ПОСТПРОЦЕССИРОВАНИЕ В БИБЛИОТЕКЕ «МОДУЛЬ ЧПУ. ТОКАРНАЯ ОБРАБОТКА»

1. УПРАВЛЕНИЕ КАТАЛОГОМ ПОСТПРОЦЕССОРОВ

Постпроцессор представляет собой программу на скриптовом языке Python, которая транслирует текст исходной управляющей программы в коды программирования конкретного устройства ЧПУ. Базовый вариант библиотеки включает в свой комплект следующий набор постпроцессоров:

Маяк 600 Т;
Балт-Систем;
НЦ-31;
FANUC Series 0i-MODEL D;
SINUMERIK 802D;
FAGOR CNC 8035 T.

Файлы постпроцессоров расположены в отдельном каталоге POSTPROCESSORS внутри папки, где находится файл rtw библиотеки. Управление каталогом постпроцессоров реализовано через команду библиотеки «Постпроцессоры». При вызове этой команды появляется диалоговая панель со списком постпроцессоров (рисунок 1). На панели доступны операции редактирования, создания, загрузки и удаления файла постпроцессора.

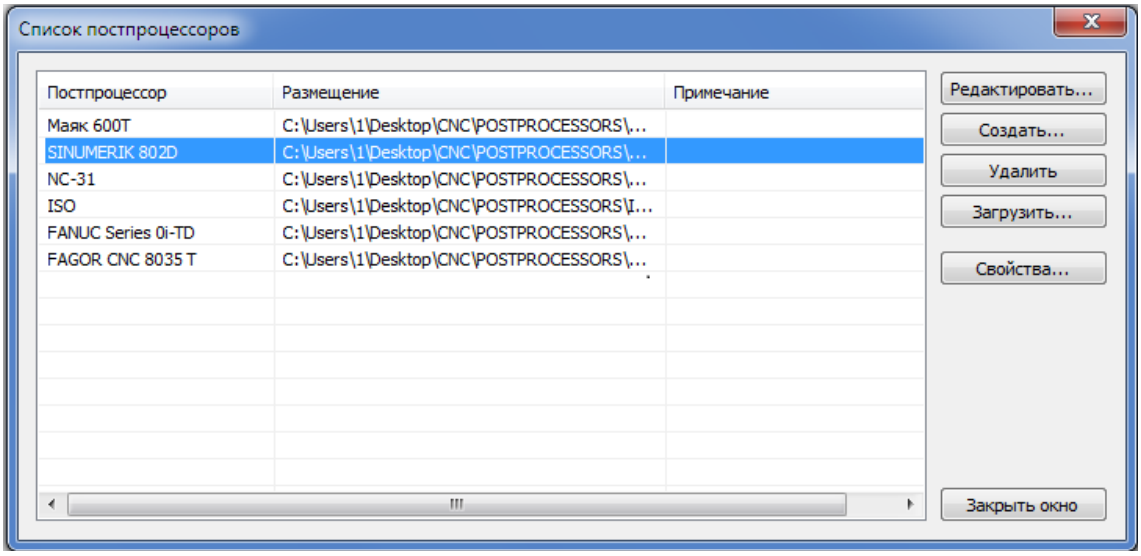

Рисунок 1. Панель для управления каталогом постпроцессоров

При нажатии кнопки «Редактировать...» запускается отладочная среда языка Python, в окне которой выводится исходный текст постпроцессора.

При нажатии кнопки «Создать...» запускается отладочная среда языка Python, в окне которой выводится текст шаблона постпроцессора. Используя шаблон, пользователь может разработать собственный постпроцессор.

При нажатии кнопки «Удалить» сначала запрашивается подтверждение на удаление файла постпроцессора. При положительном ответе пользователя постпроцессор удаляется с диска.

При нажатии кнопки «Загрузить...» открывается диалог выбора файла постпроцессора. Выбранный пользователем файл копируется в каталог постпроцессоров.

Кнопка «Свойства...» предназначена для вывода списка станочных циклов, поддерживаемых данным постпроцессором.

Для настройки пути к интерпретатору Python вызовите команду библиотеки **Настройки** и в соответствующей строке укажите полный путь к исполняемому файлу Python.

2. СХЕМА ВЗАИМОДЕЙСТВИЯ БИБЛИОТЕКИ С ПОСТПРОЦЕССОРОМ

Обмен информацией между основным модулем библиотеки и постпроцессором происходит через временные файлы, которые создаются в папке TEMP_CNC в системном каталоге временных файлов. Во папке TEMP_CNC создается папка текущей сессии, в имени которой присутствует уникальный GUID. Конвертация управляющей программы запускается через команду панели инструментов «Программа ЧПУ». В процессе конвертации программы во временной папке создаются следующие файлы:

- *input.tmp* – временный файл с исходным текстом управляющей программы на промежуточном языке;
- *output.tmp* – временный файл с текстом программы на языке конкретной системы ЧПУ, полученный в результате работы постпроцессора;
- *path.tmp* – временный файл, содержащий описания траекторий движения инструментов внутри станочных циклов.

Схема взаимодействия основного модуля библиотеки с постпроцессором в процессе конвертации управляющей программы показана на рисунке 2.

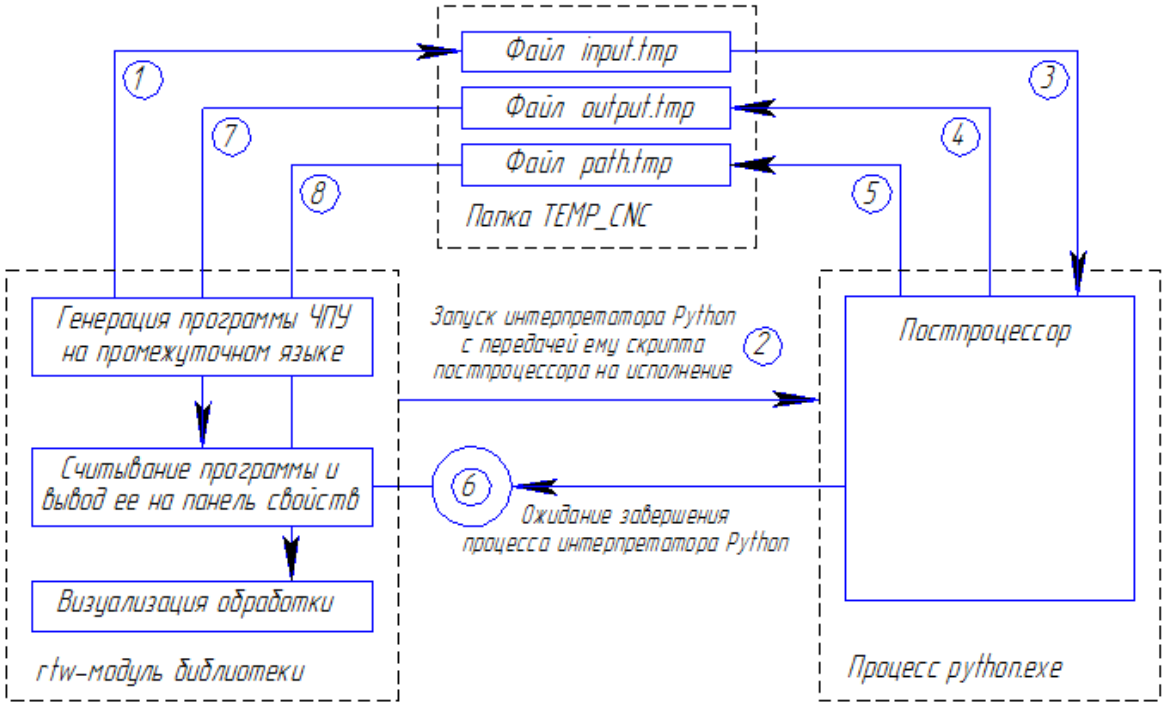

Рисунок 2. Схема взаимодействия Модуля ЧПУ с постпроцессором

На схеме цифрами обозначена следующая последовательность действий:

(1) — Генерация управляющей программы в кодах промежуточного языка и запись этой программы на диск во временный файл *input.tmp*.

(2) — Запуск процесса *python.exe* с помощью функции API Windows CreateProcess. В качестве параметра командной строки передается путь к файлу *input.tmp*.

(3) — Интерпретатор Python начинает исполнение скрипта постпроцессора, который с диска считывается файл input.tmp.

(4) — В процессе исполнения постпроцессора файл *input.tmp* разбирается по кадрам, создаются новые кадры программы, которые записываются на диск в файл *output.tmp*.

(5) — Если исходная программа ЧПУ содержит станочные циклы, описанные на промежуточном языке, и если постпроцессор реализует расчет траектории инструмента для этих циклов, то создается файл *path.tmp*, в который выводится траектория движения инструмента внутри циклов.

(6) — В течение времени исполнения постпроцессора Модуль ЧПУ ожидает завершения процесса *python.exe* с помощью функции WaitForSingleObject(pi.hProcess, 10000), где pi — структура PROCESS_INFORMATION. Максимальное время ожидания составляет 10 секунд. Если за это время процесс не завершился, то выдается ошибка о превышении времени ожидания. Необходимость ожидания завершения процесса вызвана тем, что файл *output.tmp* представляет собой разделяемый ресурс, и к нему должен быть раздельный доступ постпроцессора и модуля rtw библиотеки.

(7) — После завершения процесса библиотека считывает файл *output.tmp* с диска и выводит программу на панель свойств.

(8) — Файл *path.tmp* считывается и распознается библиотекой для получения траектории внутри станочных циклов.

Для передачи сообщений от постпроцессора Модулю ЧПУ используется стандартный буфер *stdout*. При успешной конвертации программы постпроцессор выдает сообщение OK, в случае возникновения ошибок в *stdout* выводятся сообщения об ошибках. Через *stdout* происходит получение как ошибок, которые генерирует постпроцессор, так и ошибок, которые выдает интерпретатор *python.exe*.

3. ПРОМЕЖУТОЧНЫЙ ЯЗЫК УПРАВЛЯЮЩЕЙ ПРОГРАММЫ

Промежуточный язык ЧПУ, в котором первоначально генерируется управляющая программа, основан на стандарте ISO. Язык включает в себя следующие функции.

Вспомогательные функции (M-коды):
M03 – включение вращения шпинделя по часовой стрелке;
M04 – включение вращения шпинделя против часовой стрелки;
M05 – останов вращения шпинделя;
M08 – включить основное охлаждение;
M09 – выключить охлаждение;
M30 – конец программы со сбросом модальных функций.

Подготовительные функции (G-коды):
G00 – Быстрое позиционирование;
G01 – Линейная интерполяция;
G02 – Круговая интерполяция по часовой стрелке;
G03 – Круговая интерполяция против часовой стрелки;
G04 – Пауза;
G08 – Разгон;
G09 – Торможение;
G40 – Отмена всех коррекций инструмента;
G41 – Коррекция на радиус левая;
G42 – Коррекция на радиус правая;
G66 – Синхронизация с нулем шпинделя (в ISO отсутствует);
G80 – Отмена постоянного цикла;
G96 – Постоянная скорость резания;
G97 – Отмена постоянной скорости резания;
G90 – Абсолютный размер.

Прочие функции:
S – функция главного движения;
F – функция подачи;

T – функция инструмента.

Кроме этих функций промежуточный язык содержит дополнительные возможности для включения в управляющую программу станочных циклов.

4. РЕАЛИЗАЦИЯ ПОДДЕРЖКИ СТАНОЧНЫХ ЦИКЛОВ

Включение в управляющую программу станочных циклов позволяет получать весьма короткие и эффективные программы. В этом случае программирование и контроль частично переносятся на уровень станка.

4.1. Описание станочных циклов в постпроцессоре

Чтобы реализовать поддержку станочных циклов, для этого в начале постпроцессора в комментарии должен присутствовать блок описания машинных циклов.

Формат блока описания машинных циклов рассмотрим на примере двух циклов G72 и G82 системы ЧПУ Маяк 600Т:

```
BEGIN_MACHINE_CYCLES

    BEGIN_CYCLE 72
    NAME    "Цикл продольного точения"
    COMMENT "(вдоль контура)"
    TYPE Turning
    PARAMETERS
    X   A   "X конечной точки профиля"
    Z   A   "Z конечной точки профиля"
    I   S   "глубина врезания за проход"
    J   S   "количество проходов"
    B   S   "припуск на чистовой проход по X"
    VP  S   "количество чистовых проходов"
    C   S   "главный угол резца в плане"
    K   A   "конусность"
    F   S   "скорость подачи"
    P   O   "параметр перемещения по X"
                begin_list
                    "с подачей F"
                    "с максимальной скоростью (G0)"
                end_list
    TOOL_DEMANDS "Chisel"
    PROFIL_DEMANDS "TrapecG1 | TrapecG2 | TrapecG3 | TrapecG4"
    END_CYCLE

    BEGIN_CYCLE 82
    NAME    "Цикл подрезки торцов"
    COMMENT "Цикл подрезки торцов"
    TYPE Turning
    PARAMETERS
    X   A   "X конечной точки профиля"
    Z   A   "Z конечной точки профиля"
    I   S   "глубина врезания за проход"
    J   S   "количество проходов"
    B   S   "припуск на чистовой проход по Z"
    VP  S   "количество чистовых проходов"
    C   S   "главный угол резца в плане"
    K   A   "конусность"
    F   S   "скорость подачи"
    P   O   "параметр перемещения по Z"
                begin_list
```

```
                    "с подачей F"
                    "с максимальной скоростью (G0)"
                end_list
        TOOL_DEMANDS "Chisel"
        PROFIL_DEMANDS "TrapecV1 | TrapecV2 | TrapecV3 | TrapecV4"
        END_CYCLE
END_MACHINE_CYCLES
```

Блок описания станочных циклов должен находиться внутри ключевых слов BEGIN_MACHINE_CYCLES (начало блока) и END_MACHINE_CYCLES (конец блока).

Для описания станочного цикла используются следующие ключевые слова:
BEGIN_CYCLE – начало описания цикла;
END_CYCLE – конец описания цикла;
NAME – имя цикла (в кавычках);
COMMENT – комментарий к циклу (в кавычках);
TYPE – тип цикла, может принимать следующие значения:
Turning – точение,
Drilling – сверление,
Threading – нарезание резьбы (резцом, плашкой или метчиком);
PARAMETERS – список параметров.
TOOL_DEMANDS – требования к инструменту (в кавычках), может иметь значения:
Chisel – резец,
Drill – сверло,
Tap – метчик или плашка;
PROFIL_DEMANDS – требования к описанию профиля обработки (фигура цикла).

За ключевым словом PARAMETERS идет список строк с описаниями параметров цикла.

Список параметров разработчик постпроцессора должен сформировать на основе паспорта станка. В строке описания параметра сначала идет обозначение параметра, потом тип параметра (A, O, S, D, OD), название параметра (в кавычках) и значение параметра по умолчанию (может отсутствовать). Если значения параметра по умолчанию нет, или оно равно 0, то оно просто не указывается.

Типы параметров:

A – автоматический. Параметр должна рассчитать или библиотека и вернуть его в списке переменных внешней среды (см. ниже), или его рассчитывает сам постпроцессор.

O – обязательный. Пользователь должен обязательно ввести значение для этого параметра.

S – свободный. Значение параметра можно не задавать.

D – дополнительный. Параметр не входит в список параметров цикла по паспорту, но отображается пользователю. Через него разработчик постпроцессора может получить от пользователя дополнительную информацию.

OD – обязательный дополнительный.

Между ключевыми словами begin_list и end_list задается список строк, которые выступают в качестве значения данного параметра цикла.

Переменными внешней среды управляющей программы являются:
Для токарных циклов (у которых TYPE равен Turning):
_L1 – длина подвода;
_L2 – длина перебега.

Для сверлильных циклов (Drilling):
_Depth – глубина отверстия;
_D – диаметр отверстия;

_L1 – длина подвода;
_L2 – длина перебега (для сквозных отверстий);
_Angle – угол при вершине сверла.

Для резьбонарезных циклов (Threading):
_FEED – ход резьбы (подача);
_DEPTH – глубина профиля резьбы;
_ALFA – угол профиля резьбы;
_DIRECT – направление витков резьбы (правая или левая);
_L1 – длина подвода;
_L2 – длина перебега (для сквозных отверстий).

4.2. Инициализация пользователем параметров станочного цикла

В момент подключения библиотеки происходит считывание из постпроцессора блока машинных циклов. В дальнейшем эта инсформация, описывающая станочные циклы, используется в процессе задания параметров обработок. На вкладке «Стратегия» панели свойств для большинства обработок (кроме Контура и Канавки) присутствует список «Схема» со значениями: «В элементарных перемещениях» и «В машинных циклах» (рисунок 3).

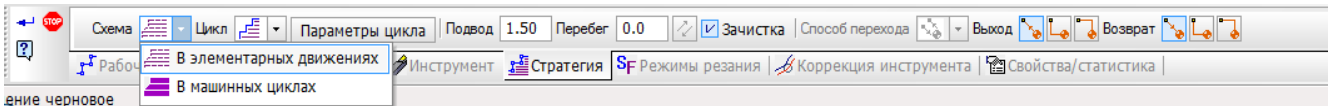

Рисунок 3. Выбор схемы обработки

При выборе схемы «В машинных циклах» активизируется список со станочными циклами, которые поддерживает постпроцессор и которые соответствуют данному виду обработки (рисунок 4). Соответствие между станочным циклом и видом обработки определяется по значению ключевого слова TYPE.

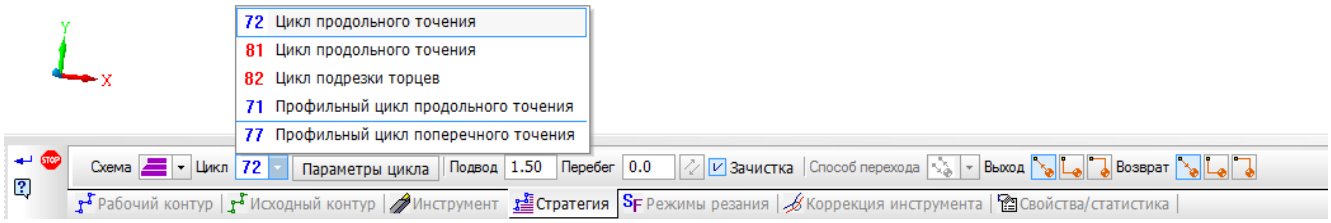

Рисунок 4. Список станочных циклов

Рядом со списком станочных циклов расположена кнопка «Параметры цикла». При нажатии на эту кнопку выводится диалог задания параметров выбранного цикла (рисунок 5).

Вся информация, которая выводится на диалоге, берется из блока описания машинных циклов постпроцессора (название цикла, комментарий, список параметров). Таблица параметров на диалоге настраивается автоматически в соответствии с описанием параметров цикла после ключевого слова PARAMETERS в блоке машинных циклов постпроцессора.

В таблице параметров выводятся обозначение параметра, его название и поле ввода числового значения или выпадающий список значений. Для дополнительных параметров с типами D или OD обозначение параметра не показываются. Параметры, имеющие тип A отображаются на диалоге как недоступные для редактирования с надписью auto. Для

параметров цикла, значения которых заданы между ключевыми словами begin_list и end_list, отображается выпадающий список значений.

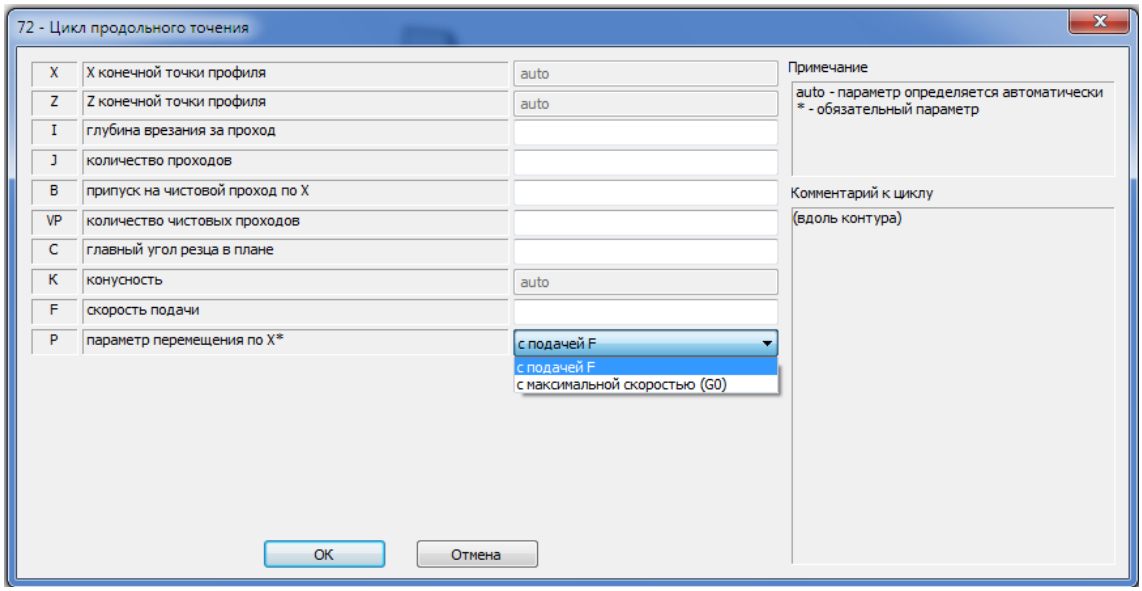

Рисунок 5. Диалог для инициализации параметров цикла

Все обязательные параметры цикла должны быть заданы в диалоге. В противном случае выдается предупреждение, что постпроцессор может сработать некорректно или выдать ошибку.

4.3. Фигуры станочных циклов

Ключевое слово PROFIL_DEMANDS в блоке описания станочного цикла должно содержать описание геометрии, для обработки которой предназначен цикл, другими словами, описание фигуры цикла.

Во многих системах ЧПУ присутствуют циклы многопроходных черновых обработок, у которых фигура обрабатываемой области представляет собой трапецию. В библиотеке реализована возможность распознавания внутри области удаления припуска таких трапециевидных токарных циклов. Для этого в библиотеке предусмотрено две фигуры в виде трапеций: с горизонтальным расположением трапеции TrapecGn и вертикальным расположением TrapecVn. Номер n в названии формы показывает ориентацию трапеции:

TrapecG1, TrapecV1 – снаружи справа;

TrapecG2, TrapecV2 – снаружи слева;

TrapecG3, TrapecV3 – изнутри справа;

TrapecG4, TrapecV4 – изнутри слева.

Если цикл позволяет удалять припуск для разных форм трапеций, то после ключевого слова PROFIL_DEMANDS можно комбинировать эти формы. Например:

PROFIL_DEMANDS "TrapecV1 | TrapecV2 | TrapecV3 | TrapecV4".

Символ «вертикальная черта» здесь означает логическое сложение (оператор «или»).

На рисунке 6 показаны формы многопроходных циклов в виде трапеций. Точка *begin* означает начальную точку цикла, точка *end* – противоположную по диагонали точку габаритного прямоугольника, очерченного вокруг профиля цикла.

Для трапециевидных циклов библиотека автоматически разбивает область припуска на фигуры трапеций в соответствии с требованиями PROFIL_DEMANDS. В случае невозможности такой разбивки выдается сообщение о невозможности применения станочного цикла к указанной геометрии.

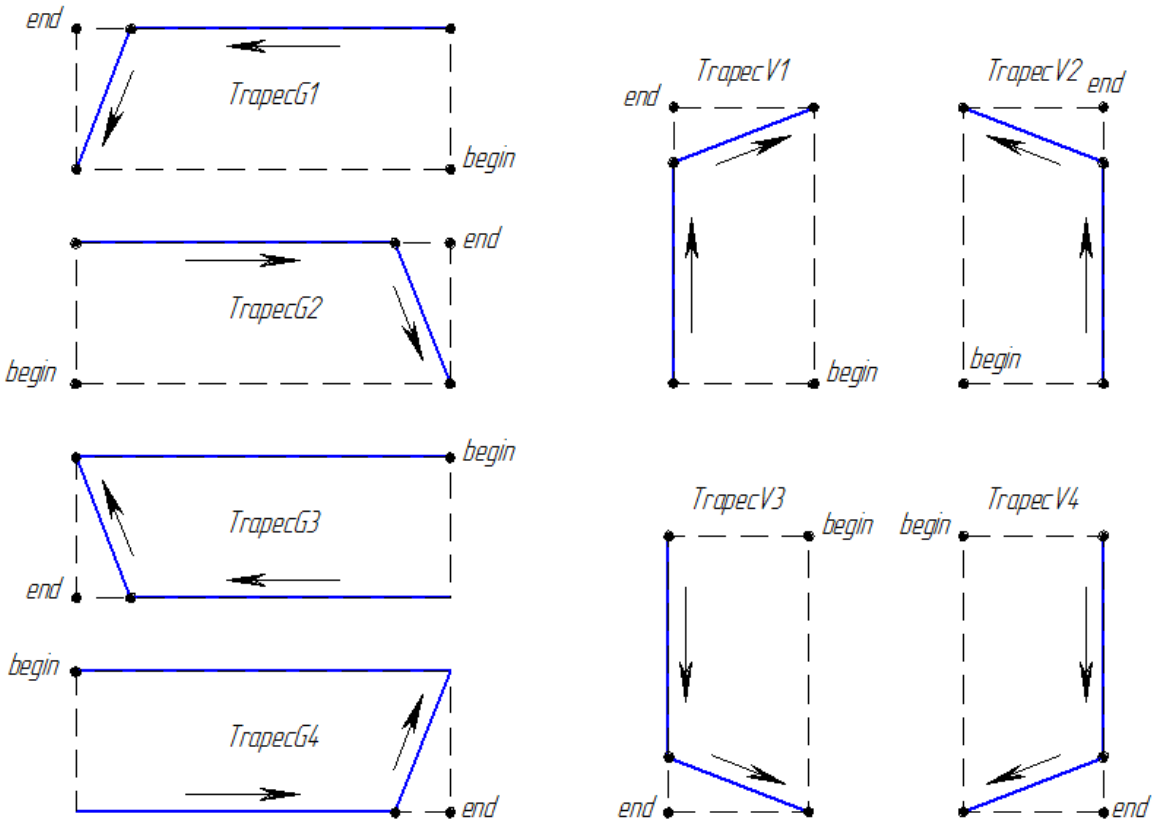

Рисунок 6. Профили станочных циклов в форме трапеций

Для других форм циклов(например, канавки или какие-то более сложные профили) автоматическое распознавание фигуры цикла в базовой версии библиотеки не реализовано. Для таких циклов ключевое слово PROFIL_DEMANDS может не содержать описания формы цикла, например, может быть такая запись PROFIL_DEMANDS ” ”. В этом случае библиотека не будет выполнять распознавания фигур цикла в области припуска, а будет передавать в постпроцессор полносью весь указанный пользователем контур в качестве профиля для обработки циклом. При этом постпроцессор может содержать проверку переданного контура на пригодность его обработки данным циклом.

На рисунке 7 показана автоматическая разбивка области припуска в результате распознавания фигур трапециевидных циклов. На рисунке также показаны области удаляемого припуска для сверлильных циклов.

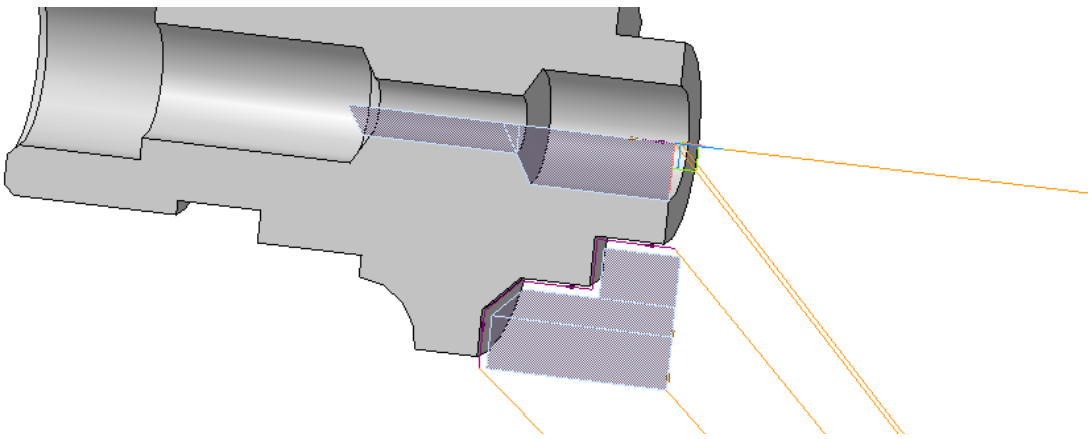

Рисунок 7. Разбивка области припуска на фигуры циклов

Для сверлильных циклов фигуры не предусмотрены, и для них ключевое слово PROFIL_DEMANDS можно вообще не задавать. Начальная и конечная точка сверления определяется по крайним точкам контура обработки.

Для резьбовых циклов предусмотрено три формы:
Cylinder – цилиндрическая резьба;
Cone – коническая резьба;
Spiral – торцовая резьба (спираль).
Эти формы также могут комбинироваться.
Если цикл позволяет нарезать как цилиндрическую, так и коническую резьбу, то запись будет иметь вид:

```
PROFIL_DEMANDS "Cylinder | Cone"
```

Схема реализации поддержки станочных циклов показана на рисунке 8.

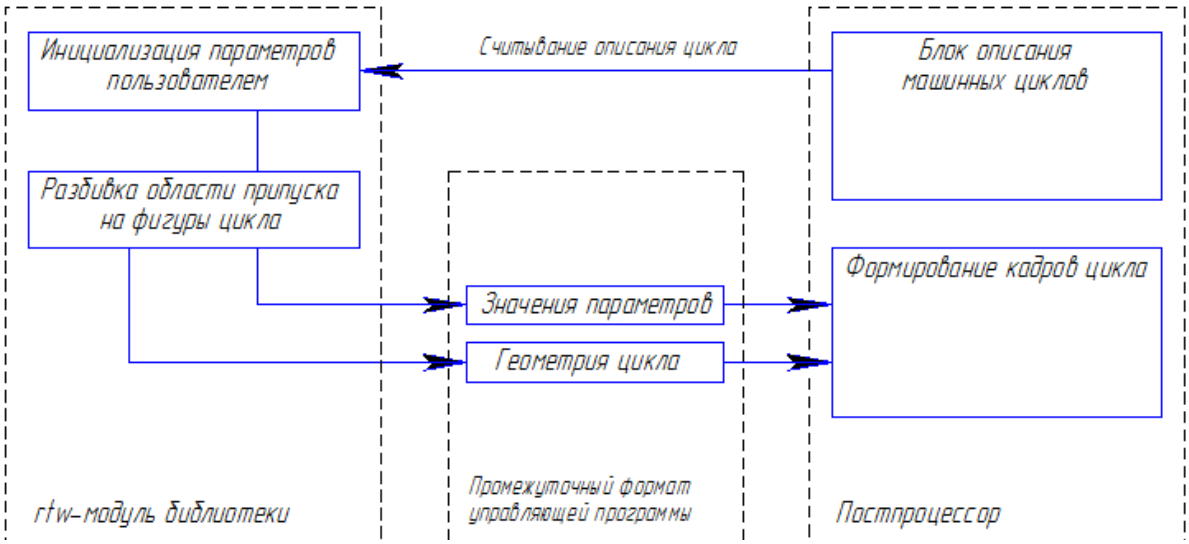

Рисунок 8. Схема поддержки станочных циклов

В составе управляющей программы в промежуточном формате библиотека формирует запись станочного цикла. Эта запись включает в себя значения параметров цикла, значения переменных внешней среды и геометрию, которую нужно обработать циклом. Для многопроходных токарных циклов геометрия цикла представляет собой описание контура в элементарных перемещениях. Описание этого контура следует за параметрами цикла в отдельных кадрах, которые завершаются командой G80 («отмена цикла»). Для сверлильных циклов и циклов нарезания резьбы плашкой/метчиком геометрия цикла описывается двумя параметрами:_Zbegin – начальная точка сверления на рабочей подаче с учетом длины подвода, _Zend — конечная точка сверления с учетом угла при вершине сверла.

Рассмотрим примеры записи некоторых циклов в промежуточном формате управляющей программы. Пример описания цикла G72 (система ЧПУ Маяк 600Т):

```
N006  S500 F1.2                  задание режимов резания
N007  G00 X156.31 Z4.31          выход в начальную точку цикла
N008  CYCLE 72  I=5 B=0.5 VP=1 C=95 P=5  _L1=5 _L2=5  _Xbegin=118 _Zbegin=4.31
_Xend=156.31 _Zend=-136.98           параметры цикла
N009  Z-136.98
N010  X156.31                    описание профиля цикла
N011  G80                        отмена постоянного цикла
N012  G00 X156.31 Z4.31          возврат в начальную точку цикла
```

Все параметры цикла передаются в одном кадре: сначала основные параметры, потом дополнительные, затем переменные внешней среды, начальная точка цикла (_Xbegin, _Zbegin) и конечная точка контура (_Xend, _Zend).

На основе данной информации постпроцессор после конвертации программы выдаст описание цикла на языке системы ЧПУ Маяк 600Т:

```
N006  S500 F1.2
N007  G00 X156.31 Z4.31
N008  G72 X156.31 Z-136.98 I5 B0.5 VP1 C95 K0 P=5
N009  G00 X156.31 Z4.31
```

Пример записи для токарного цикла G77 системы ЧПУ НЦ-31:

```
N007  G00  X68  Z2.5          выход в точку начала цикла
N008  CYCLE 77  P1=2.5 W=1 _L1=1.5 _L2=1.5 _Xbegin=53.24 _Zbegin=2.5 _Xend=68 _Zend=-23.5
N009  G01  Z-23.5
N010  G01  X68                описание профиля обработки
N011  G80
N012  G00  X68  Z2.5          возврат в точку начала цикла
```

Пример записи для сверлильного цикла G73 системы ЧПУ НЦ-31:

```
N049  G00  X0  Z-22.40        выход в точку начала цикла
N050  CYCLE 73  P=10 _D=8  _L1=0 _L2=1 _Angle=118 _Zbegin=-22.40 _Zend=-47.29
N051  G00  X0  Z-22.40        возврат в точку начала цикла
```

Пример записи для резьбонарезного цикла G31 системы ЧПУ НЦ-31:

```
N144  G00  X36  Z-0.5         выход в точку начала цикла
N145  CYCLE 31  P2=0.5 _FEED=2.  _NTHREADS=1 _DEPTH=1.62 _L1=6 _L2=5.5 _Xbegin=32.75
_Zbegin=-0.5  _Xend=32.75  _Zend=-35.5
N146  G00  X36  Z-0.5         возврат в точку начала цикла
```

Пример записи для цикла нарезания резьбы плашкой/метчиком G33 системы ЧПУ НЦ-31:

```
N164  G00  X0 Z-3.5           выход в точку начала цикла
N165  CYCLE 33  S=1 _FEED=1.5 _NTHREADS=1 _L1=3 _L2=3 _Zbegin=-3.5 _Zend=-27
N166  M03
```

4.4. Вопросы безопасного использования станочных циклов

В процессе конвертации станочного цикла постпроцессор кроме формирования конечной записи цикла на языке системы ЧПУ может дополнительно выполнять контроль входной информации, а также генерировать траекторию внутри цикла.

Контроль входного описания цикла может включать в себя:
- проверку значений параметров, введенных пользователем;
- проверку пригодности контура для обработки циклом.

Эти действия не являются обязательными для реализации внутри постпроцессора.

Если же в постпроцессоре отсутствуют какие-либо функции контроля или генерации траектории, это не лишает пользователя возможности включения станочного цикла в управляющую программу. Но в этом случае пользователь должен внимательно подходить к использованию такого станочного цикла, должен понимать смысл всех параметров цикла, а также представлять, как будет происходить обработка выбранного контура внутри станочного цикла. Даже если при этом пользователь введет какие-либо некорректные значения параметров или укажет профиль, непригодный для обработки данным циклом, есть некоторая вероятность,

что такую программу заблокирует стойка управления после передачи программы на станок. Но в любом случае необходимо выполнять обязательную предварительную проверку программы на станке при соблюдении всех мер предосторожности и техники безопасности.

Если постпроцессор формирует траекторию внутри цикла, то это дает возможность визуализировать перемещения инструмента, что дополнительно повышает безопасность применения станочного цикла.

Необходимым условием корректной визуализации является требование непрерывности траектории. Траектория токарной обработки, которую генерирует библиотека, состоит из совокупности отрезков (кадры линейной интерполяции G01), дуг (круговая интерполяция G02 или G03) и точек (остальные функции ЧПУ). Начальная точка текущего кадра должна обязательно совпадать с конечной точкой предыдущего кадра. Это нужно для того, чтобы не было неопределенности, откуда приходит инструмент в текущую точку. Данное требование относится и к траекториям станочных циклов.

Таким образом, главным условием безопасного включения в управляющую программу станочных циклов является требование, чтобы конечная точка траектории внутри цикла совпадала с начальной точкой цикла. Сама траектория внутри цикла представляет собой «черный ящик». После отработки цикла инструмент должен обязательно вернуться в точку начала цикла.

Чтобы выполнить данное условие, библиотека после кадра (или нескольких кадров) станочного цикла вставляет кадр возврата в точку, которая была перед циклом. Если не делать принудительного возврата в начало цикла, то может возникнуть ситуация, когда конечная точка цикла будет неопределенной, что может привести к непредсказуемым столкновениям на станке.

Дополнительный кадр вставляется после всех станочных циклов, кроме нарезания резьбы плашкой/метчиком. Предполагается, что стойка управления станком должна в любом случае выполнить свинчивание плашки/метчика.

Кадр, который вставляет библиотека после станочного цикла, представляет собой линейную интерполяцию на ускоренной подаче G00, т. е. отрезок. Если постпроцессор траекторию внутри цикла не генерирует, то этот отрезок вырождается в точку. Если постпроцессор генерирует траекторию внутри цикла, то начальную точку этого отрезка библиотека определит по координатам конечной точки траектории внутри цикла, замкнув таким образом «внешнюю» траекторию с внутренней траекторией станочного цикла. Если же траектория внутри цикла такова, что ее конечная точка совпадает с началом цикла, то добавочный кадр после цикла становится лишним. В этом случае при разработке постпроцессора можно предусмотреть удаление этого лишнего кадра из программы.