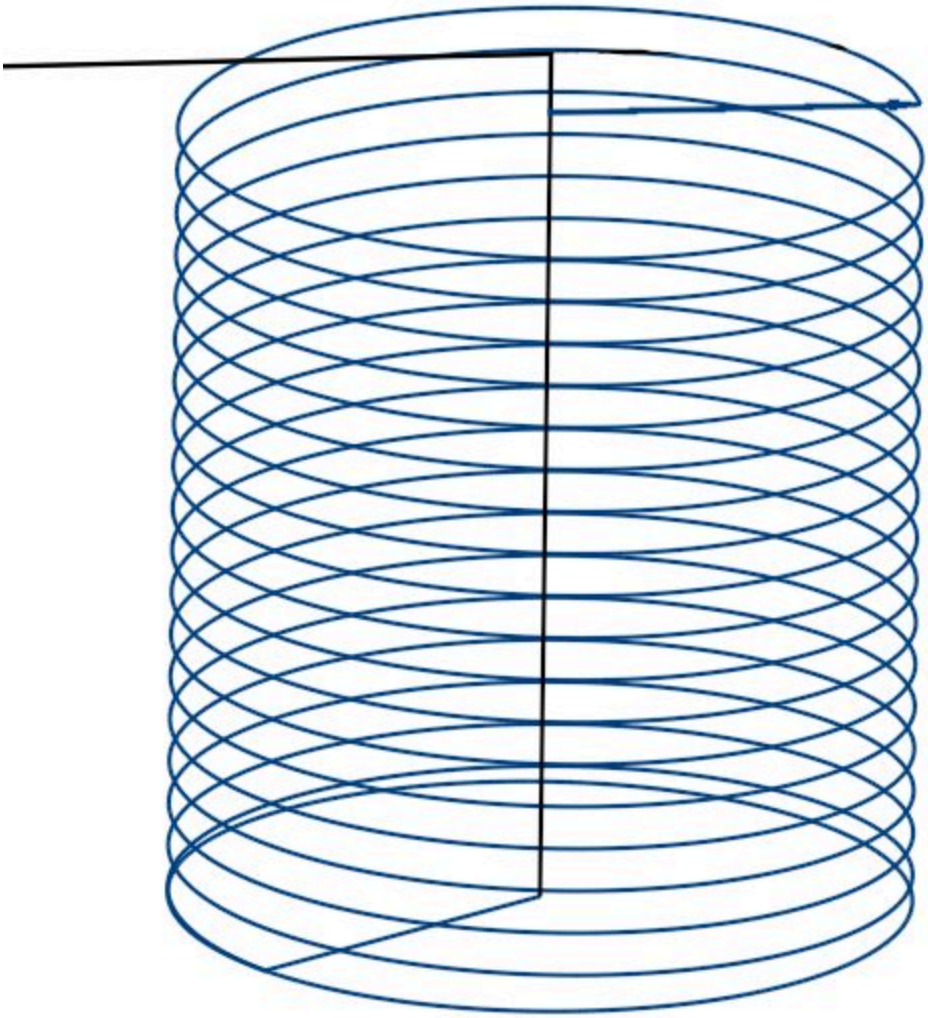
Метод	Описание метода	Описание использования
	см. выше; Входные и выходные параметры: Point (TNC3DPoint) - координаты точки перемещения; Feed (TNCFeedType) - тип подачи; Handled (логическое) - признак выполнения перемещения внутри реализации (Истина - перемещение выполнено, Ложь - перемещение не выполнено). Если после вызова OnMove параметр Handled принимает значение Истина, это значит, что перемещение было выполнено внутри реализации метода OnMove и формирователь цикла не добавляет команду перемещения в траекторию движения инструмента. Если после вызова OnMove параметр Handled принимает значение Ложь, формирователь считает, что перемещение не было выполнено внутри реализации метода OnMove и добавляет в траекторию движения инструмента команду перемещения в точку Point с типом подачи Feed.	реализации метода OnMove необходимо изменить значение возвращаемого методом параметра Feed на ffRapid по условию MoveType = mmReturnToTopLevel (типы подачи см. в описании OutStandardFeed). Если нужно выполнить перемещение инструмента в точку, отличную от рассчитанной циклом, необходимо изменить значение возвращаемого методом OnMove параметра Point. Если перемещение в рассчитанную циклом точку выполнять не нужно, тогда внутри реализации метода OnMove нужно изменить значение параметра Handled на Истина.
OnArc	Метод будет выполнен непосредственно перед каждым перемещением	При необходимости переопределения значений точки перемещения или

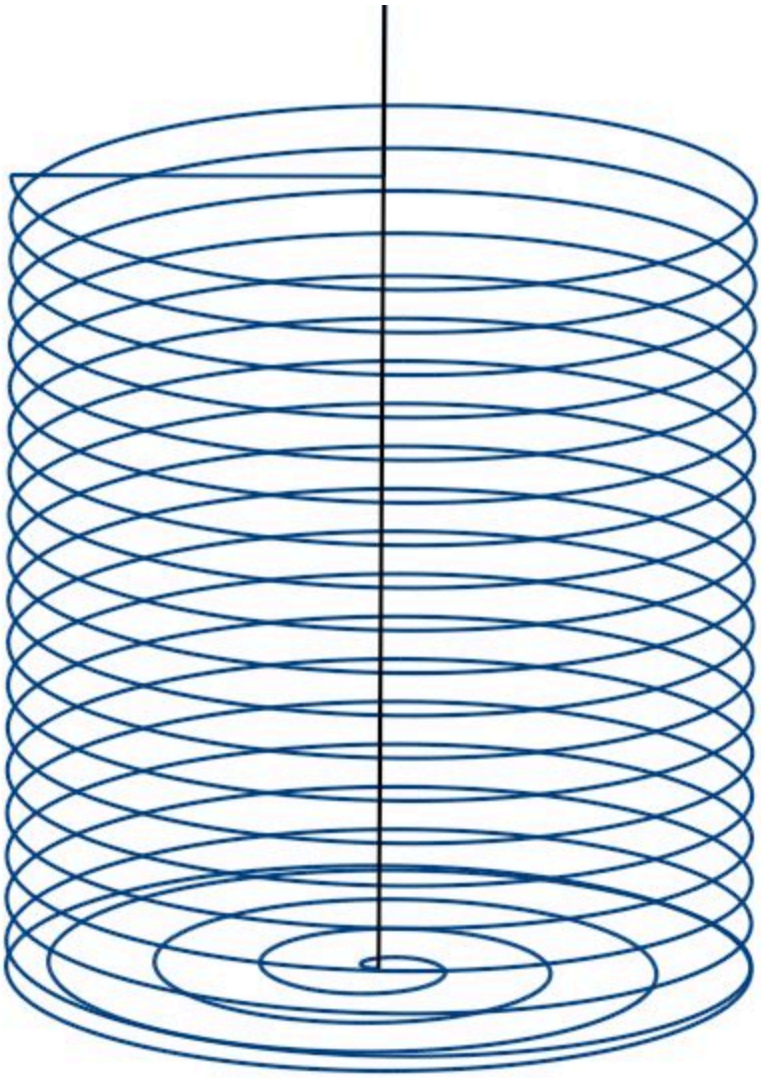
Метод	Описание метода	Описание использования
	инструмента по дуге при формировании траектории цикла. Входные параметры: MoveType (TNCMillingCycleMoveType) - тип перемещения, описание см. выше; Входные и выходные параметры: Center, Point (TNC3DPoint) - координаты центра и точки перемещения; Radius (вещественное число) - радиус дуги, Feed (TNCFeedType) - тип подачи; Handled (логическое) - признак выполнения перемещения внутри реализации (Истина - перемещение выполнено, Ложь - перемещение не выполнено). Если после вызова OnArc параметр Handled принимает значение Истина, это значит, что перемещение было выполнено внутри реализации метода OnArc и формирователь цикла не добавляет команду перемещения в траекторию движения инструмента. Если после вызова OnArc параметр Handled принимает значение Ложь, формирователь считает, что перемещение не было выполнено внутри	подачи, либо для отказа от перемещения (пропуска).

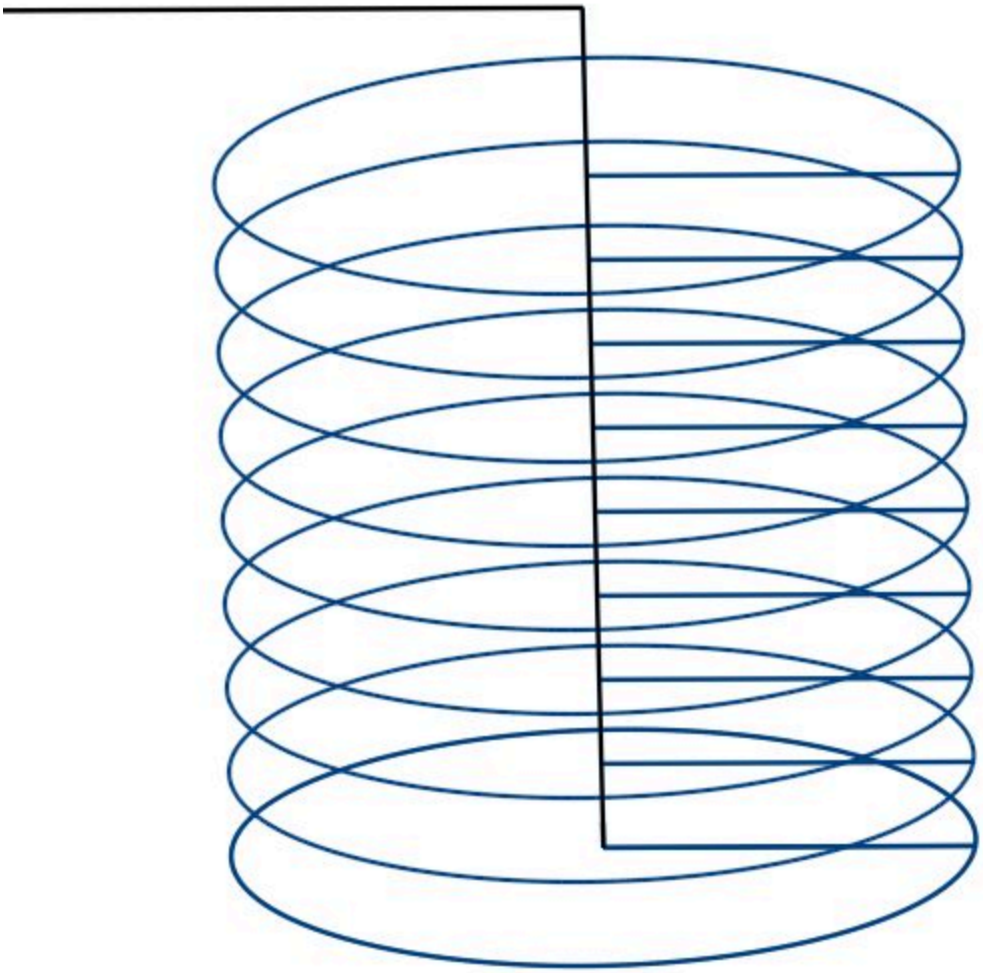
Метод	Описание метода	Описание использования
	реализации метода OnArc и добавляет в траекторию движения инструмента команду перемещения по дуге в точку Point с типом подачи Feed.	
AfterMove	Метод будет выполнен после каждого перемещения инструмента при формировании траектории цикла, даже если перемещение было отменено в OnMove/OnArc. Входные параметры: MoveType (TNCMillingCycleMoveType) - тип перемещения, описание см. выше; Point (TNC3DPoint) - координаты точки перемещения; Feed (TNCFeedType) - тип подачи. Выходных параметров нет.	При необходимости добавления тех или иных команд после выполнения перемещения, их нужно добавить в реализацию метода.

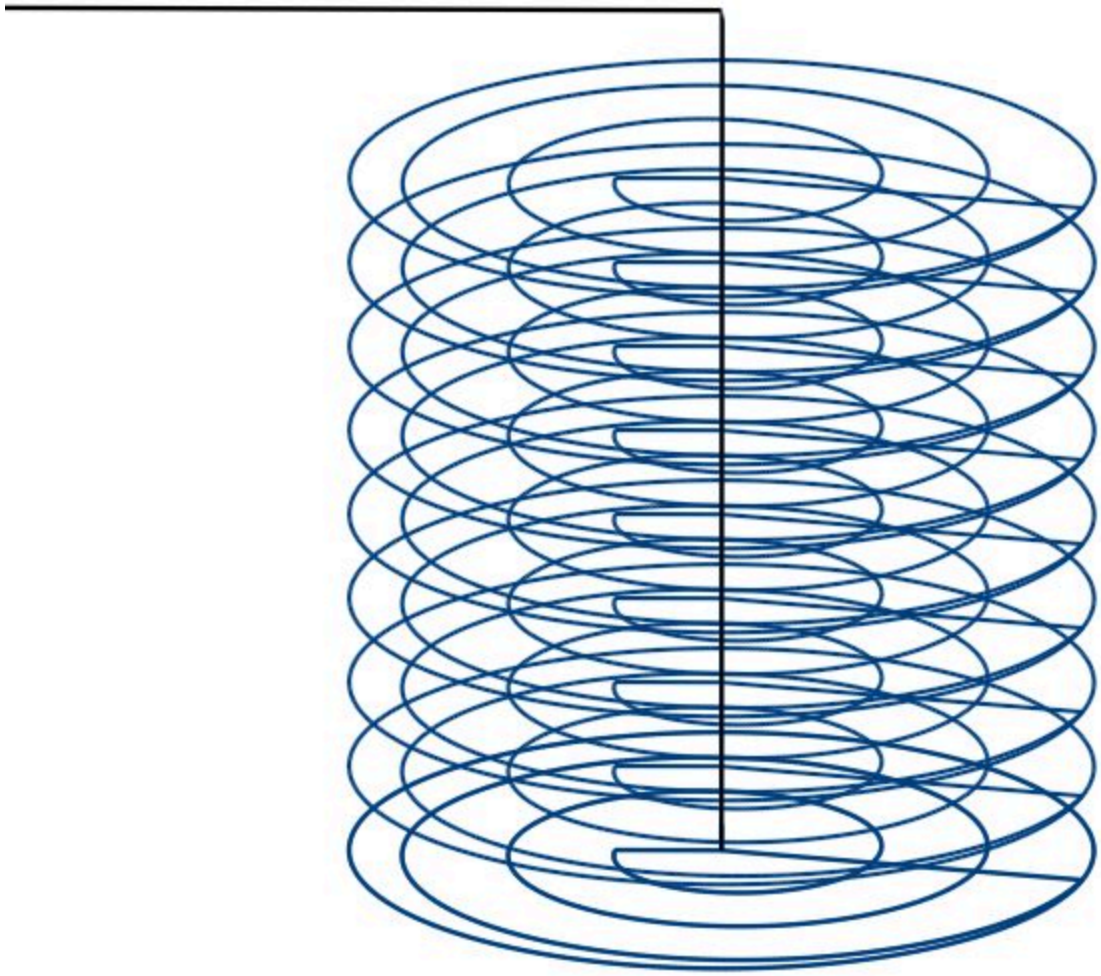
Цикл фрезерования кругового кармана

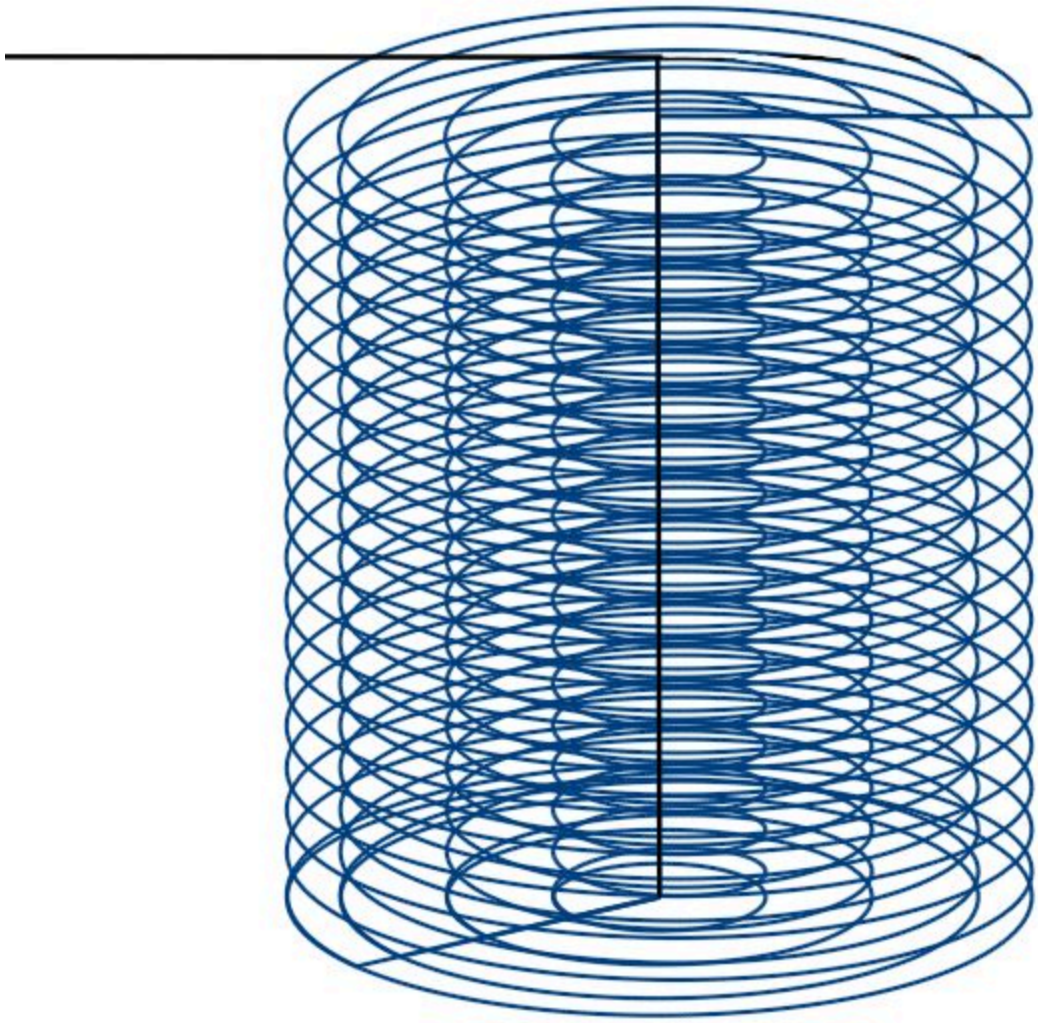
INCT_PocketCycleFormer — интерфейс CAM-системы, реализующий функционал для упрощения формирования движений для цикла фрезерования круглых карманов. Объект, реализующий этот интерфейс, возвращается методом INCT_CLData_PocketCycleFormer.CreatePocketCycleFormer

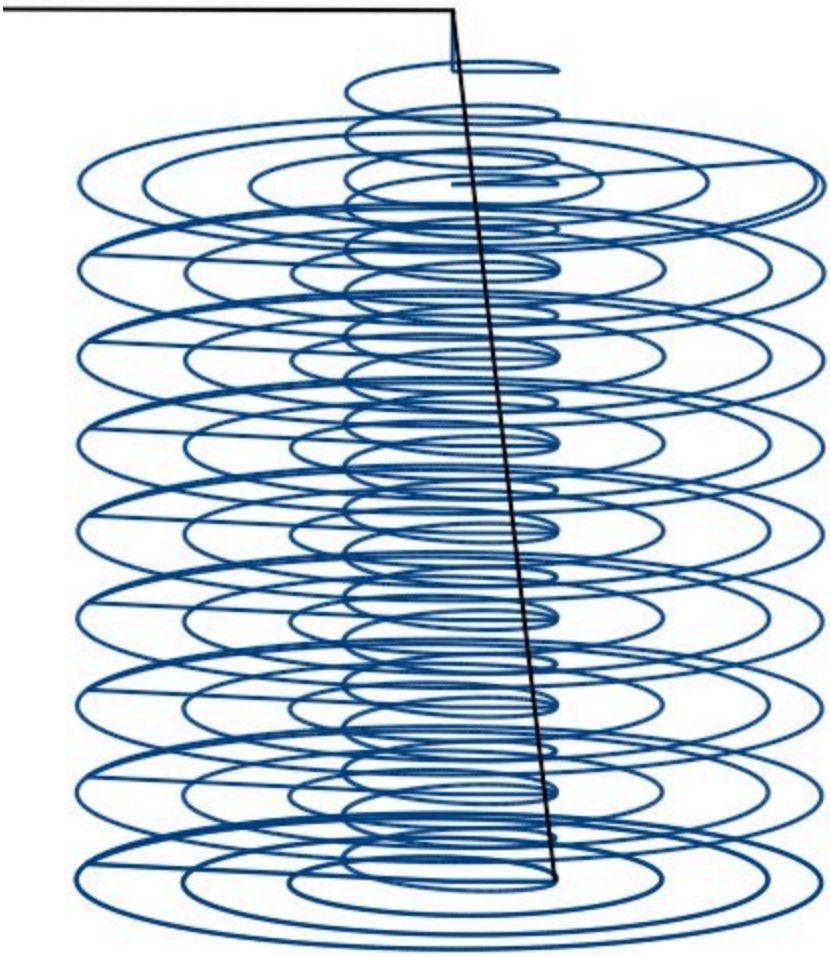












Предназначение:

Выполнение перемещений в различных типах цикла фрезерования круговых карманов, для разных стоек в общем случае повторяют одинаковую траекторию:

1. Подвод к центру отверстия на начальном уровне
2. Подвод к безопасной дистанции
3. Подвод к базовой точке
4. Врезание (по спирали или прямолинейное) до указанной глубины резания или до конца глубины
5. Отход на безопасное расстояние и т. д. циклично до глубины кармана или сразу до глубины кармана при чистовой фрезеровке.
6. Отход к центру отверстия до уровня с которого начиналась обработка в данном случае до начального уровня

Примеры таких циклов, Sinumerik: POCKET4.

Использование формирователя позволяет сформировать траекторию движения инструмента по заданным параметрам. А также добавить в траекторию движения на любом формируемом участке необходимые для конкретного цикла особенности, изменить тип или значение подачи, либо пропустить перемещение инструмента, подготовленное формирователем.

Использование формирователя:

Рекомендуем создавать новый формирователь для каждого нового цикла в кадре управляющей программы. Таким образом, интерфейс старого формирователя должен освобождаться при создании нового или в случае завершения работы интерпретатора.

1. Получить интерфейс формирователя (INCT_CLData_PocketCycleFormer.CreatePocketCycleFormer).
2. Задать индексы осей, по которым будут производиться перемещения (Set_AxisIndexes).
3. Задать текущую плоскость (Set_Plane).
4. Задать центр отверстия (Set_HoleCenter).
5. Задать безопасный уровень (Set_SafeLevel).
6. Задать начало кармана (Set_BaseLevel).
7. Задать уровень дна кармана (Set_BottomLevel).
8. Задать диаметр кармана (Set_Diameter).
9. Задать шаг плоской спирали (Set_FlatSpiralStep).
10. Задать тип шага плоской спирали (Set_FlatSpiralStepType).
11. Задать направление плоской спирали (Set_FlatSpiralDirection).
12. Задать направление обычной спирали (Set_SpiralDirection).
13. Задать шаг обычной спирали (Set_SpiralStep).
14. Задать тип обработки (Set_TypeOfHoleProcessing).
15. Задать глубину резания за проход (Set_Degression).
16. Задать радиус врезания (Set_PlungeRadius).
17. Задать припуск края кармана (Set_EdgeStock).
18. Задать припуск дна кармана (Set_BottomStock).
19. Назначить обработчик событий цикла (Set_Events), если в траекторию цикла нужно вносить изменения.
20. Сформировать команды движения инструмента по траектории цикла (MakeWorkpath).

Описание методов интерфейса:

Метод	Описание метода
Set_AxisIndexes	Метод устанавливает индексы осей X, Y, Z в которых будут производиться перемещения

Метод	Описание метода
	при формировании траектории цикла. Входные параметры: Value (TNCDCAxisIndexes) - X (целое число) - индекс оси X, Y (целое число) - индекс оси Y, Z (целое число) - индекс оси Z. Выходных параметров нет.
Get_AxisIndexes	Функция возвращает индексы осей X, Y, Z в которых будут производиться перемещения при формировании траектории цикла (TNCDCAxisIndexes). Входных параметров нет.
Set_Plane	Метод устанавливает текущую плоскость. Входной параметр: Value (TNCPlaneType) - текущая плоскость. ptXY - плоскость XY, обработка кармана производится вдоль оси Z. ptYZ - плоскость YZ, обработка кармана производится вдоль оси X. ptZX - плоскость ZX, обработка кармана производится вдоль оси Y. Выходных параметров нет.
Get_Plane	Функция возвращает текущую плоскость (TNCPlaneType). Входных параметров нет.
Set_HoleCenter	Метод устанавливает точку центра кармана. Входной параметр: Value (TNC3DPoint) - точка центра кармана. Выходных параметров нет.
Get_HoleCenter	Функция возвращает точку центра кармана(TNC3DPoint). Входных параметров нет.
Set_SafeLevel	Метод устанавливает значение безопасного расстояния Входной параметр: Value (вещественное

Метод	Описание метода
	число) - значение безопасного расстояния Выходных параметров нет.
Get_SafeLevel	Функция возвращает значение безопасного расстояния. Входных параметров нет.
Set_BaseLevel	Метод устанавливает значение начала кармана. Входной параметр: Value (вещественное число) - значение начала кармана Выходных параметров нет.
Get_BaseLevel	Функция возвращает значение начала кармана. Входных параметров нет.
Set_Diameter	Метод устанавливает значение диаметра кармана. Входной параметр: Value (вещественное число) - значение диаметра кармана. Выходных параметров нет.
Get_Diameter	Функция возвращает значение диаметра кармана. Входных параметров нет.
Set_FlatSpiralStep	Метод устанавливает значение шага плоской спирали. Входной параметр: Value (вещественное число) - значение шага плоской спирали Выходных параметров нет.
Get_FlatSpiralStep	Функция возвращает значение шага плоской спирали. Входных параметров нет.
Set_BottomLevel	Метод устанавливает значение уровня дна кармана. Входной параметр: Value (вещественное

Метод	Описание метода
	число) - значение уровня дна кармана. Выходных параметров нет.
Get_BottomLevel	Функция возвращает значение уровня дна кармана. Входных параметров нет.
Set_FlatSpiralStepType	Метод устанавливает значение типа шага плоской спирали. Входной параметр: Value (TNCFlatSpiralStepType) - значение типа шага плоской спирали. fsByMillimeters - Расчет шага спирали производится в миллиметрах. fsByPersentOfToolDiameter - Расчет шага спирали производится в процентах от диаметра инструмента. Выходных параметров нет.
Get_FlatSpiralStepType	Функция возвращает значение типа шага плоской спирали (TNCFlatSpiralStepType) Входных параметров нет.
Set_FlatSpiralDirection	Метод устанавливает значение направления плоской спирали. Входной параметр: Value (TNCSpiralDirection) - значение направления плоской спирали. sdCW - По часовой стрелке. sdCCW - Против часовой стрелки. Выходных параметров нет.
Get_FlatSpiralDirection	Функция возвращает значение направления плоской спирали (TNCSpiralDirection) Входных параметров нет.
Set_SpiralDirection	Метод устанавливает значение направления обычной спирали. Входной параметр: Value (TNCSpiralDirection) - значение направления обычной спирали. sdCW - По часовой стрелке.

Метод	Описание метода
	sdCCW - Против часовой стрелки. Выходных параметров нет.
Get_SpiralDirection	Функция возвращает значение направления обычной спирали (TNCSpiralDirection) Входных параметров нет.
Set_TypeOfHoleProcessing	Метод устанавливает значение типа обработки кармана. Входной параметр: Value (TNCTypeOfProcessing) - значение типа обработки кармана. tpHelicalPlungeRoughing - Черновое, врезание по спирали, обработка плоской спиралью tpLinearPlungeRoughing - Черновое, линейное врезание, обработка плоской спиралью tpOnlyHelicalRoughing - Черновое, полностью спиральная обработка tpOnlyHelicalFinishing - Чистовая, спиральная обработка по краю tpOnlyCircularFinishing - Чистовая, обработка по краю окружностями tpOnlyHelicalFinishingAndBottom - Чистовая, спиральная обработка по краю, обработка дна плоской спиралью. Выходных параметров нет.
Get_TypeOfHoleProcessing	Функция возвращает значение типа обработки кармана (TNCTypeOfProcessing) Входных параметров нет.
Set_SpiralStep	Метод устанавливает значение шага обычной спирали. Входной параметр: Value (вещественное число) - значение шага обычной спирали Выходных параметров нет.
Get_SpiralStep	Функция возвращает значение шага обычной спирали Входных параметров нет.

Метод	Описание метода
Set_Degression	Метод устанавливает значение глубины резания. Входной параметр: Value (вещественное число) - значение глубины резания Выходных параметров нет.
Get_Degression	Функция возвращает значение глубины резания. Входных параметров нет.
Set_DepthOfCut	Метод сохраняет значение глубины резания. Входной параметр: Value (вещественное число) - значение глубины резания. Внимание! В текущей версии значение только хранится и на траекторию не влияет. Глубину резания задаёт Set_Degression. Выходных параметров нет.
Get_DepthOfCut	Функция возвращает значение, заданное методом Set_DepthOfCut (вещественное число). Входных параметров нет.
Set_PlungeRadius	Метод устанавливает радиус врезания: боковой сдвиг от центра кармана перед врезанием по спирали (тип обработки tpHelicalPlungeRoughing). Входной параметр: Value (вещественное число) - радиус врезания. Выходных параметров нет.
Get_PlungeRadius	Функция возвращает радиус врезания. Входных параметров нет.
Set_EdgeStock	Метод устанавливает значение припуска по краю для черновой обработки. Входной параметр: Value (вещественное число) - значение припуска по краю для черновой обработки Выходных параметров нет.

Метод	Описание метода
Get_EdgeStock	Функция возвращает значение припуска по краю для черновой обработки Входных параметров нет.
Set_BottomStock	Метод устанавливает значение припуска по дну кармана для черновой обработки. Входной параметр: Value (вещественное число) - значение припуска по дну кармана для черновой обработки Выходных параметров нет.
Get_BottomStock	Функция возвращает значение припуска по дну кармана для черновой обработки Входных параметров нет.
Set_Events	Метод назначает события для управления поведением цикла. При использовании событий можно добавить в траекторию движения на любом формируемом участке необходимые для конкретного цикла особенности, изменить тип или значение подачи, либо пропустить перемещение инструмента, подготовленное формирователем. Входной параметр: Value (INCT_PocketCycleEvents) - объект, реализующий события. Подробное описание ниже. Выходных параметров нет.
Get_Events	Функция возвращает объект, реализующий события, связанный с формирователем цикла (INCT_PocketCycleEvents). Входных параметров нет.
MakeWorkpath	Метод формирует траекторию цикла по заданным параметрам.

INCT_PocketCycleEvents - интерфейс, реализуемый в интерпретаторе, который позволяет управлять поведением цикла фрезерования кругового кармана. Интерфейс представляет из

себя набор методов-событий, которые выполняются при перемещении инструмента внутри формирователя траектории цикла. Используя методы интерфейса можно добавить в траекторию движения на любом формируемом участке необходимые для конкретного цикла особенности: выстой, ориентированный останов и прочее, изменить тип подачи, либо пропустить перемещение инструмента, подготовленное формирователем.

Если для формирователя траектории цикла назначены события, то любое перемещение инструмента внутри формирователя состоит из 3-х этапов:

1. Выполняется метод INCT_PocketCycleEvents.BeforeMove. В качестве параметров методу передаются заданные циклом: тип перемещения, координаты точки перемещения, тип подачи.
2. Выполняется метод INCT_PocketCycleEvents.OnMove.
   Для OnMove в качестве параметров методу передаются: тип перемещения, координаты точки перемещения, тип подачи, признак выполнения перемещения. Все параметры, кроме типа перемещения, в реализации метода могут быть изменены. В зависимости от заданного в OnMove признака, производится выполнение перемещения инструмента в заданную точку с указанной подачей, или перемещение не производится.
3. Выполняется метод INCT_PocketCycleEvents.AfterMove. В качестве параметров методу передаются: тип перемещения, координаты точки перемещения, тип подачи, возвращённые методом OnMove.

Описание методов интерфейса:

Метод	Описание метода	Описание использования
BeforeMove	Метод будет выполнен перед каждым перемещением инструмента при формировании траектории цикла. Входные параметры: MoveType (TNCPocketCycleMoveType) - тип перемещения, описание см. ниже; Point (TNC3DPoint) - координаты точки перемещения; Feed (TNCFeedType) - тип подачи. pcmMoveToHoleCenter - Перемещение к центру кармана	При необходимости добавления тех или иных команд перед выполнением перемещения, их нужно добавить в реализацию метода. Например, если есть необходимость включить систему охлаждения станка перед созданием кругового кармана, сделать это можно добавив в реализацию метода BeforeMove вызов INCT_CLData.AddCoolant с соответствующими параметрами по условию

Метод	Описание метода	Описание использования
	pcmMoveToSafeLevel - Перемещение к безопасной дистанции pcmMoveToCutOfDepthByLine - Перемещение по линейной траектории до следующего уровня реза pcmMoveToCutOfDepthByHelic - Перемещение по спиральной траектории до следующего уровня реза pcmMoveToAside - Перемещение от центра в сторону края кармана pcmMoveToHoleEdge - Перемещение на край кармана pcmMoveOnHoleEdge - Перемещение непосредственно на траектории края кармана pcmMoveToTopLevel - Перемещение к начальной точке pcmMoveFromHoleEdgeToCenter - Перемещение от края в центр кармана. Выходных параметров нет.	MoveType = pcmMoveToSafeLevel
OnMove	Метод будет выполнен непосредственно перед каждым перемещением инструмента при формировании траектории цикла. Входные параметры: MoveType (TNCPocketCycleMoveType) - тип перемещения, описание	При необходимости переопределения значений точки перемещения или подачи, либо для отказа от перемещения (пропуска). Например, если нужно изменить тип подачи на ускоренную, при возврате на начальный уровень, после фрезерования, то в

Метод	Описание метода	Описание использования
	см. выше; Входные и выходные параметры: Point (TNC3DPoint) - координаты точки перемещения; Feed (TNCFeedType) - тип подачи; Handled (логическое) - признак выполнения перемещения внутри реализации (Истина - перемещение выполнено, Ложь - перемещение не выполнено). Если после вызова OnMove параметр Handled принимает значение Истина, это значит, что перемещение было выполнено внутри реализации метода OnMove и формирователь цикла не добавляет команду перемещения в траекторию движения инструмента. Если после вызова OnMove параметр Handled принимает значение Ложь, формирователь считает, что перемещение не было выполнено внутри реализации метода OnMove и добавляет в траекторию движения инструмента команду перемещения в точку Point с типом подачи Feed.	реализации метода OnMove необходимо изменить значение возвращаемого методом параметра Feed на ffRapid по условию MoveType = pcmMoveToTopLevel (типы подачи см. в описании OutStandardFeed). Если нужно выполнить перемещение инструмента в точку, отличную от рассчитанной циклом, необходимо изменить значение возвращаемого методом OnMove параметра Point. Если перемещение в рассчитанную циклом точку выполнять не нужно, тогда внутри реализации метода OnMove нужно изменить значение параметра Handled на Истина.
AfterMove	Метод будет выполнен после каждого перемещения инструмента при	При необходимости добавления тех или иных команд после выполнения

Метод	Описание метода	Описание использования
	формировании траектории цикла, даже если перемещение было отменено в OnMove Входные параметры: MoveType (TNCPocketCycleMoveType) - тип перемещения, описание см. выше; Point (TNC3DPoint) - координаты точки перемещения; Feed (TNCFeedType) - тип подачи. Выходных параметров нет.	перемещения, их нужно добавить в реализацию метода.

Журнал взаимодействия с интерпретатором

Процесс интерпретации текста управляющей программы, включая взаимодействие с интерпретатором, записывается в журнал, если включена отдельная опция журнала. Управление журналом производится в окне Системные настройки: страница <Дополнительные>, группа <Логирование>, опции <Вести журнал> и <Вести журнал взаимодействия с интерпретатором>. Вторая опция доступна, только когда включена первая. В NC Тюнере журнал включается так же.

Если обе опции включены, то при расчёте в режиме "Моделирования по тексту управляющей программы", а также "операций на основе текста УП" в журнал записывается процесс интерпретации текста УП: сообщения ядра о прочитанных словах и строки, которые пишет интерпретатор.

Журнал ведётся в одном файле на каждый запущенный процесс программы: все расчёты, выполненные за время работы программы, записываются в этот файл. Расположение файла:

- СПРУТКАМ - `%APPDATA%\Sprut Technology\SprutCAM\Version <номер версии>\Log\sc-<номер процесса>.log`;
- NC Тюнер - `%APPDATA%\Sprut Technology\NCTuner\Version <номер версии>\Log\ltuner-<номер процесса>.log`.

Когда файл журнала превышает 20 МБ, его прежнее содержимое переносится во вложенную папку `RotatedLogs`, а запись продолжается в файл с тем же именем.

Журнал предназначен для разработки собственных интерпретаторов.

Разработка интерпретаторов с помощью ИИ

Этот раздел описывает, как разрабатывать и сопровождать интерпретатор с помощью ИИ-ассистента: как подготовить инструменты, о чём просить ассистента, что ему нужно от вас и как проверять результат. Раздел относится к интерпретаторам на C#, созданным из шаблонов CAM системы, которые описаны в разделе Разработка собственного интерпретатора на (.NET) C#, и к нативным интерпретаторам на Delphi или C++ (см. Программная библиотека интерпретатора).

По умолчанию ассистент разрабатывает интерпретатор на C# (.NET). Нативную библиотеку на Delphi или C++ он создаёт, только если вы прямо об этом попросили или правите уже существующий нативный интерпретатор.

Как выглядит работа

Вы пишете в чате своими словами:

> В программах для нашего станка поворотная ось записана как B90, а на станке она называется B1. Программа с B90 не моделируется. Исправь.

И дальше наблюдаете. Ассистент читает УП и станок проекта, находит место, где интерпретатор читает слова осей, решает, что это за отличие — настройка или изменение кода, вносит изменение, собирает интерпретатор, прогоняет через него УП в CAM системе и показывает, что изменилось в моделировании и в журнале.

Вы читаете результат и отвечаете так же — «верно, но этот же станок пишет ещё и B=DC(30)», — и цикл повторяется.

Писать код самому не нужно, хотя вы в любой момент можете посмотреть каждое изменение и поправить его руками. Передать нельзя только решение: соответствует ли моделирование тому, что делает ваш станок, определяете вы. Всё, что в этом разделе сказано о проверке результата, существует именно для этого.

Что нужно ассистенту от вас

- УП, которые должен читать интерпретатор, и проект CAM системы со станком, для которого они написаны.
- Руководство по программированию стойки ЧПУ для нужных слов и циклов.
- Установленные CAM система или NC Тюнер, в которых должен работать интерпретатор.
- Ожидаемый результат: что станок делает с кадром, который моделируется неверно.

Чем больше настоящих программ вы дадите, тем лучше: написание слов ассистент берёт из них, а не из памяти.

Во что это обходится

Документация и навыки публикуются бесплатно, а модель, которая вам отвечает, в них не входит. ИИ-ассистент — услуга того, кто предоставляет модель; она оплачивается по объёму обработанного текста или подпиской. Понадобится учётная запись у поставщика ИИ-услуг или у разработчика ИИ-клиента; работа идёт на его условиях и по его расценкам.

Если данные должны оставаться внутри вашей сети, модель можно развернуть на собственном оборудовании и направить на неё клиент, который это умеет. Учтите, что небольшие локальные модели заметно хуже справляются с многошаговой работой по навыкам: прежде чем полагаться на такую модель, проверьте её на настоящей задаче. Остальное в этом разделе не меняется — CAM система, интерпретатор и УП и так находятся на вашем компьютере.

Порядок чтения

Страница	За что отвечает
Как устроена разработка с помощью ИИ	Из каких частей состоит работа, что делает каждая, что ассистент умеет и чего не может
Настройка инструментов	Короткий путь: установить ИИ-клиент, скачать навыки, проверить двумя запросами
Организация рабочего пространства	Рабочая папка как память ассистента: УП, проект станка, руководство, заметки, история изменений
Рабочий цикл .NET	Путь по умолчанию: интерпретатор на C# из шаблона — создание, сборка, отладка, изменения
Рабочий цикл нативного интерпретатора	Только по прямой просьбе или для уже нативного интерпретатора: Delphi, C++
Проверка и подтверждение результата	Прогон всех УП до и после изменения, уровни проверки, подготовка интерпретатора к передаче
Устранение проблем	Типичные симптомы, их причины и что делать
Расширенная настройка	Полная процедура настройки: навыки в разных клиентах, средства сборки и

Страница	За что отвечает
	отладки, журнал, файл инструкций, несколько версий продукта, отдельное рабочее место, автоматические проверки

Если вы только начинаете, прочитайте страницы по порядку до рабочего цикла .NET; страницу о нативном интерпретаторе — только если он вам нужен. Страницу Расширенная настройка можно не читать самому: её читает ассистент, когда настраивает инструменты.

Навыки для ассистента

Навык (англ. skill) — это инструкция обычными словами: что сделать, в каком порядке, что проверить и когда работа считается законченной. Кода в ней нет — это текстовый файл, который ассистент читает и которому следует, примерно как объяснение новому сотруднику. Написать или поправить такой навык может любой, кто способен объяснить порядок работы словами.

Навыки лежат в папке `skills` этого раздела: каждый навык — отдельная папка с файлом `SKILL.md`. Вместе со всей документацией раздела они публикуются в открытом репозитории и скачиваются одним архивом: скачать архив документации. Навыки написаны на английском языке; переписываться с ассистентом можно на любом. Вместе они ведут всю работу над интерпретатором: ассистент начинает с общего навыка, а по ходу работы обращается к остальным.

Навык (папка)	Для чего	Страница раздела
`interpreter-development`	Общий навык: что выяснить сначала, в каком порядке работать и когда работа закончена. По умолчанию интерпретатор разрабатывается на C# (.NET)	Как устроена разработка с помощью ИИ
`inspect-nc-programs`	Сбор УП и данных станка (оси, каналы, M-функции) и перечень всех слов, которые встречаются в программах	Организация рабочего пространства
`develop-dotnet-interpreter`	Создание, настройка, сборка и отладка интерпретатора на C# из шаблона — путь по умолчанию	Рабочий цикл .NET

Навык (папка)	Для чего	Страница раздела
`develop-native-interpreter`	Нативная библиотека интерпретатора на Delphi или C++ — только когда нужен именно нативный интерпретатор	Рабочий цикл нативного интерпретатора
`nc-interpreter-foundations`	Как CAM система читает кадр и когда вызывается код интерпретатора	Как устроена разработка с помощью ИИ
`verify-interpretation`	Прогон всех УП до и после изменения, сравнение результатов и подготовка интерпретатора к передаче	Проверка и подтверждение результата

Каждый навык можно прочитать и на этом сайте — по ссылке в таблице. Если архив недоступен, исходный файл `SKILL.md` можно скачать со страницы навыка; положите его в папку с именем навыка, например `skills/interpreter-development/SKILL.md`. Копируйте навыки все сразу: они ссылаются друг на друга. Как подключить их к вашему ИИ-клиенту — в настройке инструментов, подробно — в расширенной настройке.

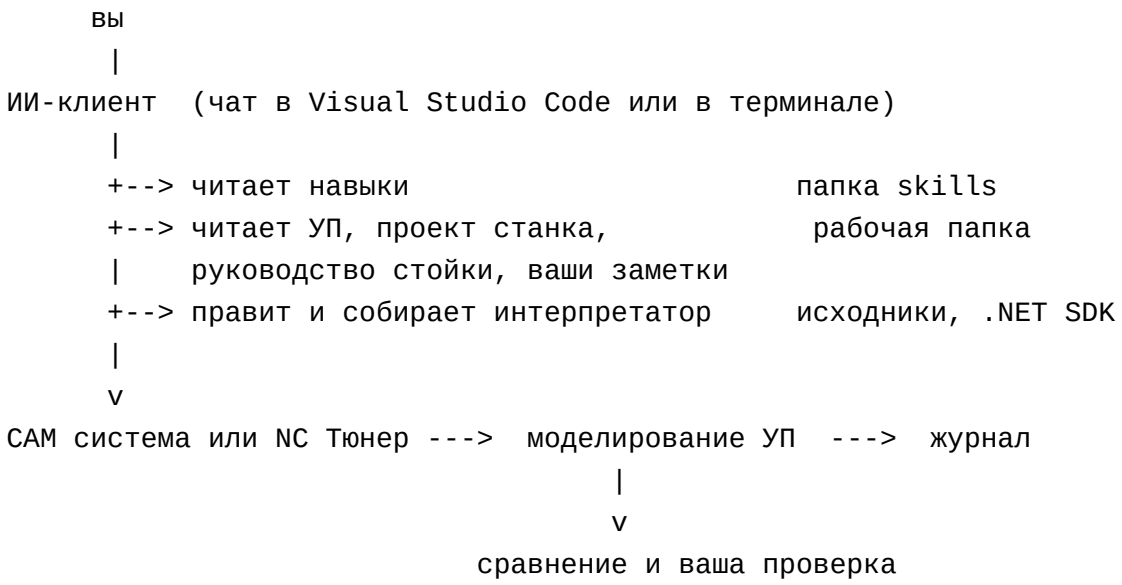
Рядом с ними можно держать свои навыки. Это обычный Markdown: так же оформляются практики, которые накапливает ваша команда, — поддерживаемые стойки, особенности программ ваших заказчиков. См. организацию рабочего пространства.

Что остаётся за вами

Считайте любой ответ черновиком. Ассистент видит только то, что вы ему дали: он не знает недокументированного поведения стойки, а правдоподобно выглядящее моделирование может быть неверным. Прежде чем передать интерпретатор пользователю, прогоните через него все УП, которые у вас есть, прочитайте предупреждения в журнале взаимодействия с интерпретатором и проверьте моделирование изменённых кадров. Не передавайте в чат пароли, архивы заказчиков и данные, которые не должны покидать вашу компанию.

Как устроена разработка с помощью ИИ

Вы разговариваете с ИИ-клиентом — чатом в редакторе кода или в терминале. Клиент читает навыки и файлы вашей рабочей папки, правит исходники интерпретатора, собирает его и запускает CAM систему или NC Тюнер, в которых интерпретатор моделирует УП. Особых серверов не нужно, а из расширений Visual Studio Code — только сам чат и расширение C# для сборки и отладки: ассистент работает с теми же файлами и программами, что и вы.



Из чего всё состоит

Часть	Что это	Что она вам даёт
ИИ-клиент	Чат-расширение Visual Studio Code или программа в терминале	Разговор с ассистентом, которому разрешено читать файлы, править их и запускать команды в рабочей папке
Навыки (skills)	Текстовые файлы `SKILL.md`	Написанный словами порядок работы, поэтому цикл каждый раз одинаков: сначала факты, потом изменение, потом проверка. Ведёт работу общий навык `interpreter-development`, остальные он подключает по ходу

Часть	Что это	Что она вам даёт
Рабочая папка	Обычная папка на вашем компьютере	УП, проект станка, руководство стойки и ваши заметки — всё, что ассистент знает о вашей задаче
Исходники интерпретатора	Проект на C# из шаблона или нативный проект	То, что ассистент меняет
.NET SDK и Visual Studio Code с расширением C#	Средства сборки и отладки	Сборка интерпретатора и запуск под отладчиком клавишей `F5`
CAM система или NC Тюнер	Установленный продукт	Загружает интерпретатор и моделирует УП в операции на основе текста УП
Журнал взаимодействия с интерпретатором	Файл журнала продукта	Что ядро и интерпретатор сделали с каждым кадром: прочитанные слова, предупреждения, ошибки

Как CAM система читает кадр

Работой управляет ядро CAM системы, а не интерпретатор: ядро читает текст УП, по таблице регистров интерпретатора разбирает кадр на слова, ставит их в очередь и вызывает код интерпретатора, чтобы он выполнил каждое слово. Какие функции и в каком порядке вызываются, описано в разделе Процесс интерпретации текста управляющей программы; навык `nc-interpreter-foundations` пересказывает это для ассистента вместе с ловушками, на которых ошибаются чаще всего: первое движение операции, координаты в повёрнутой системе, шпиндели, циклы.

Смысл каждого слова задаёт интерпретатор, а одно и то же слово на разных стойках значит разное. Поэтому ассистент всегда начинает с фактических программ и руководства стойки, а не с названия слова. Интерпретатор моделирует то, что сделала бы стойка, а не то, что задумывала CAM операция.

C# (.NET) по умолчанию

Новый интерпретатор ассистент создаёт на C# из шаблона CAM системы. Такой интерпретатор собирается одной командой, отлаживается в Visual Studio Code и легко читается людьми, которые знают станок лучше, чем программирование. Нативный интерпретатор на Delphi или

C++ — только когда вы прямо об этом попросили или когда правите уже нативный. Подробно — Рабочий цикл .NET и Рабочий цикл нативного интерпретатора.

Что ассистент делает хорошо, а что не может

Он хорошо собирает перечень всех слов, которые встречаются в ваших программах, находит место в интерпретаторе, где читается слово, объясняет незнакомый код, готовит небольшое изменение в стиле окружающего кода, прогоняет все УП до и после изменения и сравнивает результаты кадр за кадром.

Он не может знать поведение стойки, которого нет ни в руководстве, ни в программах, увидеть станок, судить о безопасности перемещений и отвечать за интерпретатор, переданный заказчику. Почему успешной сборки и моделирования без ошибок недостаточно и как проверять результат — в разделе Проверка и подтверждение результата.

Настройка инструментов

Настройкой занимается сам ассистент. Ваша часть невелика: установить ИИ-клиент, скачать архив с навыками и передать ассистенту инструкцию. Остальное — разложить навыки, проверить средства сборки и отладки, найти установленную CAM систему — ассистент делает сам и докладывает о результате. Без вас не обойдутся только разрешения и действия в окнах программ: разрешение установить недостающие программы; перезапуск ИИ-клиента, если он нужен; в CAM системе или NC Тюнере — создание интерпретатора в окне Создать интерпретатор УП, кнопка Задать путь к файлу СПРУТКАМ в этом окне (в NC Тюнере на кнопке — его название), после неё — выход из сеанса Windows и вход заново, и включение журнала в окне Системные настройки. О каждом из этих действий ассистент скажет сам, когда до него дойдёт очередь.

1. Установите ИИ-клиент

Удобнее всего работать в Visual Studio Code: в нём же интерпретатор собирается и отлаживается (см. Отладка интерпретатора в Visual Studio Code). Чат — такое же расширение, как и остальные: откройте панель Extensions (`Ctrl+Shift+X`), найдите нужное по названию и установите. Подойдёт любой клиент, которому можно разрешить читать и править файлы рабочей папки и запускать в ней команды, — например, Kilo Code или Claude Code; клиент, работающий в терминале, устанавливается своим способом.

Понадобится учётная запись у поставщика ИИ-услуг — см. во что это обходится.

Для интерпретатора на C# понадобятся также Visual Studio Code с расширением C# от Microsoft и .NET SDK той версии .NET, на которой работает ваша версия продукта (см. Версии продукта). Устанавливать их заранее не обязательно: на шаге 3 ассистент проверит, чего не хватает, и предложит установить недостающее — с вашего разрешения.

2. Скачайте навыки

Скачайте архив документации и распакуйте его. Всё содержимое архива лежит в корневой папке `public-docs`, навыки — в ней в папке `src/ai-assisted-development/skills/`: шесть папок, в каждой файл `SKILL.md`.

3. Передайте это ассистенту

Откройте в Visual Studio Code рабочую папку (см. организацию рабочего пространства), откройте чат, разрешите ему выполнять команды в рабочей папке и поставьте задачу. Навыки копируются в папку пользователя (`%USERPROFILE%`), то есть вне рабочей папки, поэтому клиент спросит разрешение и на это. Поправьте пути (`<архив>` — полный путь к папке `public-docs` из распакованного архива) и вставьте:

```
Настрой инструменты для разработки интерпретаторов по инструкции
<архив>\src\ai-assisted-development\advanced-setup.md
(та же страница на сайте: https://docs.sprut.ru/technical/Interpreters/3/ru/src/ai-
assisted-development/advanced-setup.html)

Распакованный архив документации: <архив>
CAM система или NC Тюнер установлены в: <папка установки>

Скопируй все навыки из архива в папку навыков этого клиента
для пользователя, а не для рабочей папки,
проверь .NET SDK и Visual Studio Code с расширением C#;
чего не хватает — назови и предложи установить.
Затем скажи, что именно изменил и какие навыки теперь видишь.
```

Новые навыки клиенты подхватывают по-разному: Claude Code — сразу (если папки навыков до этого не было — после команды `/reload-skills`), Kilo Code — в новой сессии или после команды `/reload`, другие клиенты — по своей документации. Если клиенту нужен перезапуск, ассистент скажет об этом: сделать это за вас он не сможет.

4. Проверьте, что всё получилось

Две просьбы в чат — этого достаточно. Обе безопасны: ассистент только читает и ничего не изменяет.

Проверка навыков. Напишите:

> Какие навыки для разработки интерпретаторов ты сейчас видишь и с какого начнёшь работу?

Хороший ответ перечисляет шесть навыков по именам папок и называет `interpreter-development` первым. Если ассистент рассуждает о «типичных» навыках или просит прислать файлы, навыки не подключились — вернитесь к шагу 3 или к расширенной настройке.

Проверка чтения программ. Положите в папку `programs` рабочей папки одну-две настоящие УП и напишите:

> Прочитай УП в папке `programs` и перечисли, какие слова и в каком написании в них встречаются. Ничего не меняй.

Хороший ответ — перечень слов из ваших файлов: адреса, G- и M-функции, написания осей вроде `B90` и `B1=90`. Если в ответе слова, которых в программах нет, ассистент рассуждает по памяти: укажите ему файлы явно.

5. Обновления

Навыки обновляются вместе с документацией. Когда выйдет новая версия, скачайте архив заново и попросите ассистента заменить папки навыков целиком.

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, какая стойка доставила хлопот в прошлом месяце и почему интерпретатор читает конкретное слово 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 следующей сессии.

Что в ней держать

Одна папка, открытая в Visual Studio Code:

```
work\
├── AGENTS.md          что ассистент должен знать каждый раз
├── interpreters\      исходники интерпретаторов, по папке на интерпретатор
├── programs\          настоящие УП станка и ожидаемые результаты
├── projects\          проекты CAM системы со станком
├── manuals\           руководство по программированию стойки
└── notes\             ваши практики и разобранные случаи
```

Имена не важны, важно разделение: исходники отдельно от тестовых данных, ваши знания отдельно от официальной документации.

Программы и станок

Положите в `programs\` несколько настоящих УП этого станка, а не одну: сформированные его постпроцессором, написанные вручную, из других CAM систем. Написание слов ассистент берёт из них: одна и та же ось может встретиться как `B90`, `B=90` и `B1=90`, и интерпретатор должен понимать каждое написание, которое встречается в ваших программах. Если есть постпроцессор, который формирует эти программы, положите и его: по нему видно, какие варианты слов появятся.

Рядом держите проект CAM системы с этим станком (`*.stcp`) и, если есть, ожидаемый результат: что станок делает с программой. Руководство стойки в `manuals\` — главный источник смысла слов; без него ассистент будет угадывать.

Попросите ассистента составить перечень слов по всем программам — это делает навык `inspect-nc-programs`. Перечень показывает, какие слова интерпретатор уже понимает, а какие нет, и служит списком того, что нужно проверить после каждого изменения.

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:

- станок, стойка ЧПУ и её версия, какой интерпретатор её обслуживает;
- версии CAM системы и NC Тюнера, в которых интерпретатор должен работать;
- где что лежит: программы, проекты, руководство, заметки;
- правила работы — сначала прочитать программы и руководство, изменение делать минимальным, после каждого изменения прогонять все программы и сравнивать с предыдущим результатом;
- ограничения, которые вы уже однажды сформулировали, — записанные так, как вы их сказали. Иначе через месяц ассистент 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.

- Одна тема — один файл, а не один растущий: маленький файл легко найти и дёшево прочитать.
- Описывайте текущее состояние. Оговорки вроде «раньше было иначе» со временем портятся; если папка под контролем версий, история и так есть.

Держите в `notes\` и письменный список просьб и решений по текущей задаче. Длинный разговор ассистент сжимает, и подробности из него выпадают; список — нет. Перед докладом «готово» попросите ассистента пройти по нему пункт за пунктом и показать подтверждение по каждому.

Когда порядок работы начинает повторяться — одна и та же проверка, одна и та же последовательность для семейства стоек, — запишите его как свой навык: папку с файлом `SKILL.md`, как у поставляемых навыков. Клиент находит навыки только в своей папке навыков, поэтому кладите свой навык туда же, куда установлены остальные (см. расширенную настройку); файл в `notes\` ассистент прочитает, только если вы на него укажете. Навык, в котором названы заказчики или особенности именно ваших станков, не передавайте вместе с интерпретатором и не публикуйте.

Два способа хранить историю

Папки и копии — историю ведёте вы сами

Перед существенным изменением вы копируете папку интерпретатора, рядом с программами держите результаты прежних прогонов, а историей служат папки с датами. Это тоже рабочий способ. Плата за него — история держится на вашей памяти: забыли сделать копию — предыдущего состояния больше нет.

Git — историю ведёт инструмент

Та же папка, но за прошлое отвечает инструмент, сделанный именно для этого. Он же снимает главное опасение при работе с ассистентом: каждое состояние сохранено, поэтому неудачная правка стоит минуты на откат, а не работающего интерпретатора.

Учить git для этого не нужно. Просите словами, а делает ассистент:

- «зафиксируй, ось B1 читается» — он фиксирует изменение с сообщением о том, зачем;
- «сравни с тем, что было в пятницу» — он показывает, что изменилось;
- «найди, когда перестал моделироваться цикл G83» — он ищет по истории;
- «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 git-scm.com. Попросите заодно исключить из истории результаты сборки (`bin`, `obj`) и журналы.

Когда открытой папкой должна быть папка интерпретатора

Для интерпретатора на C# Visual Studio Code должен быть открыт на папке самого интерпретатора — так работают сборка и отладка клавишей `F5`. Памятка и заметки при этом оказываются выше открытой папки. Достаточно двух приёмов:

- Держите всё в одном репозитории. Пусть репозиторием будет папка `work\` целиком, а папки интерпретаторов лежат внутри неё. Git работает из любой вложенной папки, и история остаётся одна.
- Сделайте памятку видимой из папки интерпретатора. Claude Code сам читает файлы инструкций в папках выше открытой, ему ничего не нужно. Kilo Code читает только открытую папку: укажите памятку в его настройках или положите в папку интерпретатора короткий `AGENTS.md` из двух строк: правила — в `..\..\AGENTS.md`, заметки — в `..\..\notes\`. Подробно — в расширенной настройке.

Навыки ставьте для пользователя, а не для рабочей папки: тогда они доступны при любой открытой папке, и их достаточно положить один раз.

Чего в ней быть не должно

Паролей, токенов, файлов лицензий, архивов заказчиков, не относящихся к интерпретатору, и посторонних репозиториев. Всё, что лежит в рабочей папке, ассистент вправе прочитать, а если вы ведёте историю — это остаётся в ней. Если программа заказчика нужна только как пример, дайте сокращённую: несколько кадров с проверяемым словом и заголовок программы.

Рабочий цикл .NET

Интерпретатор на C# — это проект .NET, который ссылается на пакет SDK интерпретаторов и собирается в библиотеку `*.dll`; рядом с ней лежит файл настроек `*.snci`. CAM система или NC Тюнер загружает библиотеку и вызывает её при моделировании УП. Это путь по умолчанию: ассистент выбирает его для каждого нового интерпретатора, если вы не попросили иного.

Как создать интерпретатор из шаблона, описано в разделе Разработка собственного интерпретатора на (.NET) C#, как запустить его под отладчиком — в отладке интерпретатора в Visual Studio Code. Подробности для ассистента — в навыке `develop-dotnet-interpreter`.

С чего начинать

Чаще всего не с нуля. Если есть интерпретатор для той же или похожей стойки, поведение которого проверено на настоящих программах, — скопируйте его и доработайте. Копии нужны новые идентификаторы: GUID, который возвращает `GetIdentifier` (он же `LibraryID` в файле `*.snci`), и GUID в поле `ID`. Два интерпретатора с одинаковыми идентификаторами CAM система не различает.

Если нужен новый, создайте его пунктом Создать интерпретатор УП в списке Расширения CAM системы или пунктом Управление интерпретаторами в NC Тюнере — оба открывают одно и то же окно Создать интерпретатор УП (создатель интерпретатора). Шаблон, из которого создаётся интерпретатор, рассчитан на одно семейство стоек; ассистент проверяет и исправляет то, чего создатель интерпретатора не делает сам: заполняет `ID` в файле `*.snci`, ставит выпущенную версию пакета SDK, убирает обработчики слов, которые на вашей стойке значат другое. У дочернего интерпретатора он исправляет и ссылку на родителя: `ParentID` должен совпадать с `ID` родителя, а `ParentFileName` — называть файл `*.snci` родителя.

Интерпретатор может наследовать другой — см. Наследование интерпретатора. Прежде чем брать поставляемый интерпретатор родителем, проверьте его назначение в перечне интерпретаторов: у некоторых указано «Только для импорта траектории» — они не связывают кадры УП с движениями.

В любом случае начинайте с известного состояния:

1. Поставьте интерпретатор под контроль версий, прежде чем что-либо менять.
2. Соберите его без изменений. Если он не собирается как есть, дальнейшая диагностика бессмысленна.
3. Прогоните через него все ваши УП и сохраните результат как исходный для сравнения.

Версии продукта

У каждой версии продукта своя версия .NET: СПРУТКАМ 20 и NC Тюнер 3 работают на .NET 10, СПРУТКАМ 19 и NC Тюнер 2 — на .NET 8.

Интерпретатор ссылается на пакет `SprutTechnology.NCInterpreterSDK.Net`. Его версию ассистент определяет так:

1. Читает, с какой версией пакета собран установленный продукт: файл `Bin64\STNCInterpreterSDKProject.stbuildresult.json`, раздел `build_results.versions`. В установке NC Тюнера этого файла может не быть — тогда версию берут из CAM системы той же версии (NC Тюнеру 3 соответствует СПРУТКАМ 20, NC Тюнеру 2 — СПРУТКАМ 19). Если файл в NC Тюнере есть, номер в нём задаёт сборка NC Тюнера, которая пакеты SDK не публикует, поэтому пакета с точно таким номером в источнике может не быть.
2. Ставит самую новую выпущенную версию пакета (без `-dev`, `-rc` и подобных окончаний) с тем же основным номером и не новее версии из шага 1. Для NC Тюнера это номер из его файла, а если файла нет — из файла CAM системы той же версии. Номер из файла NC Тюнера служит только верхней границей.

Выпущенные пакеты лежат в публичном источнике пакетов производителя `https://nexus.sprut.ru/repository/master/index.json`. Подключить его можно командой:

```
dotnet nuget add source "https://nexus.sprut.ru/repository/master/index.json" -n
"sprut-master-feed"
```

Поиск по источнику может показывать устаревшую «последнюю» версию: смотрите список версий самого пакета. Версию, с которой собиралась ночная или тестовая сборка продукта, заказчик скачать не может, поэтому в проект, который уходит с вашего компьютера, она не попадает.

Если интерпретатор должен работать в нескольких версиях продукта, один проект собирается для каждой из них; проверять его нужно в каждой.

Цикл

1. Сформулируйте задачу как видимую разницу в моделировании: какая программа, какой кадр, что должно произойти.
2. Попросите ассистента найти место, где интерпретатор читает это слово, и объяснить, что он делает сейчас.
3. Решите вместе с ним, что это за изменение:
   - вариация — другое имя оси, M-функция станка, слово без влияния на траекторию. Её место — файл `InterpreterSettings.ini` рядом с библиотекой, а не код: тогда следующий такой случай инженер поддержки решит без сборки. Первый раз ключ настройки добавляется в код, дальше это настройка;

    - новое поведение — цикл, фрейм, режим подачи. Это изменение кода, в файле той роли, к которой оно относится;
    - ограничение ядра или SDK — нужного вызова нет. Ассистент описывает обход и его пределы и записывает, что пришлось бы изменить в продукте.
4. Внесите одно небольшое изменение и соберите проект. Предупреждения сборки — сведения о вашем изменении, а не шум.
5. Прогоните все УП и сравните с предыдущим результатом — см. Проверка и подтверждение результата.

Отладка

Клавиша `F5` в Visual Studio Code собирает интерпретатор и запускает конфигурацию, выбранную на панели Run and Debug: Launch CAM — CAM систему, Launch Tuner — NC Тюнер; точка останова срабатывает, когда моделируется УП. Перед отладкой включите журнал, как описано в расширенной настройке, — иначе в журнале не будет ни сообщений ядра о прочитанных словах, ни строк, которые пишет интерпретатор.

Чтобы смоделировать программу, создайте операцию на основе текста УП, вставьте текст программы, выберите интерпретатор в поле Файл интерпретатора на вкладке Стратегия параметров операции и рассчитайте операцию.

Сохраняйте сборку и отладку такими, какими их сделал шаблон: коллега должен суметь открыть папку, нажать `F5` и остановиться на точке останова. Ассистент не заменяет их своими скриптами и переменными среды.

Что проверять в коде

- Изменения содержат то, о чём просили, и ничего больше: ни переформатирования, ни посторонних переписываний.
- Концы строк и кодировка файлов шаблона сохранены — иначе история показывает изменённой каждую строку.
- Числа в программе читаются независимо от языка Windows: точка — всегда десятичный разделитель.
- Состояние интерпретатора хранится в его объекте, а не в статических полях: CAM система создаёт новый экземпляр на каждый расчёт операции, отдельные — для родителя и для контура цикла, и статическое поле у них не общее; NC Тюнер, наоборот, держит один экземпляр на канал — от первой операции канала до последней, — и модальное состояние и статическое поле переходят из операции в операцию: например, `G0` из конца одной операции действует в начале следующей.
- Слово, которое интерпретатор не моделирует, не исчезает молча: оно либо значится как читаемое без влияния на траекторию, либо даёт одно предупреждение на программу, понятное оператору станка.

Правила, которые не зависят от ядра, — разбор чисел и выражений, параметры циклов, правила для осей и шпинделей, чтение файла настроек — ассистент может покрыть модульными тестами. Тесты дополняют прогон настоящих программ, но не заменяют его.

Рабочий цикл нативного интерпретатора

Нативный интерпретатор — это библиотека `*.dll`, собранная на Delphi, C++ или другом языке, который компилируется в машинный код. Ассистент берётся за нативный интерпретатор, только если вы прямо об этом попросили или правите уже существующий нативный интерпретатор; новый интерпретатор по умолчанию создаётся на C# — см. Рабочий цикл .NET.

Справка по библиотеке — в разделах Программная библиотека интерпретатора, Файл настроек интерпретатора и Интерфейс интерпретатора. Подробности для ассистента — в навыке `develop-native-interpreter`.

Что нужно ассистенту

- Исходники интерпретатора и проект, которым он собирается, — или явная просьба создать новый.
- Компилятор: RAD Studio для Delphi, компилятор C++ или другой.
- Объявления интерфейсов SDK. Они приходят пакетами NuGet из того же источника, что и пакет SDK для C# (см. Версии продукта), с той же версией: для Delphi — `SprutTechnology.NCInterpreterSDK.bpl.x64` (модули `IDL.<Имя>.pas` и пакеты `*.bpl`, `*.dcp`), для C++ и других языков — `SprutTechnology.NCInterpreterSDK.tlb` (библиотеки типов `*.tlb`, заголовки `*.h` и описания `*.idl`). Оба пакета подтягивают по пакету на интерфейс, со своими номерами версий: `STNCInterpreter`, `STNCAnalyzer`, `STNCFormers`, `STNCCustomInterpreter` и `STNCConnect` (интерфейс защиты, его использует защищённый интерпретатор). Ассистент берёт объявления из них, а не придумывает. Если пакета нужной версии в источнике нет, обратитесь в службу поддержки.
- Файл `*.snci`, с которым интерпретатор загружается.

Что обязательно для библиотеки

- Библиотека экспортирует две функции: `NCT_GetInterface` и `NCT_FreeMemory`. Если хотя бы одной из них нет, продукт библиотеку не загрузит.
- Строка, которую возвращает `GetIdentifier`, равна `LibraryID` в файле `*.snci`. Оба идентификатора, `LibraryID` и `ID`, уникальны.
- Для текущих продуктов собирается 64-разрядная библиотека, её имя указано в `LibraryFileName_x64`. 32-разрядная, с именем в `LibraryFileName_x32`, нужна только для 32-разрядного продукта старой версии.
- Дочерний интерпретатор получает родителя через `SetInterface` и вызывает его по правилам наследования интерпретатора.

Регистры можно задать и в файле `*.snci`, и в самой библиотеке. Удобнее объявлять их в библиотеке вместе с обработчиками: тогда новое слово — одна строка в списке регистров и один обработчик.

Цикл

1. Соберите исходники без изменений и проверьте, что библиотека загружается и моделирует ваши УП. Сохраните результат как исходный.
2. Перед поиском ошибки в моделировании проверьте экспорт функций, разрядность, идентификаторы и пару `*.snci` и `*.dll`: большая часть «не работает» начинается здесь.
3. Внесите одно небольшое изменение, соберите, прогоните все УП и сравните с предыдущим результатом — см. Проверка и подтверждение результата.
4. Отлаживайте под отладчиком вашего компилятора, присоединившись к CAM системе или NC Тюнеру, и читайте журнал взаимодействия с интерпретатором.

Защищённый интерпретатор

Интерпретатор, который проверяет свои данные через интерфейс защиты, описан в разделе Интерпретаторы с защитой. Проверяйте его из контейнера, в котором он будет поставляться: из обычной папки он данных защиты не получает. Без своих данных защищённый интерпретатор должен отказаться моделировать. Как именно он об этом сообщит, решает его автор: ассистент делает так, чтобы интерпретатор выдавал понятное сообщение, но уже существующий может молча давать пустую траекторию — тогда сначала проверьте данные защиты.

Чего не делать

- Ядро и библиотеки продукта не меняются ради того, чтобы интерпретатор заработал.
- Особенности реализации на Delphi не считаются частью двоичного интерфейса: в C++ и других языках интерфейс сверяют с библиотекой типов и с работающим интерпретатором.
- Компиляция — ещё не конец работы: библиотека должна загрузиться и смоделировать ваши программы.

Проверка и подтверждение результата

Помощь ИИ меняет то, как получается черновик. Она не меняет того, как принимается интерпретатор. Успешная сборка и моделирование без ошибок 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 УП через интерпретатор не прогонялись
Технически проверено	Интерпретатор загрузился, программы прошли, журнал и смоделированные команды сохранены, но не сравнивались и не разбирались
Проверено на эталонных программах	Все программы прогнаны до и после изменения, каждое отличие объяснено, предупреждения журнала прочитаны, проверены все версии продукта, для которых предназначен интерпретатор
Принято для станка	Вы или заказчик подтвердили моделирование для конкретного станка

Последнее состояние присваивает только человек. Никогда не описывайте результат как готовый на основании кода завершения или правдоподобной траектории.

Прогон программ

Перед любым прогоном включите журнал, как описано в расширенной настройке: без него сравнивать предупреждения не с чем.

Одну программу можно проверить вручную: операция на основе текста УП, выбранный интерпретатор, расчёт, затем моделирование и журнал. Прогонять так весь набор после каждого изменения долго. Ассистент может сделать небольшую программу прогона, которая без ваших щелчков открывает проект, моделирует каждую УП с интерпретатором и сохраняет смоделированные команды и журнал этого прогона. Строить её можно только на открытом

программном интерфейсе CAM системы для внешних приложений — CAMIPC: он описан в документации по CAM API, а библиотека для подключения к CAM системе входит в её установку. Если в документации нет нужного шага (например, как загрузить текст УП в операцию или выбрать её интерпретатор), а также в NC Тюнере, у которого такого описанного интерфейса нет, прогон ручной: вы рассчитываете операции, а ассистент готовит перечень программ и действий, затем собирает и сравнивает результаты. Программу прогона держите в репозитории интерпретатора и никогда не передавайте заказчику.

Прежде чем доверять программе прогона, проверьте её на заведомой ошибке: дайте ей УП с ошибкой и с неизвестным словом и убедитесь, что она насчитала обе. Счётчик, который читает не то поле журнала, показывает «0 ошибок» на чём угодно.

Программы прогоняйте по одной: прогоны в одном процессе CAM системы пишут в один файл журнала, а параллельно запущенные процессы могут вообще не запустить ядро.

Сравнение

1. Сравните сводки всех программ до и после. Изменившееся число предупреждений или команд в программе, которой изменение не касалось, объясняйте первым.
2. Для каждой изменившейся программы начинайте с первого отличия и проследите его до кадра и слова, которое его дало. Непреднамеренное отличие важнее преднамеренного.
3. Читайте сами предупреждения, а не только их число: каждое называет слово, ось или цикл.
4. Граничные случаи, которых нет в программах, проверьте короткой программой: слово во всех написаниях из перечня, на границах, пропущенное, повторённое. Собирайте её из заголовка настоящей программы той же операции и заменяйте только проверяемые кадры — без заголовка программа может не дать ни одного движения.

Отличие, вызванное самой проверкой, а не интерпретатором — другая система координат тестовой операции, другой станок в проекте, — не результат: назовите его и исключите.

Проверка через постпроцессор

Если есть постпроцессор заказчика и проект CAM системы, замкните круг: сформируйте УП из операций проекта, прогоните их через интерпретатор и сравните моделирование с траекторией самой операции. Расхождение может оказаться ошибкой постпроцессора. Интерпретатор моделирует то, что сделала бы стойка с программой, а не то, что задумывала операция: о такой ошибке сообщите автору постпроцессора, а не прячьте её в интерпретаторе.

Проверка на безопасность

Перед передачей просмотрите каждое изменение на случаи, когда моделирование выглядит безопаснее станка: пропущенный отвод, угаданная начальная точка, цикл, который не моделируется, но оставляет операцию с виду законченной. Каждый такой случай проверьте короткой программой. После каждого круга правок по замечаниям снова прогоните весь набор программ.

Подготовка к передаче

Заказчик получает папку на каждую версию продукта: файл `*.snci`, библиотеку интерпретатора и `InterpreterSettings.ini`, а файл `*.deps.json` — только если без него интерпретатор не загружается (это проверяется на неизменённой установке). Ассистент копирует нужные файлы по именам, а не архивирует папку сборки целиком: в ней лежат отладочные файлы и служебные файлы пакета SDK. К ним — README на языке заказчика: что поддерживается, ограничения, настройки, какая папка для какой версии продукта, нужна ли продукту лицензионная опция для загрузки собственного интерпретатора и какие упрощения сделаны намеренно, чтобы их не приняли за ошибки.

Перед отправкой проверьте:

- в файлах нет следов вашего компьютера: имён пользователей, домашних папок, внутренних серверов. Библиотеку для передачи собирают без отладочной информации или с заменой путей: обычная сборка записывает в неё путь к файлу `*.pdb` на вашем компьютере;
- в файлах нет имён, которые не относятся к этому заказчику: имён других заказчиков, вашего компьютера и его пользователей, внутренних серверов и источников пакетов — ни в именах файлов, путях, настройках, конфигурациях запуска, ссылках на пакеты SDK, ни в README. Искать нужно по содержимому каждого файла, в кодировках UTF-8 и UTF-16, а не только по именам; то же — для всего, что отправляется вместе с пакетом;
- проект, если заказчик просит исходники, собирается из чистой копии с выпущенным пакетом SDK;
- интерпретатор работает на неизменённой установке продукта.

Собирайте пакет из зафиксированных в истории исходников и записывайте, что, когда и кому отправлено.

Чего ассистент делать не должен

Он не должен присваивать результату состояние «Принято для станка», менять эталонные программы, станок или проект, чтобы сравнение сошлось, утверждать проверку версии продукта, которую не запускал, и отправлять пакет без вашего разрешения.

Устранение проблем

Ассистент не видит навыки

Спросите, какие навыки у него сейчас есть. Если шести навыков для интерпретаторов среди них нет, проверьте, что папки навыков лежат в той папке, которую действительно читает ваш клиент, что в каждой есть файл `SKILL.md` и что клиент их подхватил: Claude Code видит новые навыки сразу, а если папки навыков до копирования не было — после команды `/reload-skills`; Kilo Code — в новой сессии или после команды `/reload`; другие клиенты — по своей документации, часто только после перезапуска. Где какой клиент ищет навыки — в расширенной настройке.

Ассистент отвечает по памяти вместо программ

Так бывает, когда в запросе не названы файлы или когда от ответа не требуют ссылок на данные. Укажите программы и руководство явно, спросите, из какого файла и какого кадра он взял слово, и попросите работать по навыкам — они требуют сначала прочитать программы.

Ассистент докладывает об успехе, ничего не запустив

Сборка — это не результат. Спросите, какие программы прогнаны, где журнал этого прогона и чем результат отличается от предыдущего. Если этого нет, состояние — Не проверено, см. Проверка и подтверждение результата.

Интерпретатор не загружается

Сначала прочитайте, какой Файл интерпретатора указан на вкладке Стратегия параметров операции, и журнал CAM системы. Если интерпретатор не загружен, каждое слово программы будет ошибкой.

Любую неудачу загрузки интерпретатора на C# CAM система описывает одинаково — сообщением «Внешняя библиотека не поддерживает интерфейс NCT:» с путём к библиотеке; то же сообщение выдаётся для нативной библиотеки без нужных экспортов. Настоящая причина в сообщение не попадает. Чаще всего это:

- рядом с файлом `*.snci` нет библиотеки, названной в нём, или нет её зависимостей;
- библиотека собрана для другой версии .NET, чем нужна этой версии продукта;
- при создании объекта интерпретатора возникло исключение — обычно при создании регистров.

Попросите ассистента проверить файлы и создать объект интерпретатора в маленькой консольной программе с тем же пакетом SDK — так видно настоящее сообщение об ошибке.

Для нативного интерпретатора проверьте разрядность библиотеки, экспорт `NCT_GetInterface` и `NCT_FreeMemory` и совпадение `GetIdentifier` с `LibraryID` — см. Рабочий цикл нативного интерпретатора.

Два интерпретатора путаются

CAM система различает интерпретаторы по идентификаторам в файле `*.snci`. Скопированный интерпретатор с теми же `LibraryID` и `ID` неотличим от оригинала: моделирует то один, то другой. Дайте копии новые идентификаторы.

Точка останова не срабатывает

Проверьте по порядку: рядом с библиотекой есть файл `*.pdb` (сборка без отладочной информации делает любую точку останова «мёртвой»); операция использует именно ваш интерпретатор; сборка перед запуском прошла успешно — откройте панель терминала и прочитайте её результат; в CAM систему передан файл `*.snci` из папки той версии, которую вы запустили.

Настройки из InterpreterSettings.ini не действуют

В шаблонах есть известные ошибки чтения `InterpreterSettings.ini`: файл может не читаться совсем, а в журнале остаётся только информационная строка, начинающаяся с `Error reading settings file`. Отфильтрованный по ошибкам журнал её не покажет. Попросите ассистента проверить шаблон, из которого создан интерпретатор, — навык `develop-dotnet-interpreter` перечисляет эти ошибки и их исправления.

Ось моделируется не там или не моделируется вовсе

Помощник из пакета SDK, через который шаблоны .NET находят оси станка, предупреждает о неизвестном имени оси один раз, а дальше молча пропускает её значения. Проверьте, что имя оси в программе совпадает с адресом оси в схеме станка, и что сопоставление имён работает для всех написаний из перечня слов: `B90`, `B=90`, `B1=90`.

Второй канал двигает оси первого или не моделируется

Интерпретатор должен двигать те оси, которые ему передаёт CAM система для канала его программы: для второго канала — например, `X2`, `Y2`, `Z2`. Если второй канал двигает оси первого или его траектория сжимается в точку, сравните эти оси с осями, которые интерпретатор передаёт в команды движения. Расхождение — ошибка интерпретатора или его файла `*.snci`, а не CAM системы.

Синхронизацию каналов проверяйте в CAM системе: в зависимости от версии и лицензии NC Тюнер может моделировать каналы не одновременно, а по очереди, и тогда точки ожидания ничего не задерживают.

Пустая траектория без ошибки

Частые причины:

- в программе нет заголовка, с которым её выводит постпроцессор: проверочная программа, собранная из одних проверяемых кадров, может не дать ни одного движения — собирайте её из заголовка настоящей программы той же операции;
- интерпретатор-родитель предназначен только для импорта траектории (в списке интерпретаторов у таких указано «Только для импорта траектории»);
- имя оси в программе не совпадает с адресом оси в схеме станка (см. раздел выше);
- интерпретатор с защитой не получил своих данных защиты — например, запущен из обычной папки, а не из контейнера — и написан так, что тогда молча ничего не моделирует. См. Интерпретаторы с защитой.

Результат не совпадает с тем, что даёт другая версия продукта

У каждой версии CAM системы и NC Тюнера своя версия .NET и своя версия пакета SDK (см. Версии продукта). Проверьте, в какой версии продукта шёл прогон и для какой собрана библиотека, — см. Несколько версий продукта на одном компьютере.

Документация, которую читает ассистент, устарела

Локальная копия — это снимок: скачайте архив заново, когда выходит новая версия, и записывайте, на какой версии основано решение. Подробности, зависящие от версии, — версию пакета SDK, имена интерфейсов — 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 или читайте, когда команде нужен воспроизводимый результат.

О чём идёт речь

Этот раздел обращён к ассистенту, который читает страницу, чтобы настроить инструменты: вот предметная область, в которой он работает.

Стойка ЧПУ выполняет управляющую программу (УП) — текст из кадров, каждый кадр состоит из слов: адресов с числами, G- и M-функций, вызовов циклов. CAM система умеет моделировать такую программу: показать траекторию инструмента и движения станка, которые она вызовет. Для этого ей нужен интерпретатор — библиотека, которая объясняет ядру CAM системы, что значит каждое слово на конкретной стойке. Каждой стойке и её диалекту нужен свой интерпретатор, и доработка интерпретатора — работа человека, которому помогает ассистент. Обычно это технолог, инженер поддержки или тестировщик, который знает станок гораздо лучше, чем программирование.

Интерпретатор состоит из двух файлов: файла настроек `*.snci` и библиотеки `*.dll`. Библиотека бывает двух видов:

- .NET — проект на C#, созданный из шаблона CAM системы, — путь по умолчанию;
- нативная — библиотека на Delphi, C++ или другом компилируемом языке — только по прямой просьбе пользователя или для уже нативного интерпретатора.

Для чего настраиваются инструменты: читать настоящие УП и руководство стойки, править и собирать интерпретатор, прогонять через него программы в CAM системе или NC Тюнере и сравнивать результат с предыдущим. Правильность моделирования определяет этот человек; ассистент готовит основания, по которым он судит.

Настройка состоит из трёх частей, в таком порядке: навыки, средства сборки и отладки, рабочая папка. В сообщении о проблеме указывают версии CAM системы или NC Тюнера, .NET SDK, ИИ-клиента и навыков: большая часть сообщений «вчера работало» объясняется различием версий.

Навыки в разных ИИ-клиентах

Навыки публикуются с этой документацией: архив ветки скачивается по ссылке со страницы Разработка интерпретаторов с помощью ИИ, навыки лежат в нём в папке `src/ai-assisted-development/skills/` внутри корневой папки архива `public-docs`. Навык — это папка с именем

навыка и файлом `SKILL.md` внутри. Копируйте все шесть папок сразу: навыки ссылаются друг на друга по названиям.

Клиент	Куда скопировать
Kilo Code	`.kilo/skills/` рабочей папки или `%USERPROFILE%\.kilo\skills\`, чтобы навыки были во всех проектах; новая сессия их подхватит, команда `/reload` — без перезапуска
Claude Code	`.claude/skills/` рабочей папки или `%USERPROFILE%\.claude\skills\`, чтобы навыки были во всех проектах; новые навыки видны сразу, а если папки навыков до этого не было — после команды `/reload-skills`
Другие клиенты	Своя папка навыков рабочей папки или пользователя, запись в конфигурации агента или пункт настройки — по документации клиента

Если в клиенте нет механизма навыков, вставьте содержимое `SKILL.md` нужных навыков в чат или сошлитесь на их файлы из файла инструкций клиента (`AGENTS.md`, `CLAUDE.md` и подобных).

Каким бы способом вы их ни подключили, проверьте, что они загружены: спросите у ассистента, какие навыки он сейчас видит. Навыки установлены для пользователя — доступны при любой открытой папке; установлены для рабочей папки — только в ней.

Навыки обновляются вместе с документацией. Заменяйте папки целиком, а не правьте их на месте: свои дополнения держите отдельными навыками рядом с ними (см. организацию рабочего пространства).

Средства сборки и отладки

Для интерпретатора на C# нужны:

- .NET SDK той версии .NET, которую использует нужная версия продукта (см. Версии продукта). Проверьте командой `dotnet --list-sdks`.
- Visual Studio Code с расширением C# от Microsoft: в нём отладчик, на который рассчитаны конфигурации запуска шаблонов. Расширение XML Language Support by Red Hat удобно для правки файлов `*.snci`, но не обязательно.

- Путь к программе продукта для отладки. Шаблон запускает CAM систему или NC Тюнер из папки, записанной в переменных среды `NCKernelDir` (CAM система) и `TunerDir` (NC Тюнер). Окно Создать интерпретатор УП (дальше — создатель интерпретатора) записывает одну из них — переменную того продукта, из которого оно открыто, — когда в нём нажимают кнопку Задать путь к файлу СПРУТКАМ (в NC Тюнере на кнопке — его название) и выбирают исполняемый файл продукта; после этого нужно выйти из сеанса Windows и войти снова или хотя бы запустить Visual Studio Code заново из меню «Пуск». Если переменной нет, путь можно записать в `launch.json` — стандартную папку установки продукта, а не папку, которая есть только на вашем компьютере. Имя исполняемого файла CAM системы в `launch.json` подставляет создатель интерпретатора; в шаблоне, скопированном вручную, без создателя, остаётся `${env:CAMName}`, которую ничто не задаёт, — впишите вместо неё имя файла. Подробно — Отладка интерпретатора в Visual Studio Code.
- Журнал: в окне Системные настройки на странице Дополнительные, в группе Логирование, включены опции Вести журнал и Вести журнал взаимодействия с интерпретатором (вторая доступна, только когда включена первая); в NC Тюнере — так же. Без них в журнале нет ни сообщений ядра о прочитанных словах, ни строк, которые пишет интерпретатор. Подробно — см. Журнал взаимодействия с интерпретатором.

Для нативного интерпретатора нужен его компилятор и пакеты с объявлениями интерфейсов SDK — см. Рабочий цикл нативного интерпретатора.

Чего из перечисленного не хватает, нужно назвать, а не подставлять догадку; недостающее устанавливается только с разрешения пользователя.

Файл инструкций рабочей папки

Требования, действующие всегда, размещайте там, где ассистент читает всегда: в файле инструкций рабочей папки (`AGENTS.md` или файл, который читает ваш клиент) или в навыках. Что в нём держать, описано в организации рабочего пространства.

Для интерпретатора на C# открытой папкой должна быть папка самого интерпретатора, а клиенты по-разному ищут файлы инструкций. Claude Code читает `CLAUDE.md` (а если его нет — `AGENTS.md`) и в открытой папке, и во всех папках выше неё, поэтому памятка рабочей папки, лежащей выше папки интерпретатора, видна ему и так. Kilo Code читает `AGENTS.md` только в открытой папке, поэтому:

- в Kilo Code можно перечислить дополнительные файлы инструкций в глобальной настройке `instructions` файла `kilo.jsonc` (обычно `%USERPROFILE%\.config\kilo\kilo.jsonc`) — пропишите там полный путь к `AGENTS.md` рабочей папки, и он будет читаться в любом проекте;

- если клиент так не умеет, положите в папку интерпретатора короткий `AGENTS.md` со ссылкой на общий файл и папку заметок.

На практике оправдывают себя такие правила:

- сначала прочитать настоящие программы и руководство стойки и называть, откуда взято каждое слово;
- C# (.NET) по умолчанию, нативный интерпретатор — только по прямой просьбе;
- вариации — в `InterpreterSettings.ini`, а не в код;
- одно изменение за раз, затем сборка, прогон всех программ и сравнение с предыдущим результатом — до любого доклада о готовности;
- не менять установленный продукт, его шаблоны и библиотеки;
- не отправлять интерпретатор заказчику без разрешения.

Несколько версий продукта на одном компьютере

Интерпретатор работает в той версии продукта, для которой собран: у каждой версии свой .NET и своя версия пакета SDK (см. Версии продукта). Определитесь, для каких версий разрабатывается интерпретатор, и прогоняйте программы в каждой из них. Один проект может собираться для нескольких версий — по папке результата на каждую и по конфигурации запуска на каждую. Если версия не установлена на этом компьютере, проверку в ней нельзя считать сделанной: так и скажите.

Переменные `NCKernelDir` и `TunerDir` указывают на одну установку. Если рядом стоят выпущенная и бета-версия, проверьте, на какую указывает переменная, прежде чем сравнивать результаты: прогон в другой версии завершается успешно и просто даёт другой результат.

Отдельное рабочее место для ассистента

- Копия или ветвь интерпретатора, а не файлы, которые уже у заказчика.
- Своя папка для результатов прогонов, вне дерева исходников.
- Прогон из копии результата сборки, чтобы новая сборка не подменила библиотеку посреди набора программ.
- Сокращённые программы и проекты вместо полных архивов заказчика, где этого достаточно.
- Никаких паролей, токенов, файлов лицензий, архивов заказчиков и посторонних репозиториев в рабочей папке и в файлах инструкций — подробнее в разделе Чего в ней быть не должно.

Автоматические проверки

Программа прогона, которая без щелчков моделирует все УП и сохраняет смоделированные команды и журнал, делает проверку воспроизводимой: фиксированные программы, фиксированный проект, фиксированная версия интерпретатора. Её делает ассистент по навыку `verify-interpretation` на открытом программном интерфейсе CAM системы; правила, без которых ей нельзя доверять, и случаи, когда прогон остаётся ручным, — в разделе Прогон программ. Автоматика может сообщить, что моделирование изменилось, но никогда — что изменение верно: различия разбирает человек.

Приложения

См. вложенные страницы.

Перечень интерпретаторов

В настоящее время для использования доступны интерпретаторы следующих систем ЧПУ.

Группа станка	Система ЧПУ	Комментарий	Примечание	Список реализованных команд* *(Команды могут быть реализованы не в полном объёме, см. примечание)*
Фрезерный	APT	Только для импорта траектории		FEDRAT: MMPM, MMPR; RAPID, MULTAX, GOTO, ARCSLP XY, YZ, ZX, CLW, CCLW, CIRCLE, TOOLNO, LOADTL, SPINDL, COOLNT, CUTCOM, LENGTH, PLUS, RADIUS, RIGHT, LEFT
	Apt_Simplify_3D	Только для импорта траектории		FEDRAT: MMPM, MMPR; M101, M103, RAPID, GOTO
	ISO	Только для импорта траектории		G0, G01, G02, G03, G4, G90, G91, G17, G18, G19, X, Y, Z, I, J, K, R, G20, G21, S, M3, M4, M5, M13, M14, M0, M1, M2, M30,

				M7, M8, M9, T, F, G43, G44, G49, H, G40, G41, G42, D, G53, G10, G54..59, G73, G74, G76, G80, G81, G82, G83, G84, G85, G86, G87, G88, G89, G98, G99, P, G94, G95
	Global control	Только для импорта траектории	Необходима дополнительная лицензия	G0, G1, G2, G3, G90, G91, G161, G162, G70, G71, G04, X, Y, Z, I, J, G74, M0, M2, M30, F, G40, G41, G42, D
	Fanuc 30i	Для моделирования и импорта траектории		N, G0, G01, G02, G03, G90, G91, G17, G18, G19, G20, G21, X, Y, Z, A, B, C, I, J, K, R, S, M3, M4, M5, M0, M1, M2, M30, M8, M9, M6, T, F, G43, G43.4, G44, G49, H, G41, G42, D, G53, G92, G54..G59, G73, G74, G76, G80, G81, G82, G83, G84, G85, G86, G87, G88, G89, G98, G99, G94, G95, G96, G97, G68.2, G69, G53.1, M98,

			M99, O, G4, P, L, G52
Fanuc 0i EmcoMill E600	Для моделирования и импорта траектории	Необходима дополнительная лицензия	Команды Fanuc 30i (интерпретатор-наследник), а также M13, M14 (включение шпинделя с СОЖ), G65 (выполняется как G83)
Fanuc 0i MF	Для моделирования и импорта траектории	Необходима дополнительная лицензия	Команды Fanuc 30i (интерпретатор-наследник)
Fanuc Mill Custom	Для моделирования и импорта траектории	Настраиваемый интерпретатор	N, G0, G01, G02, G03, G90, G91, G17, G18, G19, G20, G21, X, Y, Z, A, B, C, I, J, K, R, S, M3, M4, M5, M0, M1, M2, M30, M8, M9, M6, T, F, G43, G43.4, G44, G49, H, G40, G41, G42, D, G53, G92, G54..G59, G73, G74, G76, G80, G81, G82, G83, G84, G85, G86, G87, G88, G89, G98, G99, G94, G95, G96, G97, G68.2, G69, G53.1, M98, M99, O, P, L, G4,

				G52; коды смены инструмента, СОЖ, включения шпинделя с СОЖ, тормозов осей и вспомогательные коды задаются в настройках интерпретатора
	Haas VF-2	Для моделирования и импорта траектории		G0, G01, G02, G03, G04, G90, G91, G17, G18, G19, X, Y, Z, A, B, C, I, J, K, R, S, M3, M4, M5, M0, M1, M2, M30, M8, M9, T, F, G43, G44, G49, G40, G41, G42, D, G53, G92, G54..59, G73, G74, G76, G77, G80, G81, G82, G83, G84, G85, G86, G89, G98, G99, G52
	Heidenhain iTNC 530	Для моделирования и импорта траектории		L, FMAX, F, X, Y, Z, IX, IY, IZ, A, B, LC, IA, IB, IC, CC, C, CP, CR, R, DR-, DR+, IPA, MM, INCH, M91, S, M8, M9, M3, M4, M5, M13, M14, M0, M1, M2, M30, DL+, DR2+,

				RL, RR,R0, CYCL CALL, M99, M89, CALL LBL, Q, CYCL DEF200, 201, 202, 203, 205, 206, 207, 7, 9, 13, ORIENT, ANGLE
	HNC-8	Для моделирования и импорта траектории	Необходима дополнительная лицензия	N, G0, G01, G02, G03, G90, G91, G17, G18, G19, G20, G21, X, Y, Z, A, B, C, I, J, K, R, S, M3, M4, M5, M0, M1, M2, M30, M7, M8, M9, M6, T, F, G43, G43.4, G44, G49, H, G41, G42, D, G53, G92, G54..G59, G73, G74, G76, G80, G81, G82, G83, G84, G85, G86, G87, G88, G89, G98, G99, G94, G95, G96, G97, G68.2, G69, G53.1, M98, M99, O, G4, P, L, M19, M12, M12.1, M13, M13.1
	Mazatrol SmoothG	Для моделирования и импорта траектории	Необходима дополнительная лицензия	G00, G01, G02, G03, X, Y, Z, G90, G91, F, I, J, K, R, G17, G18, G19, G53, G28, M30, M0, M1, M5, M6,

				M3, M4, S, M8, M9, G40, G41, G42, G4, G94, G95, G50, G52, G97, T, G98, G99, G80, G82, G81, G88, G89, G85, G87, G83, G84, G73, G86, G76, G74, G54..G59
	Mitsubishi M80a	Для моделирования и импорта траектории	Необходима дополнительная лицензия	N, G0, G01, G02, G03, G90, G91, G17, G18, G19, G20, G21, X, Y, Z, A, B, C, I, J, K, R, S, M3, M4, M5, M0, M1, M2, M30, M8, M9, M6, T, F, G43, G43.4, G44, G49, H, G41, G42, D, G53, G92, G54..G59, G73, G74, G76, G80, G81, G82, G83, G84, G85, G86, G87, G88, G89, G98, G99, G94, G95, G96, G97, G68.2, G69, G53.1, M98, M99, O, G4, P, L, M19
	NC210	Для моделирования и импорта траектории	Необходима дополнительная лицензия	G0, G01, G02, G03, G4, G9, TMR, G90, G91, G17, G18, G19, X, Y, Z, A, B, C, I, J, K, R, M3, M4,

				M5, M13, M14, M19, M0, M1, M2, M30, M7, M8, M9, T, G94, G95, F, G40, G41, G42, G79, G80, G81, G82, G83, G84
	Sinumerik 840D	Для моделирования и импорта траектории		N, G0, G01, G02, G03, G90, G91, G17, G18, G19, X, Y, Z, A, B, C, I, J, K, CR, TURN, F, S, T, M6, M3, M4, M5, M0, M1, M2, M30, M8, M9, G40, G41, G42, G53, G153, SUPA, G54..G57, G70, G71, G700, G710, G58, G59, TRANS, ATRANS, G4, G94, G95, MCALL, M17, TRAORI, TRAFOOF, CYCLE800, AC, IC, CYCLE81, CYCLE82, CYCLE83, CYCLE84, CYCLE840, CYCLE85, CYCLE86, CYCLE87, CYCLE88, CYCLE89, A3, B3, C3, CALL(Subprogram), POCKET4

Sinumerik 828D EmcoMill E350	Для моделирования и импорта траектории	Необходима дополнительная лицензия	Команды Sinumerik 840D (интерпретатор-наследник), а также M13, M14 (включение шпинделя с СОЖ)
Sinumerik 840D MasterMill HPC3	Для моделирования и импорта траектории	Необходима дополнительная лицензия	Команды Sinumerik 840D (интерпретатор-наследник), а также M80, M82 (выбор головы), M101, H, RTCPON, RTCPOF
Tormach PCNC Mach3	Для моделирования и импорта траектории		G4, P, G53, G92, G0, G1, G2, G3, X, Y, Z, A, B, C, I, J, K, R, G73, G80, G81, G82, G83, G84, G85, G86, G87, G88, G89, G17, G18, G19, G90, G91, G93, G94, G95, G20, G21, G40, G41, G42, D, G43, G44, G49, H, G98, G99, G54..G59, M0, M1, M2, M30, M3, M4, M5, M8, M9, F, S, T, M98, M99, O, P, L
Tormach PCNC PathPilot	Для моделирования		G4, P, G53, G92, G0, G1, G2, G3, X, Y, Z, A, B, C, I,

		и импорта траектории		J, K, R, G73, G80, G81, G82, G83, G84, G85, G86, G87, G88, G89, G17, G18, G19, G90, G91, G93, G94, G95, G20, G21, G40, G41, G42, D, G43, G49, H, G98, G99, G54..G59, M0, M1, M2, M30, M3, M4, M5, M8, M9, F, S, T, M98, M99, O, P, L

Токарный	HNC-8	Для моделирования и импорта траектории	Необходима дополнительная лицензия	G0, G01, G02, G03, G04, G90, G91, G17,G18, G19, G20, G21, X, Y, Z, U, W, A, B, C, I, J, K, R, S, M3, M4, M5, M0, M1, M2, M30, M7, M8, M9, M6, T, F, G43, G43.4, G44, G49, H, G40, G41, G42, D, G53, G92, G54..G59, G80, G83, G84, G98, G99, G94, G95, G96, G97, G68.2, G69, G53.1, M98, M99, O, G4, M19, G12, G12.1, G13, G13.1, G32, G52, G70, G71, G72, G73, G74, G75, G76
	Mazatrol SmoothC	Для моделирования и импорта траектории	Необходима дополнительная лицензия	G00, G01, G02, G03, X, Z, U, W, I, K, G17, G18, G19, F, R, G98, G99, G4, S, G96, G97, G50, M3, M4, M5, M9, M0, M1, M30, M8, G52, G54..G59, G53, G28, T, G80, G83, G84, G85, G70, G71, G72, G73, G74, G75, G76, G32

Токарно-фрезерный	Fanuc 21i	Для моделирования и импорта траектории	Необходима дополнительная лицензия	G4, G50, G52, G53, G92, G70, G71, G72, G73, G74, G75, G76, G92, G0, G01, G02, G03, G12.1, G13.1, G112, G113, G7.1, G107, X, Y, Z, B, C, U, V, W, H, I, J, K, R, G32, G96, G97, G98, G99, G20, G21, G40, G41, G42, G80, G83, G84, G85, G87, G88, G89, G28, M50, M51, G54..G59, G17, G18, G19, M0, M1, M2, M30, M3, M33, M4, M34, M5, M35, M8, M9, M98, P, M99, O
	Fanuc 0i EmcoTurn E45	Для моделирования и импорта траектории	Необходима дополнительная лицензия	Команды Fanuc 21i (интерпретатор-наследник) с другими номерами: G70, G71 (единицы измерения), G72..G78 (многократно повторяемые циклы), G21 (цикл нарезания резьбы), G94, G95 (тип

				подачи), G192 (ограничение частоты вращения шпинделя), M303, M304, M305 (вместо M33, M34, M35)
	Fanuc 0i Schaublin SA-125-CN	Для моделирования и импорта траектории	Необходима дополнительная лицензия	Команды Fanuc 21i (интерпретатор-наследник) с другими номерами: G70, G71 (единицы измерения), G72..G78 (многократно повторяемые циклы), G21 (цикл нарезания резьбы), G94, G95 (тип подачи), G92 (ограничение частоты вращения шпинделя), M203, M204, M105 (вместо M33, M34, M35); Q в цикле G83 задаётся в микрометрах
	FS-8	Для моделирования и импорта траектории	Необходима дополнительная лицензия	N, G0, G01, G02, G03, G90, G91, G17, G18, G19, X, Y, Z, A, B, C, I, J, K, CR, TURN, F,

				S, T, G40, G41, G42, G53, G153, SUPA, G54..G57, G70, G71, G700, G710, G58, G59, TRANS, ATRANS, G4, G94, G95, MCALL, M17, TRAORI, TRAFOOF, CYCLE800, AC, IC, CYCLE81, CYCLE82, CYCLE83, CYCLE84, CYCLE840, CYCLE85, CYCLE86, CYCLE87, CYCLE88, CYCLE89, A3, B3, C3, CALL(Subprogram), DIAM90, DIAMON, DIAMOF, LIMS
	HAAS ST20Y	Для моделирования и импорта траектории	Необходима дополнительная лицензия	G0, G01, G02, G03, G04, G17, G18, G19, X, Y, Z, I, J, K, L, R, C, G40, G41, G42, M19, M3, M133, M4, M134, G96, G97, S, P, M5, M135, M8, M9, G98, G99, G50, M1, M30, G54..G59, G28, G53, T, N, M154, M155, U, V, W, G4, G112, G113,

				G70, G71, G72, G73, G74, G75, G76, G80, G82, G81, G88, G89, G85, G87, G83, G84, G86, G186, G184, G95, G195, G196, G241, G242, G243
	NC220	Для моделирования и импорта траектории	Необходима дополнительная лицензия	DFP, EPF, UCG, FIL, SPA, SPP, CLP, G0, G1, G2, G3, G4, G9, TMR, G90, G91, G17, G18, G19, G70, G71, X, Y, Z, A, B, C, I, J, K, R, S, M3, M4, M5, M13, M14, M19, G96, G97, SSL, M41, M42, M43, M0, M1, M2, M30, M7, M8, M9, M6, T, G94, G95, F, G40, G41, G42, G79, G33, G80, G81, G82, G83, G84
	Okuma OSP-P300	Для моделирования и импорта траектории	Необходима дополнительная лицензия	G0, G01, G02, G03, G101, G102, G103, G132, G133, G90, G91, G17, G18, G19, G137, G138, G119, G136, X, Y, Z, C, I, J, K, L, F, S, SB, T, M3, M4, M13,

				M14, M5, M12, M0, M1, M2, M30, M8, M9, M174, M175, M262, M263, M41, M42, M43, M44, M146, M147, M109, M110, G94, G95, G96, G97, G50, G33, G4, G77, G78, G178, G179, G180, G181, G182, G183, G184, M136, G71, G72, G73, G74, G85, G86, G87, M22, M23, M32, M33, M34, M73, M74, M75, G80, G81, G82, E
	Sinumerik 840D	Для моделирования и импорта траектории		N, G0, G01, G02, G03, G90, G91, G17, G18, G19, X, Y, Z, A, B, C, I, J, K, CR, TURN, F, S, S2, S3, S4, T, M1, M2, M3, M4, M5, M8, M9, M30, M23, M24, M25, M33, M34, M35, M43, M44, M45, G40, G41, G42, D, G53, G153, SUPA, G54..G57, G70, G71, G700, G710, G58, G59, TRANS, ATRANS,

				G4, G94, G95, G96, G97, LIMS, DIAM90, DIAMON, DIAMOF, G63, G33, SF, TRAFOOF, MCALL, STOPRE, AC, IC, CYCLE81, CYCLE82, CYCLE83, CYCLE84, CYCLE85, CYCLE86, CYCLE87, CYCLE88, CYCLE89, CYCLE840, CYCLE93, CYCLE95, CYCLE97, CYCLE90, SETMS, SPCO, SPCOF, SPOS, TRANSMIT
Робот	ABB robot	Для моделирования и импорта траектории		MoveAbsJ, \Cons, \WObj, MoveLDO, MoveL, \Corr, MoveC, PERS, v5, v10⋯.V7000
	Estun	Для моделирования и импорта траектории	Необходима дополнительная лицензия	A1..A6, E1, J, MovL P, MovL V, MovC P, MovC V, MovJ J, MovJ V, End, Start
	Fanuc robot (контроллер R-30iB)	Для моделирования и импорта траектории		Joint, Linear, Circular, J1..J6, E1..E3, UFRAME, UTOOL_NUM, J, L, C, FINE, CNT,

				ACC, X, Y, Z, W, P, R, /POS, /EN, UF, UT, F, N, U, D, T, B, deg, mm, GP1, GP2, /END, /MN, %, sec, msec, mm/sec, cm/min, inch/min, A1..A6
	Kuka robot	Для моделирования и импорта траектории		LIN, C_DIS, CIRC, PTP, $VEL.CP, DEF, END, $TOOL, TOOL_DATA, $BASE, BASE_DATA
	Motoman robot	Для моделирования и импорта траектории		/JOB, S, L, U, R, B, T, E1..E6, E, ///COMM, ///TOOL, ///USER, //POS, //INST, ///POSTYPE, ///PULSE, ///RECTAN, C, EC, MOVL, MOVJ, MOVC, SMOVL, EIMOVL, EIMOVC, UF, V, VJ, +MOVJ
	Nachi robot (AW Format)	Для моделирования и импорта траектории	Необходима дополнительная лицензия	MOVEJ L, MOVE C, MOVE L, S, A, H, A1..A6, REM, CALL, STOP, END

Примечание: Все интерпретаторы реализуют только перечень команд, генерируемый соответствующими постпроцессорами из дистрибутива CAM системы.

Интерпретаторы "Только для импорта траектории" не поддерживают соответствие строка УП - траектория движения инструмента.

ВНИМАНИЕ

При выборе интерпретатора обратите внимание на его предназначение (поле Предназначение на панели предварительного просмотра).

Выбранный интерпретатор должен быть предназначен для моделирования по управляющей программе. В ином случае траектория движения инструмента может быть некорректной (иметь смещение относительно системы координат заготовки, сдублированные подходы/отходы, неверную начальную позицию и прочее).

ВНИМАНИЕ

Некоторые команды могут быть реализованы частично и не иметь всех параметров для работы, прежде чем начать работу с интерпретатором просьба убедиться в том что вам действительно подходит команда в текущей реализации. Если возникнут вопросы обращайтесь к поставщику услуг.