Использование языка высокого уровня для написания кода безопасного запуска параметрических программ и создания циклов нарезания резьбы[1]

Г.М. Мартинов,
д.т.н., проф., martinov@ncsystems.ru,
А.Э. Эрднеев,
магистр. 1-го курса, and_erd@mail.ru,
МГТУ «СТАНКИН», г. Москва

В статье представлена программа тестирования для системы ЧПУ «АксиОМА Контрол» и разработанная методика тестирования. Рассмотрены практические аспекты разработки станочных циклов в системе ЧПУ «АксиОМА Контрол» с применением языка ISO 7-bit, а также языка высокого уровня, в частности, реализующих операции нарезания резьбы метчиком. Представлены разработанные программа и подпрограмма, реализующие нарезание резьбы.

The article presents the testing program for the control system «AxiOMA Control» and the developed testing method. Practical aspects of the development of machine cycles in the CNC system «AxiOMA Control» with the use of ISO 7-bit, as well as high-level language, in particular, realizing thread tapping operations, are considered. The developed program and subprogram implementing the threading are presented.

Тестирование программного обеспечения - это проверка соответствия между реальной и ожидаемой отработкой программы, осуществляемая с применением конечного набора тестов, выбранного определенным образом.

Тестирование может быть разных видов: функциональное, системное, модульное и т.д.

В данной работе рассматривается регрессионное тестирование, показывающее степень влияния новых функций, улучшений и исправленных дефектов на существующую функциональность продукта, а также проверяющее наличие старых дефектов.

Такой способ тестирования предусматривает минимальное участие человека, так как автоматизация производственных процессов становится все более актуальной задачей. Соответственно, для большей производительности необходимо использовать многоканальную систему для возможности запуска программ на разных каналах [1]. Также иногда возникает необходимость одновременной работы нескольких каналов системы управления. Каналы должны работать не просто независимо от операций, а операции могут быть синхронизированы у каналов. Для этого необходимо, чтобы на одном канале программа была синхронизирована с другой. Чтобы это проверить, необходимо обеспечить возможность запуска программы с одного канала на другой. При выполнении программа возвращает какой-либо флаг обратно на первый канал, таким образом, два канала обмениваются информацией. Раньше для тестирования нескольких каналов запускали программу поочередно на разных каналах, что занимало много времени, так как необходимо было следить за окончанием выполнения программы на одном канале, после чего запускать на другом.

При использовании нового метода нужно лишь запустить УП на одном канале и программы будут выполняться на остальных автоматически. От тестировщика требуется лишь изредка смотреть за состоянием проверки.

Программа для тестирования была написана с применением языка высокого уровня для отечественной системы ЧПУ «АксиОМА Контрол». Она позволяет синхронизировать каналы и обеспечить передачу данных между ними [2].

Далее представлено краткое описание программы для тестирования. Сначала указывается канал, на котором будет проводиться тестирование, количество программ и их названия.

```
int nCh = 1; // выбор рабочего канала
int nPartProgNo = 2; // количество программ
string chanProgs[nPartProgNo ]; // содержит имена запускаемых УП
chanProgs[0] = "01.nc";
chanProgs[1] = "02.nc";
```

Далее проводится проверка работы канала. Программа работает только в режиме «Авто», в противном случае терминал выдаст ошибку, используя команду «terminate». При правильно выбранном канале выводится сообщение, информирующее о правильности работы.

```
if(@CH[0].MODE != 1)
{
    terminate(5554, "Kanal ne v rejime Auto"); //сообщение о некорректно выбранном режиме
}
MSG("Auto mode");
```

После проверки режима проводится проверка состояния канала. При состоянии канала «готов», выведется сообщение, информирующее о правильности работы, и отработка программы продолжится, в противном случае выведется сообщение об ошибке.

```
if(@CH[0].STATE != 2)
{
```

[1] Работа выполнена при поддержке Минобрнауки России в рамках выполнения государственного задания (№ 2.1237.2017/ПЧ)

```
        terminate(5555, "Nevernoe sostoyanie kanala"); //некорректное состояние канала.
}
MSG("Kanal gotov");
```

На данном этапе проверяется правильность загрузки и запуска каждой программы с применением цикла «for». В случае, если программа не будет загружена (запущена) с первого раза, то будет выделено время на повторную попытку её загрузки (запуска). Если же программа не запустится (загрузится) со второго раза - терминал выдаст ошибку при помощи команды «terminate».

```
for(Iterator = 0; Iterator < nPartProgNo; Iterator++)
{
        if(channel_load(nCh, chanProgs[Iterator]) == false) //не удалось загрузить с первого раза
        {
                channel_reset(nCh); //reset канала
                sleep(10 *msSleep);
                if(channel_load(nCh, chanProgs[Iterator]) == false) //ошибка, повторная загрузка не удалась
                {
                        terminate(5556, "Ne udalos zagruzit kanal");
                }
                channel_wait(nCh);
        }
        MSG("Kanal zagrujen");

        if(channel_run(nCh, chanProgs[Iterator]) == false) //не удалось запустить с первого раза
        {
                channel_reset(nCh); //reset канала
                sleep(10 *msSleep);
                if(channel_run(nCh, chanProgs[Iterator]) == false) //ошибка, повторный запуск не удался
                {
                        terminate(5558, "Ne udalos zapustit");
                }
                channel_wait(nCh);
        }
        MSG("Kanal zapuwen");
}
```

Программа будет выполняться до тех пор, пока цикл «for» не “прогонит” все программы.

Методика тестирования

1. Запустить терминал «АксиОМА Контрол».
2. Запустить программу тестирования на втором канале.
3. Выбрать программу тестирования.
4. Запустить управляющую программу, если интерпретатор во время запуска и при выполнении программы не выводил сообщение об ошибке, то программа отработана верно [3].

Примерами сообщений, выводимых интерпретатором, являются:
- синтаксическая ошибка в управляющей программе;
- ошибка при верификации движения на выходе интерполятора, нулевая подача в команде движения.

рис. 1 Выполнение программы

Представленная методика тестирования показывает правильность написания разрабатываемого цикла.

Цикл нарезания резьбы

В системе ЧПУ «АксиОМА Контрол», на G-коде, с применением языка высокого уровня была создана специальная функция, позволяющая выполнять нарезание резьбы метчиком, как однопроходное, так и многопроходное. Разница между двумя циклами заключается в количестве проходов, то есть один и несколько проходов за один цикл соответственно [4].

Нарезание резьбы метчиками является наиболее распространённым способом ее изготовления. Для образования резьбы необходимо придать метчику два движения: главное вращательное и поступательное с подачей, равной шагу резьбы за каждый оборот метчика.

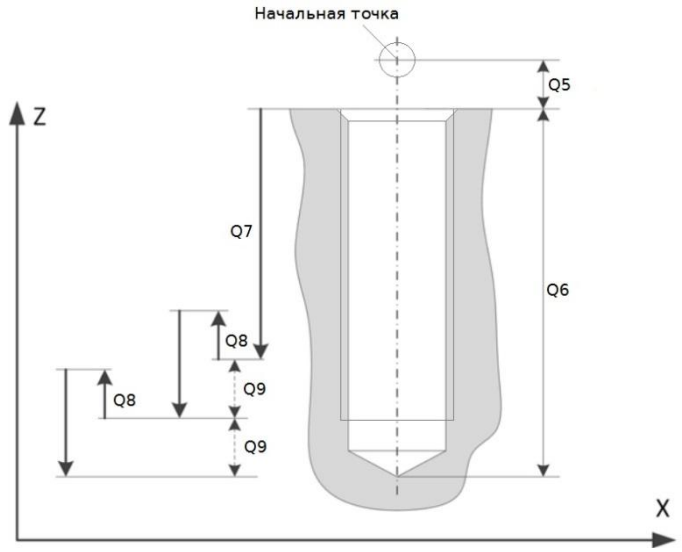

рис. 2 Многопроходное нарезание резьбы метчиком

Программа

```
#use "metch_cycles.c"
G90
Tapping("Z", "C", 200, 0, 50, 20, 60, 20, 60, 20, 50, 25, 0, 200);
$dPerm[1] = @ITPPATH;
$dPerm[3] = fabs(@ITPPATH - $dPerm[2]);
if ($dPerm[3] > 0.1)
{
      terminate (5559, "dPath i ITPPATH ne ravni!")
}
M02
```

Таким образом, при отработке представленной выше УП проводится многопроходное нарезание резьбы глубиной 200 мм, первый и последующий проходы равны 60 мм. Помимо этого ведётся проверка путём сравнивания переменных dPath и @ITPPATH. dPath рассчитывает пройденный путь внутри программы, @ITPPATH является переменной станка. При их несоответствии будет вызвана ошибка.

Функция имеет 14 основных параметров, при помощи которых можно задать основные характеристики требуемой резьбы:

1. strToolAx (string) – имя оси инструмента;
2. strSpindleAx (string) – имя круговой оси;
3. dRetPlane (double) – плоскость отвода вдоль оси инструмента (абсолютная система координат);
4. dRefPlane (double) – плоскость поверхности начала обработки вдоль оси инструмента (абсолютная система координат);
5. dSafeDist (double) – безопасное расстояние до начала резьбы (мм);
6. dDepth (double) – длина резьбы (мм);
7. dStep1 (double) – глубина резания при первом проходе (мм);
8. dStepBack (double) – величина отхода метчика перед очередным рабочим проходом, при приравнивании параметра к нулю проводится отход к точке dExitStep (мм);
9. dStep2 (double) – глубина резания при последующих проходах (мм);
10. dPitch (double) – шаг резьбы (мм);
11. dTapFeed (double) – подача при врезании (мм/мин);
12. dRetFeed (double) – подача при выводе метчика (мм/мин);
13. dExitStep (double) – точка, на которую будет выходить метчик перед очередным рабочим проходом при dStepBack = 0;
14. dZ (double) – переменная, необходимая для расчёта пройденного пути dPath.

Подпрограмма

Помимо основной программы была разработана подпрограмма цикла нарезания резьбы метчиком для системы ЧПУ «АксиОМА Контрол».

Реализация первого прохода нарезания резьбы представляет собой независимый блок, в котором проводятся операции подвода инструмента к обрабатываемой поверхности на расстояние dSafeDist, нарезание резьбы на глубину dStep1 при вычисленных параметрах dSpindlePath и dTapFeed, поднятие инструмента на расстояние dStepBack при отрицательном значении dSpindlePath и dRetFeed.

```
G90
G0 AXSTR(strToolAx)=(dRefPlane + dSafeDist)
G91
G1 AXSTR(strToolAx)=-(dStep1 + dSafeDist) AXSTR(strSpindleAx)= dSpindlePath F=dTapFeed
G1 AXSTR(strToolAx)=(dStepBack) AXSTR(strSpindleAx)=- dSpindlePath F=dRetFeed
```

Реализация последующих проходов представляет собой циклически повторяющиеся действия, в которых проводится расчёт оставшихся проходов в зависимости от заданной глубины dDepth и проходов dStep1 и dStep2, вводится переменная dOst, в которой записывается оставшееся расстояние после первого прохода, создаётся цикл for для регулирования числа проходов, которое необходимо сделать станку, после чего выполняется нарезание на глубину dStep2, в dOst записывается новое значение. Далее проверяется, имеет ли остаток меньшее значение, чем шаг dStep2. В случае если нет, то инструмент поднимается на dStepBack, после чего цикл повторяется, а если да - происходит проверка равенства остатка нулю. В первом случае инструмент достиг глубины dDepth и поднимается на безопасное расстояние над отверстием на dSafeDist, во втором случае, он поднимается на dStepBack, опускается на величину dOst, после чего поднимается на безопасное расстояние.

```
double dSch = (dDepth - dStep1) / dStep2; //количество оставшихся проходов
double dOst = dDepth - dStep1; //оставшееся расстояние
for (int iSc = 1; iSc <= dSch; iSc++) //определяет, сколько шагов надо сделать
{
    G1 AXSTR(strToolAx)=-(dStep2 + dStepBack) AXSTR(strSpindleAx)=dSpindlePath F=dTapFeed
    dOst = dOst - dStep2;
    if ((dSch - iSc) < 1) //есть шаг, который меньше шага 2
    {
        if (dOst == 0) //если остаток равен 0
        {
            G1 AXSTR(strToolAx)=(dDepth + dSafeDist) AXSTR(strSpindleAx)=-dSpindlePath F=dRetFeed
            dPath = dPath + dDepth + dSafeDist;
        }
        else //если остаток не равен 0
        {
            G1 AXSTR(strToolAx)=(dStepBack) AXSTR(strSpindleAx)=-dSpindlePath F=dRetFeed
            G1 AXSTR(strToolAx)=-(dOst + dStepBack) AXSTR(strSpindleAx)=dSpindlePath F=dTapFeed
            G1 AXSTR(strToolAx)=(dDepth + dSafeDist) AXSTR(strSpindleAx)=-dSpindlePath F=dRetFeed
            dPath = dPath + dDepth + dSafeDist;
        }
    }
    else //если нет шага, который меньше шага 2
    {
        G1 AXSTR(strToolAx)=(dStepBack) AXSTR(strSpindleAx)=-dSpindlePath F=dRetFeed
    }
}
```

Однопроходной цикл предусмотрен для случая, если глубина резания при первом проходе dStep1 равна длине резьбы dDepth. Он является более простым в реализации, так как выполняется нарезание на глубину dStep1, которая равна глубине dDepth, после чего инструмент выводится из отверстия и отводится на безопасное расстояние dSafeDist.

```
if (dDepth == dStep1) //если глубина равна шагу один
{
    dStepBack = dStep1 + dSafeDist;
    G90
    G0 AXSTR(strToolAx)=(dRefPlane + dSafeDist)
    G91
    G1 AXSTR(strToolAx)=-(dStep1 + dSafeDist) AXSTR(strSpindleAx) =dSpindlePath F=dTapFeed
    G1 AXSTR(strToolAx)=(dStepBack) AXSTR(strSpindleAx)=-dSpindlePath F=dRetFeed
    G90
    G0 AXSTR(strToolAx)=dRetPlane
}
```

Синхронизация подачи и вращения шпинделя

Важным требованием в работе цикла нарезания резьбы метчиком является синхронизация подачи и вращения шпинделя, т.е. выполнения условия, чтобы за пройденное расстояние шпиндель делал определённое число оборотов. При несоблюдении данной синхронизации произойдет срыв резьбы или аварийная ситуация и брак в изделии, что может привести к большим экономическим потерям, в особенности если подобных операций на одном

изделии выполняется большое количество [5]. Для этого производится расчет числа оборотов шпинделя dNumRots, после чего вычисляется расстояние, которое он пройдёт.

double dNumRots = (dDepth + dSafeDist) / dPitch;
*double dSpindlePath = dNumRots * 360.0;*

Заключение

Были рассмотрены базовые циклы для различных видов обработки, реализуемые в системах ЧПУ. Рассмотрены основные принципы создания циклов на языке G-кода с применением языка высокого уровня в системе ЧПУ «АксиОМА Контрол». Разработан и представлен цикл нарезания резьбы. Преимуществами разработанного цикла является рассмотрение различных вариантов задания параметров и выполнение задачи в соответствии с требуемыми технологиями обработки.

Литература

1. Сосонкин В. Л., Мартинов Г.М. Системы числового программного управления // Учебное пособие.М. : Логос, 2005. – 296 с.
2. Соломенцев Ю.М., Сосонкин В.Л., Мартинов Г.М. Построение персональных систем ЧПУ (PCNC) по принципу открытых систем. - Информационные технологии и вычислительные системы. 1997, #3, с. 68-75.
3. Рыбников С.В. Руководство программиста по созданию управляющих программ. Руководство по программированию. МГТУ "СТАНКИН"
4. Мартинов Г.М. Язык высокого уровня для создания параметрических управляющих программ // Техническое описание. -М.: МТГУ "СТАНКИН", 2012.
5. Аверьянов О.И., Клепиков В.В. Режущий инструмент. Учебное пособие - М.:МГИУ, 2007. - 144с.