СЧПУ серии IntNC PRO

РУКОВОДСТВО ПО ПРОГРАММИРОВАНИЮ

ПРОГРАММЫ ПЛК

ВЕРСИЯ 1.0

www.inelsy.com

Оглавление

Список сокращений

- ДОС – датчик обратной связи;
- ПЛК – программируемый логический контроллер;
- ПО – программное обеспечение;
- ПЭС – программа электроавтоматики станка;
- СЧПУ – система числового программного управления;
- УП – управляющая программа;
- УЧПУ – устройство числового программного управления;
- GNU – General Public License – лицензия на свободное программное обеспечение;
- IDE – Integrated Development Environment – интегрированная среда разработки.

Введение

Настоящее руководство по программированию (далее РП) предназначено для изучения принципов создания программ ПЛК для системы ЧПУ серии **IntNC PRO**.

Системы ЧПУ серии **IntNC PRO** имеют программно реализованный встроенный логический контроллер, программы для которого разрабатываются на основе языка программирования высокого уровня IntLang.

Описание языка программирования **IntLang** приведено в Руководстве по программированию «ЯЗЫК ПРОГРАММИРОВАНИЯ IntLang СЧПУ серии IntNC PRO».

Настоящее РП распространяется на все модификации СЧПУ серии **IntNC PRO**.

Сохраняется право внесения изменений!

1 . Особенности реализации программ ПЛК

1.1 Встроенный логический контроллер

Системы ЧПУ серии **IntNC PRO** имеют встроенный механизм выполнения логических программ управления – программно реализованный встроенный логический контроллер.

Интегрированный в системное программное обеспечение (рис. 1.1) логический контроллер гарантирует:

- одно адресное пространство для выполнения системных задач и программ логического управления;
- синхронизацию между различными задачами УЧПУ;
- выполнение до 4-х программ ПЛК в режиме реального времени;
- выполнение до 32-х программ ПЛК в фоновом режиме.

Рис. 1.1. Функциональная схема системного программного обеспечения

1.2 Язык программ ПЛК

Для создания программ ПЛК используется процедурный язык программирования IntLang, разработанный на основе стандарта ANSI C.

Язык программирования IntLang имеет следующие особенности:

- простую языковую базу;
- минимальное число ключевых слов;
- систему типов;
- области действия имён;
- определяемые пользователем собирательные типы данных – структуры и объединения;
- передачу параметров в функцию по значению;
- препроцессор для определения макросов и включения файлов с исходным кодом;
- математические функции и функции работы с массивами.

1.3 Организация программ ПЛК

Программы ПЛК реализуются в виде текстовых файлов с расширением `cfg` и входят в состав конфигурационных файлов УЧПУ для станка.

Программы ПЛК размещаются в директории пользовательских файлов «`source/platform/имя_проекта`» и их имена включаются в файл «`source/platform/имя_проекта/target.cfg`».

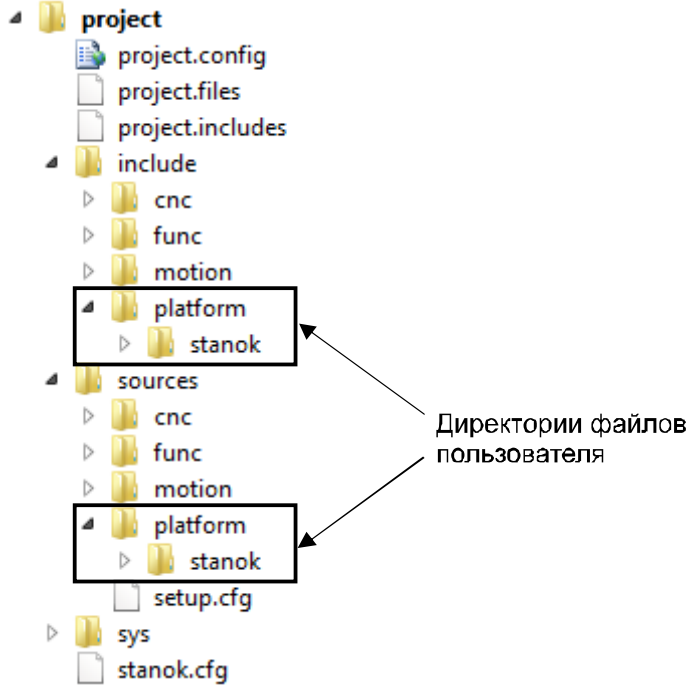

Рис. 1.2. Организация конфигурационных файлов проекта «stanok»

2 . Создание программ ПЛК

2.1 Краткое описание языка программ ПЛК

Краткое описание языка программ ПЛК содержит информацию об основных элементах языка IntLang.

Полное описание языка IntLang приведено в Руководстве по программированию «ЯЗЫК ПРОГРАММИРОВАНИЯ IntLang СЧПУ серии IntNC PRO».

2.1.1 Набор символов

Множество символов языка содержит буквы, цифры и знаки пунктуации.

Набор символов содержит прописные и строчные буквы латинского алфавита, 10 десятичных цифр арабской системы исчисления и символ подчеркивания (_). Они используются для формирования констант, идентификаторов и ключевых слов. Прописные и строчные буквы обрабатываются как разные символы.

Прописные английские буквы:	A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
Строчные английские буквы:	a b c d e f g h i j k l m n o p q r s t u v w x y z
Десятичные цифры:	0 1 2 3 4 5 6 7 8 9
Символ подчеркивания:	_

Знаки пунктуации и специальные символы из набора символов имеют самое разное предназначение, от организации текста программы до определения задач, которые будут выполнены программой.

,	запятая	>	правая угловая скобка
.	точка	!	восклицательный знак
;	точка с запятой	|	вертикальная черта
:	двоеточие	/	знак деления
?	знак вопроса	\	знак обратного деления
’	одинарная цитатная скобка	~	тильда

"	двойная цитатная скобка	+	плюс
(	левая круглая скобка	#	номер
)	правая круглая скобка	%	процент
[	левая прямоугольная скобка	&	амперсанд
]	правая прямоугольная скобка	^	крышечка
{	левая фигурная скобка	*	звездочка
}	правая фигурная скобка	-	минус
<	левая угловая скобка	=	равно

2.1.2 Ключевые слова

Ключевые слова – заранее определенные идентификаторы, которые имеют специальное значение. Их использование строго регламентировано. Имя элемента программы не может совпадать по написанию с ключевым словом.

break	double	int	switch
case	else	long	typedef
char	enum	return	union
const	extern	short	unsigned
continue	float	signed	void
default	for	static	while
do	if	struct	

2.1.3 Базовые типы данных

В языке реализован набор типов данных, называемых базовыми типами. Спецификации этих типов перечислены ниже.

Целые типы:	char, short, int, long, enum
Типы с плавающей точкой:	float, double
Прочие:	void, const

2.1.4 Области значений

Область значений – интервал от минимального до максимального значения, которое может быть представлено в переменной данного типа. В табл. 2.1 приведен размер занимаемой памяти и области значений переменных каждого типа.

Таблица 2.1. Область значений типов

Тип	Размер, байт	Область значений
unsigned char	1	$0 \div 255$
signed char (char)	1	$-128 \div 127$
unsigned short	2	$0 \div 65535$
signed short (short)	2	$-32768 \div 32767$
unsigned int (unsigned)	4	$0 \div 4294967295$
signed int (int)	4	$-2147483648 \div 2147483647$
unsigned long	8	$0 \div 18446744073709551615$
signed long (long)	8	$-9223372036854775808 \div 9223372036854775807$
float	4	$1.175494351 \cdot 10^{-38} \div 3.402823466 \cdot 10^{+38}$
double	8	$2.2250738585072014 \cdot 10^{-308} \div 1.7976931348623158 \cdot 10^{+308}$

2.1.5 Объявления переменных

Переменные используются для хранения значений. Переменная характеризуется типом и именем. Типы переменных приведены в табл. 2.2. Имя переменной может начинаться с подчеркивания или буквы, но не с числа. Имя переменной может включать в себя символы английского алфавита, цифры и знак подчёркивания, но не должно совпадать с ключевыми словами.

Таблица 2.2. Типы переменных

Тип переменной	Описание
Простая переменная	Отдельная переменная с одним значением целого типа или с плавающей точкой
Перечисляемая переменная	Простая переменная целого типа, принимающая одно значение из предопределенного набора именованных констант enum имя_перечисления {список значений };

Продолжение таблицы 2.2.

Тип переменной	**Описание**
Структура	Переменная, содержащая совокупность элементов, которые могут иметь различные типы struct имя_структуры {описание элемента структуры, ... }; Поля бит в структурах: struct имя_структуры { описание элемента структуры : кол-во бит, ... };
Объединение	Переменная, содержащая совокупность элементов, которые могут иметь различные типы, но занимают одну и ту же область памяти union имя_объединения {описание элемента объединения, ... };
Массив	Переменная, содержащая совокупность элементов одинакового типа Одномерный: имя_массива[размер] Двумерный: имя_массива[размер][размер]

2.1.6 Операции

Любое выражение состоит из операндов, соединенных знаками операций. Знак операции - это символ или группа символов, которые сообщают о необходимости выполнения определенных арифметических, логических или других действий.

Операции имеют либо один операнд (унарные операции), либо два операнда (бинарные операции), либо три (тернарная операция). Операция присваивания может быть как унарной, так и бинарной.

Унарные операции приведены в табл. 2.3. Унарные операции выполняются справа налево.

Операции увеличения и уменьшения увеличивают или уменьшают значение операнда на единицу и могут быть записаны как справа так и слева от операнда.

Если знак операции записан перед операндом (префиксная форма), то изменение операнда происходит до его использования в выражении. Если знак операции записан после операнда (постфиксная форма), то операнд вначале используется в выражении, а затем происходит его изменение.

Таблица 2.3. Унарные операции

Знак операции	**Операция**
–	арифметическое отрицание (отрицание и дополнение)
~	побитовое логическое отрицание (дополнение)
!	логическое отрицание
*	разадресация (косвенная адресация)
&	вычисление адреса
+	унарный плюс
++	увеличение (инкремент)
––	уменьшение (декремент)

Бинарные операции приведены в табл. 2.4. В отличие от унарных, бинарные операции выполняются слева направо.

Таблица 2.4. Бинарные операции

Группа операций	**Знак операции**	**Операция**
Мультипликативные	*	умножение
	/	деление
	%	остаток от деления
Аддитивные	+	сложение
	–	вычитание
Операции сдвига	≪	сдвиг влево
	≫	сдвиг вправо
Операции отношения	<	меньше
	<=	меньше или равно
	>	больше
	>=	больше или равно
	==	равно
	!=	не равно

Поразрядные операции	&	поразрядное И
	|	поразрядное ИЛИ
	^	поразрядное исключающее ИЛИ
Операция последовательного вычисления	,	последовательное вычисление
Операции присваивания	=	присваивание
	*=	умножение с присваиванием
	/=	деление с присваиванием
	%=	остаток от деления с присваиванием
	–=	вычитание с присваиванием
	+=	сложение с присваиванием
	≪=	сдвиг влево с присваиванием
	≫=	сдвиг вправо с присваиванием
	&=	поразрядное И с присваиванием
	|=	поразрядное ИЛИ с присваиванием
	^=	поразрядное исключающее ИЛИ с присваиванием

Тернарное выражение состоит из трех операндов, разделенных знаками тернарной операции (?) и (:), и имеет вид: <операнд1>?<операнд2>:<операнд3>.

2.1.7 Операторы

Оператор – законченная конструкция языка, реализующая определенные действия в программе. Операторы языка приведены в табл. 2.5.

Таблица 2.5. Операторы

Оператор	**Описание**	**Синтаксис**
Простой оператор	Любое выражение, которое заканчивается точкой с запятой.	выражение;

Продолжение таблицы 2.5.

Оператор	**Описание**	**Синтаксис**

Составной оператор	Последовательность операторов, заключенная в фигурные скобки.	{
		[объявление]
		...
		оператор;
		[оператор];
		...
		}

Оператор if	Условный оператор.	if (<выражение>) <оператор1> [else <оператор2>]
Оператор for	Оператор цикла, позволяющий повторять некоторый набор операторов в программе определенное количество раз.	for([<начальное-выражение>]; [<условное-выражение>]; [<выражение-цикла>]) тело-оператора
Оператор while	Оператор цикла, применяемый, когда заранее неизвестно количество повторений.	while (<выражение>) тело оператора
Оператор do while	Оператор цикла с постусловием, в котором сначала выполняется оператор, затем анализируется условие.	do тело-оператора while (<выражение>)

Оператор switch	Выбор одного оператора(-ов) из нескольких.	switch (<выражение>) {
		[объявление]
		...
		[case константное-выражение:]
		[список операторов]
		...
		[case константное-выражение:]
		[список операторов]
		...
		[default:
		[список операторов]]
		}

Оператор break	Прерывает выполнение операторов do, for, switch или while, в которых он появляется.	break;

Продолжение таблицы 2.5.

Оператор	**Описание**	**Синтаксис**
Оператор continue	Передает управление на следующую итерацию в операторах цикла do, for, while.	continue;
Оператор return	Оператор return завершает выполнение функции, в которой он задан, и возвращает управление в вызывающую функцию.	return [выражение];

2.1.8 Функции

Функция – совокупность объявлений и операторов, предназначенная для выполнения некоторой отдельной задачи. Количество функций в программе не ограничивается.

С использованием функций связаны три понятия - определение функции (описание действий, выполняемых функцией), объявление или прототип функции (задание формы обращения к функции) и вызов функции:

- Определение функции специфицирует имя функции, тип и число её формальных параметров, а также тело функции, содержащее объявления и операторы («тело функции»); в нем также может устанавливаться тип возвращаемого значения и класс памяти.
- Объявление или прототип функции определяет её имя, тип возвращаемого значения и класс памяти; в нем также могут быть установлены типы и идентификаторы для некоторых или всех аргументов функции.
- Вызов функции передает управление и фактические аргументы заданной функции.

2.1.9 Директивы препроцессора

Директивы препроцессора – инструкции препроцессору, то есть текстовому процессору, который обрабатывает текст исходного файла. Директивы препроцессора приведены в табл. 2.6.

Знак решётки (#) должен быть первым неразделительным символом в строке, содержащей директиву. Некоторые директивы содержат аргументы или значения. Любой текст, который следует за директивой (кроме аргумента или значения, который является частью директивы) должен быть заключен в скобки комментария (/* */).

Таблица 2.6. Директивы препроцессора

Директива	**Описание**	**Синтаксис**
#define	Используется для замены часто используемых в программе констант, ключевых слов, операторов и выражений содержательными идентификаторами.	#define <идентификатор> <текст>
#include	Включает содержимое исходного файла, имя пути которого задано, в текущий исходный файл.	#include "имя пути" #include <имя пути>
#if, #elif, #else, #endif	Управляют условной компиляцией, то есть позволяют подавить компиляцию части исходного файла, проверяя постоянное выражение или идентификатор.	#if <ограниченное-константное-выражение> <текст> #elif <ограниченное-константное-выражение> <текст> #else <текст> #endif

Директивы препроцессора могут появляться в произвольном месте исходного файла, но они будут воздействовать только на оставшуюся часть исходного файла, в котором они появились.

2.1.10 Математические функции

Таблица 2.7. Математические функции

Функция	**Описание**
int isnan(double x); int isnanf(float x);	Функция используется для проверки, является ли аргумент x не числом NaN.
double cos(double x); float cosf(float x);	Функция возвращает значение косинуса аргумента x.
double sin(double x); float sinf(float x);	Функция возвращает значение синуса аргумента x.
double tan(double x); float tanf(float x);	Функция возвращает значение тангенса аргумента x.

Продолжение таблицы 2.7.

Функция	**Описание**
double acos(double x); float acosf(float x);	Функция возвращает главное значение арккосинуса аргумента x.
double asin(double x); float asinf(float x);	Функция возвращает главное значение арксинуса аргумента x.
double atan(double x); float atanf(float x);	Функция возвращает главное значение арктангенса аргумента x.
double atan2(double y, double x); float atan2f(float y, float x);	Функция возвращает главное значение арктангенса аргумента y/x.
double sqrt(double x); float sqrtf(float x);	Функция возвращает значение квадратного корня аргумента x.
double fabs(double x); float fabsf(float x);	Функция возвращает абсолютное значение (модуль) аргумента x.
double pow(double x, double p); float powf(float x, float p);	Функция возвращает значение аргумента x, возведенного в степень p (x^p).
double exp(double x); float expf(float x);	Функция возвращает значение экспоненты аргумента x (e^x).
double exp2(double x); float exp2f(float x);	Функция возвращает значение числа 2 в степени x (2^x).
double log(double x); float logf(float x);	Функция возвращает значение натурального логарифма аргумента x.
double log10(double x); float log10f(float x);	Функция возвращает значение логарифма по основанию 10 аргумента x.
double log2(double x); float log2f(float x);	Функция возвращает значение логарифма по основанию 2 аргумента x.
double min(double x, double y); float minf(float x, float y);	Функция возвращает наименьшее из двух значений аргументов x и y.
double max(double x, double y); float maxf(float x, float y);	Функция возвращает наибольшее из двух значений аргументов x и y.
double floor(double x); float floorf(float x);	Функция округляет аргумент x до наибольшего целого числа, которое меньше или равно аргументу.
double ceil(double x); float ceilf(float x);	Функция округляет аргумент x до наименьшего целого числа, которое больше или равно аргументу.

Продолжение таблицы 2.7.

Функция	Описание
double trunc(double x); float truncf(float x);	Функция округляет аргумент x путем отброса дробной части, то есть возвращает целую часть аргумента.
double round(double x); float roundf(float x);	Функция округляет аргумент x до ближайшего целого числа.
double fmod(double x, double y); fmodf(double x, double y);	Функция возвращает остаток от деления x на y.

2.1.11 Функции работы с памятью

Таблица 2.8. Функции работы с памятью

Функция	Описание
void memcpy (void *dst, void *src, int size);	Функция копирует size байт из области памяти, адресуемой аргументом src, в область памяти, адресуемую аргументом dst.
void memmove (void *dst, void *src, int size);	Функция копирует size байт из области памяти, адресуемой аргументом src, в область памяти, адресуемую аргументом dst.
void memset (void *ptr, int n, int size);	Функция заполняет первые size байт области памяти, адресуемой аргументом ptr, символом n после его преобразования в unsigned char.

2.2 Среда проектирования и разработки

Для создания программ ПЛК используется кросс-платформенная свободно распространяемая интегрированная среда разработки IDE Qt Creator, которая представляет собой комплекс настраиваемых программных средств для разработки программного обеспечения.

Данное решение предлагает:

- редактор кода с подсветкой синтаксиса, определяемой пользователем;
- удобную навигацию внутри проекта;
- дополнительные элементы, помогающие визуализировать проект;
- поддержку для сборки приложений;
- использование различных компиляторов;
- возможность вывода сообщений об ошибках и предупреждений.

2.2.1 Открытие и редактирование проекта

После запуска Qt Creator открывается режим «Начало» (рис. 2.1), в котором пользователь может:

- открыть проект;
- открыть недавние сессии и проекты;
- создать новый проект;
- открыть справочную информацию.

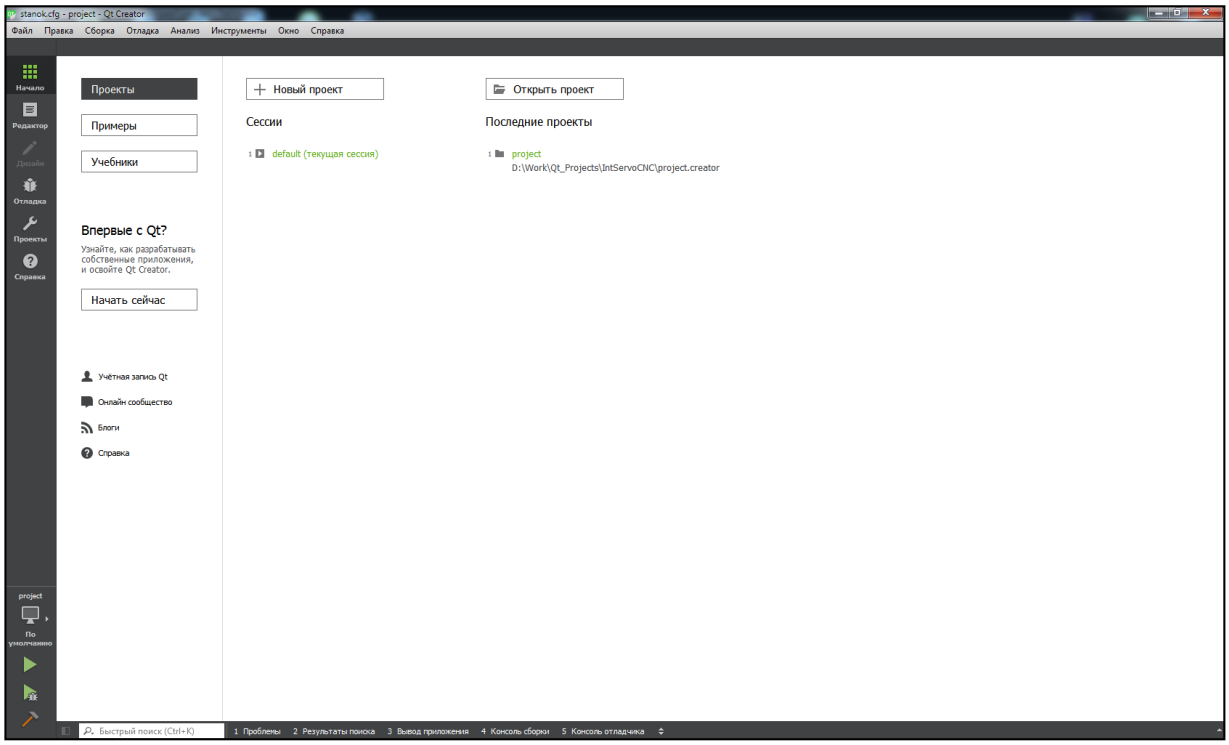

Рис. 2.1. Начальное окно Qt Creator

Для переключения режимов предназначена левая боковая панель – переключатель режимов:

- режим «Редактор» используется для редактирования проекта и файлов исходных кодов;
- режим «Отладка» используется для просмотра состояние вашей программы во время отладки;
- режим «Проекты» используется для настройки сборки и запуска проекта (режим доступен, если открыт проект);
- режим «Справка» используется для просмотра документации.

Для открытия проекта следует нажать на кнопку «Открыть проект» (сочетание клавиш Ctrl+Shift+O), перейти в каталог, в котором находятся конфигурационные файлы, и выбрать файл «project.creator». Если имя проекта присутствует в списке последних проектов, выбрать его из данного списка.

После открытия проекта Qt Creator переходит в режим «Редактор» (рис. 2.2).

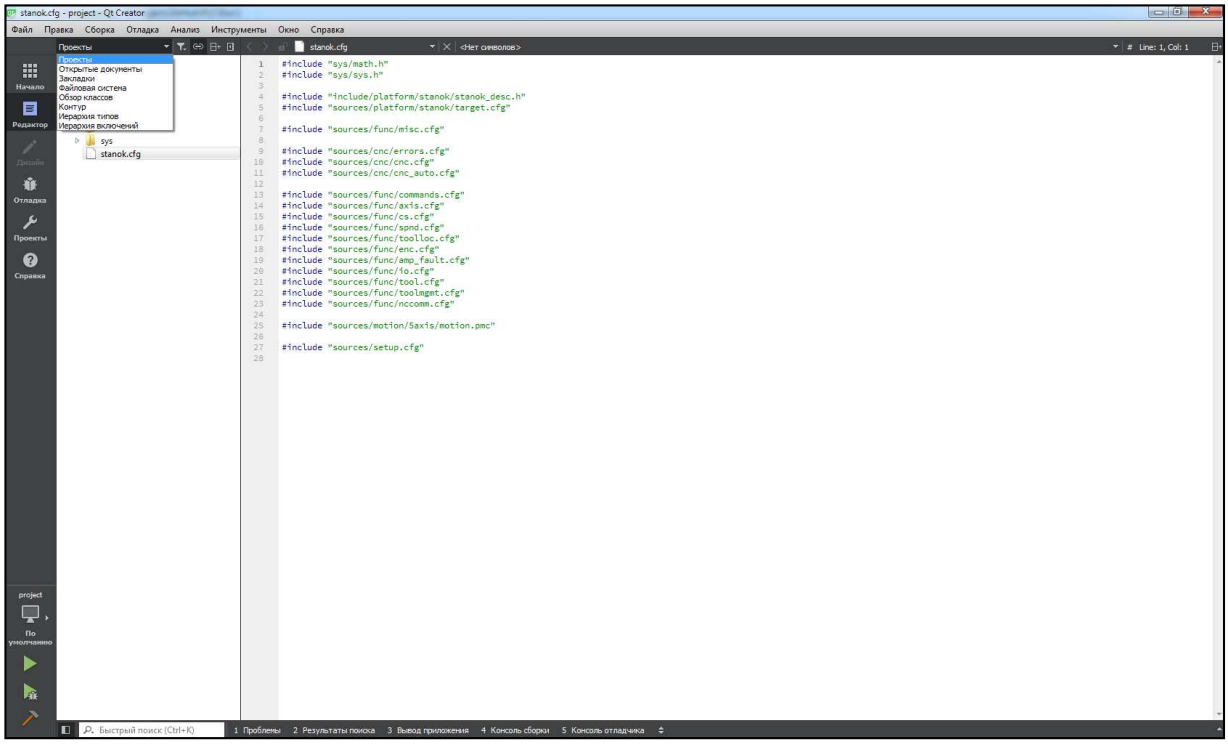

Рис. 2.2. Содержимое проекта

В меню «Проекты» боковой панели выбирается её содержимое:

- пункт «Проекты» показывает список открытых проектов в текущей сессии;
- пункт «Открытые документы» показывает открытые в настоящий момент документы;
- пункт «Закладки» показывает установленные закладки для текущей сессии;
- пункт «Файловая система» показывает содержимое проекта в каталоге;
- пункт «Обзор классов» показывает функции и пользовательские типы;
- пункт «Иерархия включений» показывает зависимости между файлами проекта.

Дерево файлов проекта на боковой панели позволяет перемещаться между директориями проекта, открывать файлы в редакторе. С помощью контекстного меню возможно добавлять существующие файлы и каталоги, переименовывать, удалять файлы и т.д.

Нижняя панель имеет несколько вкладок: «Проблемы», «Результаты поиска», «Вывод приложения», «Консоль сборки» и др, число которых настраивается пользователем.

Вкладка «Проблемы» (рис. 2.3) предоставляет список произошедших во время сборки ошибок и предупреждений. Нажатие правой кнопкой мыши на строке вызовет контекстное меню, с помощью которого можно копировать содержимое, показать в редакторе, в консоли сборки и т.д.

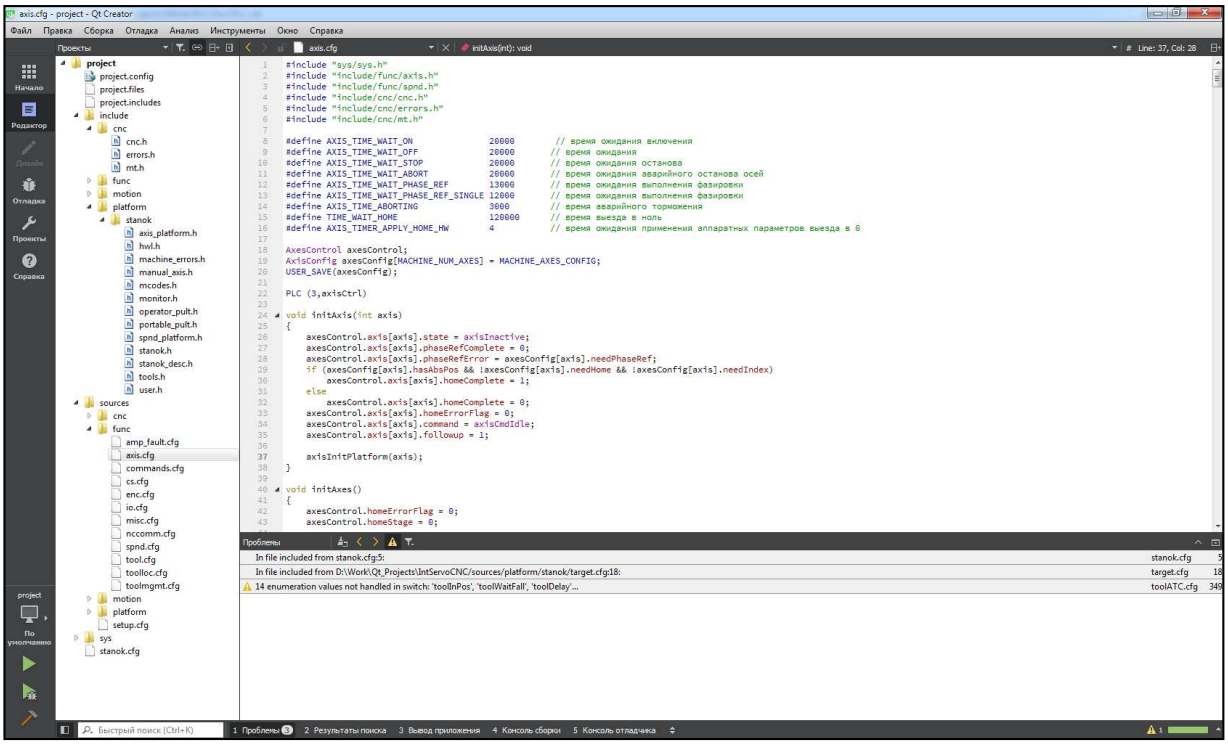

Рис. 2.3. Вывод ошибок и предупреждений

Вкладка «Результаты поиска», вызываемая также сочетанием клавиш Ctrl+Shift+F, отображает результаты глобальных поисков, таких как поиск внутри текущего документа, проекта, во всех проектах или на диске. Рис. 2.4 показывает пример результатов поиска всех упоминаний «PLC» в текущем проекте.

Вкладка «Вывод приложения» отображает статус программы при её выполнении и отладочную информацию.

Вкладка «Консоль сборки» предоставляет список произошедших во время сборки ошибок и предупреждений, который является более расширенным по сравнению с вкладкой «Проблемы».

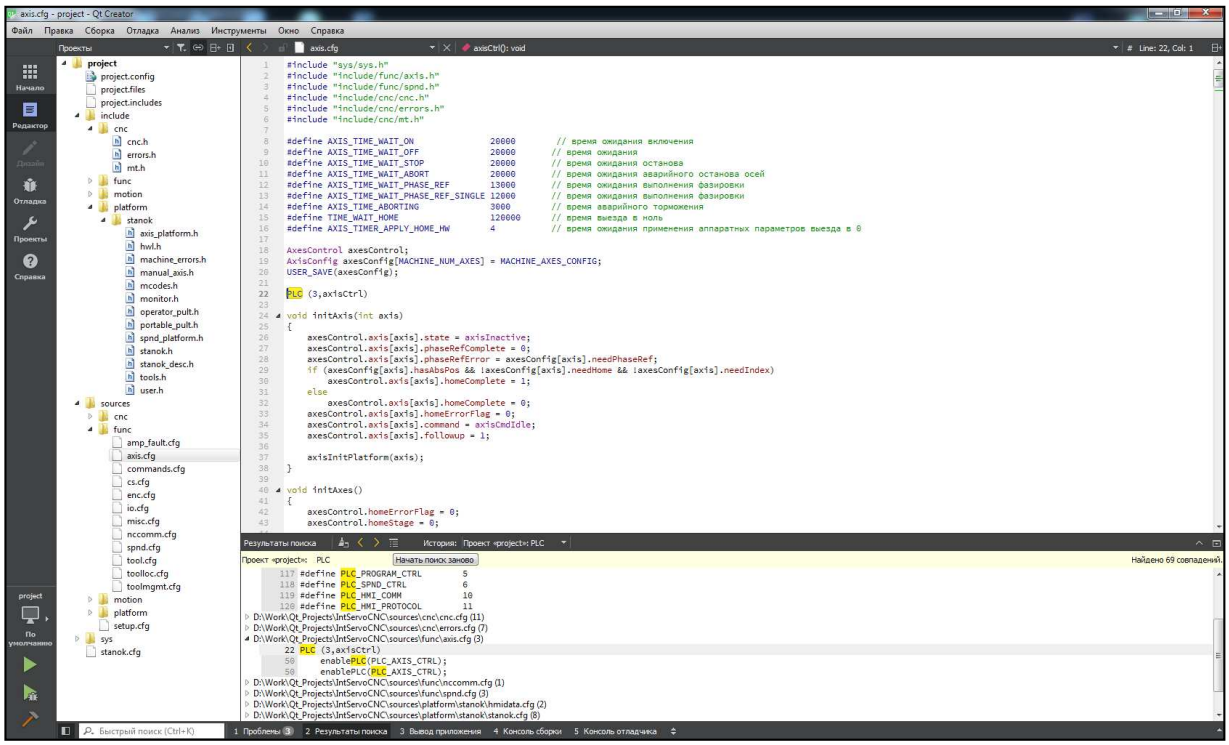

Рис. 2.4. Результаты поиска

2.2.2 Сборка проекта

Режим «Проекты» используется для настройки сборки проекта (рис. 2.5).

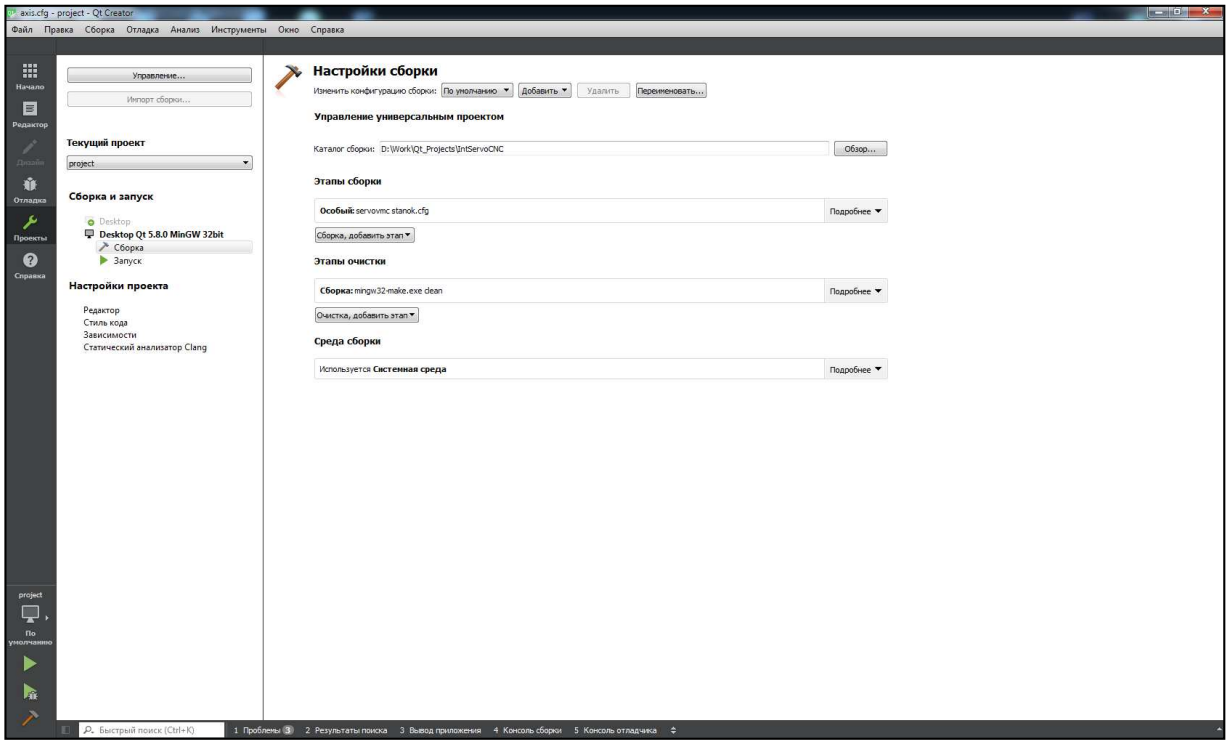

Рис. 2.5. Настройка сборки

В окне «Настройка сборки» указывается каталог сборки – каталог, в котором находятся конфигурационные файлы, и этап сборки «Особый: servovmc имя_проекта.cfg»

Сборка проекта выполняется из верхнего меню «Сборка» выбором пункта «Собрать проект» (сочетание клавиш Ctrl+B) или нажатием нижней кнопки левой боковой панели (рис. 2.6).

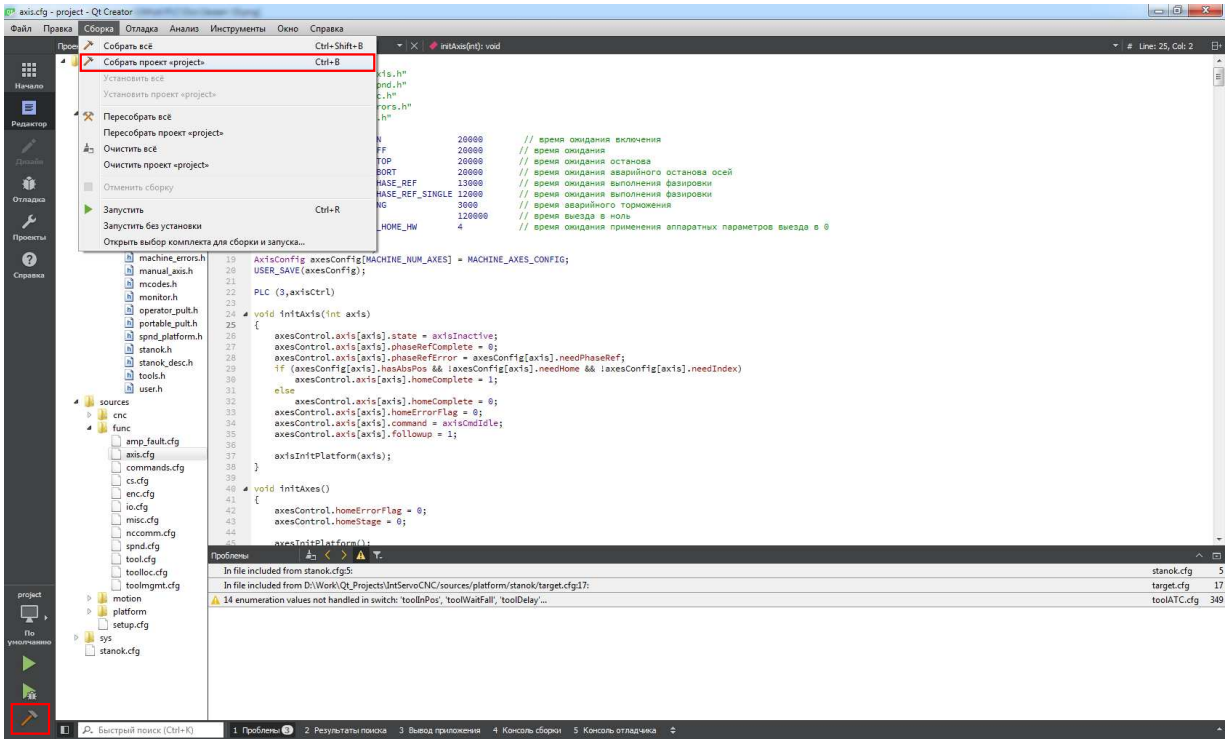

Рис. 2.6. Сборка проекта

После успешной сборки проекта в каталоге сборки будет создан файл конфигурации «config.mcc», который записывается в УЧПУ.

2.3 Объявление и реализация программ ПЛК

Для создания программы ПЛК необходимо создать новый файл с расширением `cfg` в каталоге «`source/platform/имя_проекта`».

В рассматриваемом примере: «`source/platform/stanok`».

После открытия проекта в окне дерева файлов проекта на боковой панели правой кнопкой мыши на папке с именем проекта вызвать контекстное меню, в котором выбрать пункт «Добавить новый» (рис. 2.7).

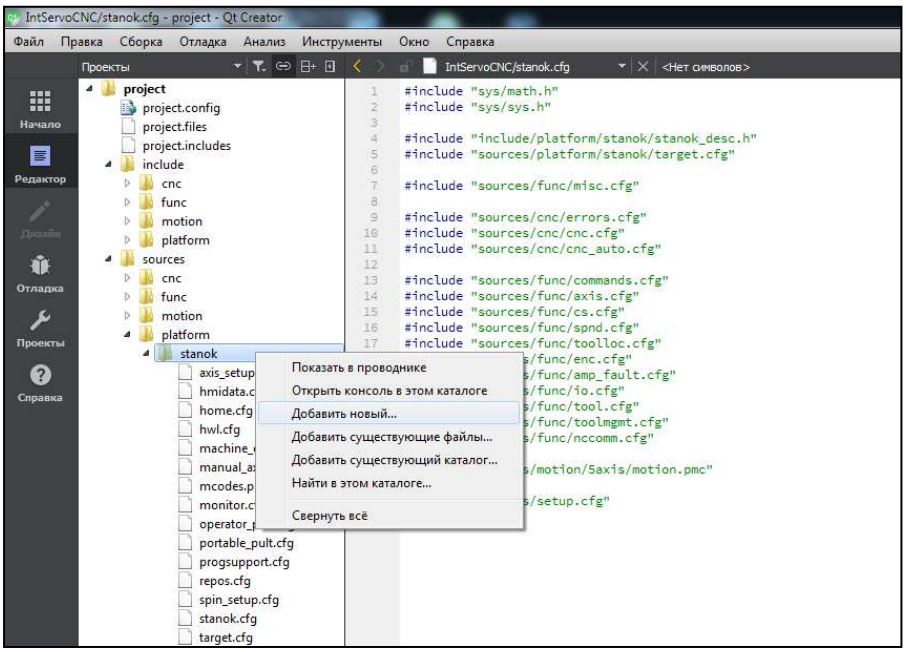

Рис. 2.7. Создание нового файла в проекте

В появившемся окне «Новый файл» выбрать шаблон «C++» и «Файл исходных текстов C++» (рис. 2.8).

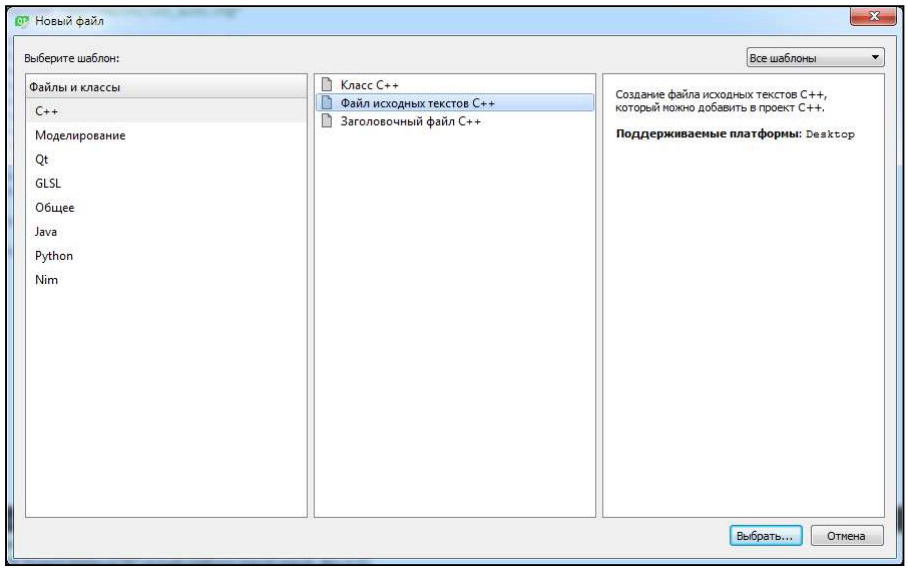

Рис. 2.8. Выбор типа файла

В следующем окне «Файл исходных текстов C++» задать имя файла с расширением `cfg`, в котором будет реализована программа ПЛК (рис. 2.9).

В рассматриваемом примере: «user_plc.cfg».

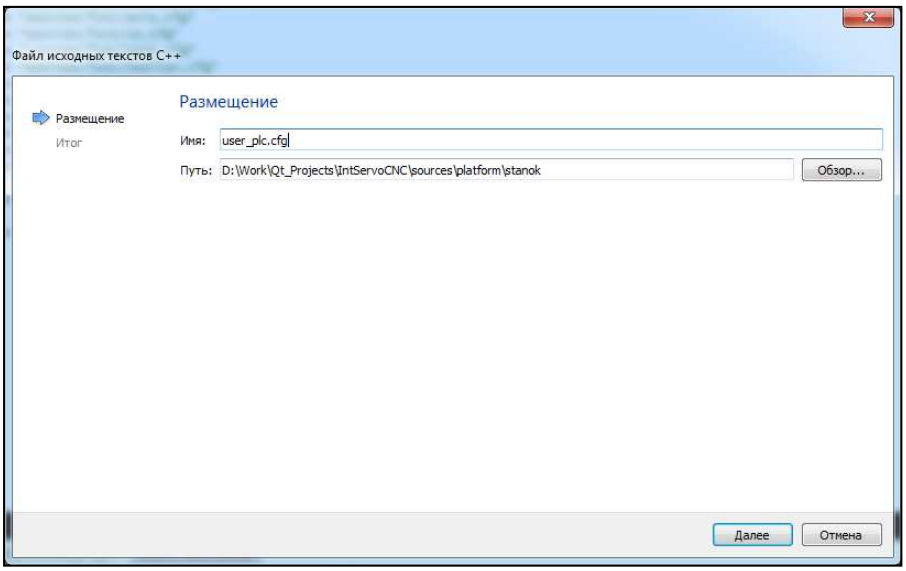

Рис. 2.9. Задание имени файла программы ПЛК с расширением

Добавить файл в текущий проект, нажав кнопку «Завершить». После добавления нового файла его имя должно появиться в окне дерева файлов проекта.

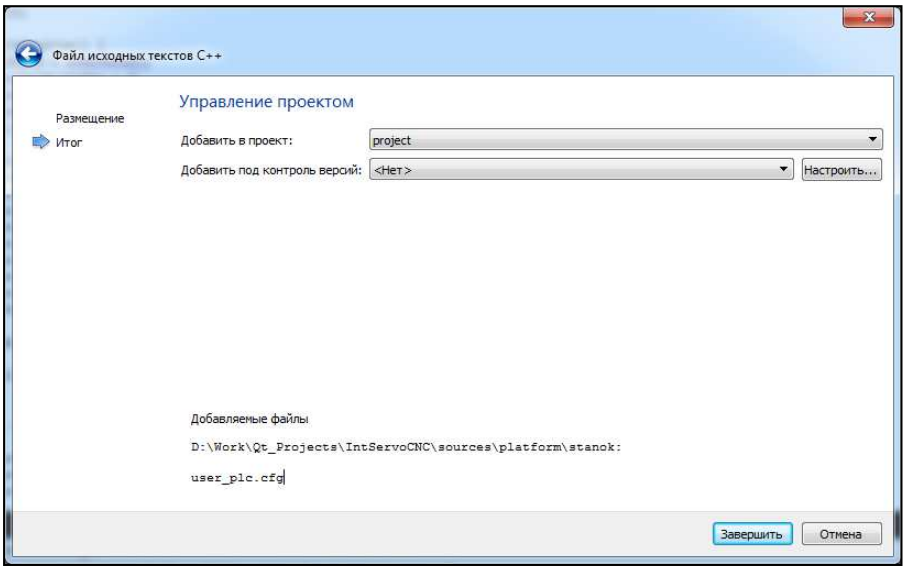

Рис. 2.10. Добавление нового файла в проект

Открыв созданный файл «user_plc.cfg» в редакторе, следует объявить в нём ПЛК программу строкой `PLC (номер_программы, имя_функции)` и реализовать определение функции.

В рассматриваемом примере: `PLC (9, user_conrol)`.

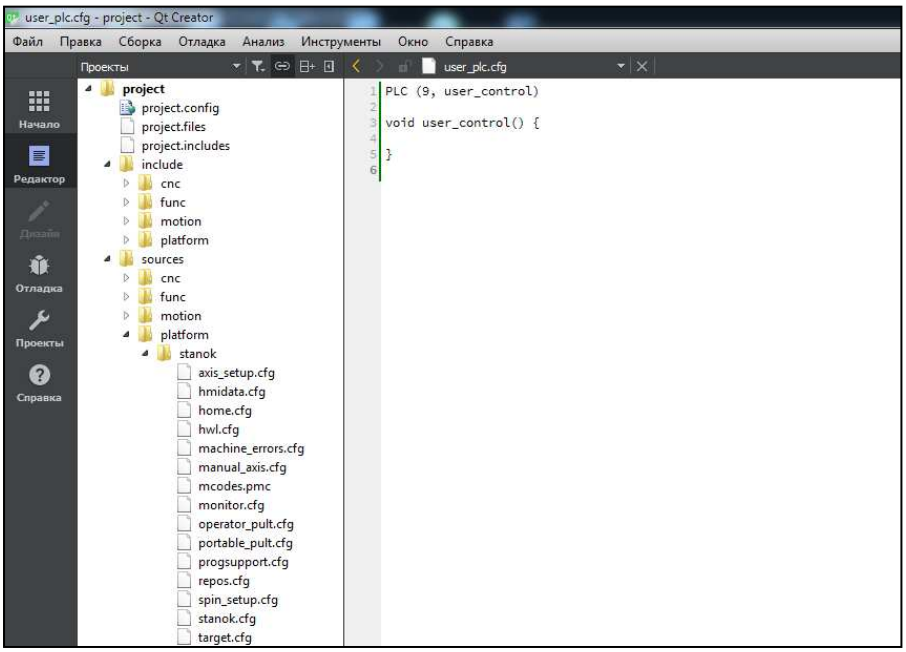

Рис. 2.11. Объявление программы ПЛК

В файл «source/platform/имя_проекта/target.cfg» добавить строку с именем созданного файла.

В рассматриваемом примере: `#include "user_plc.cfg"`.

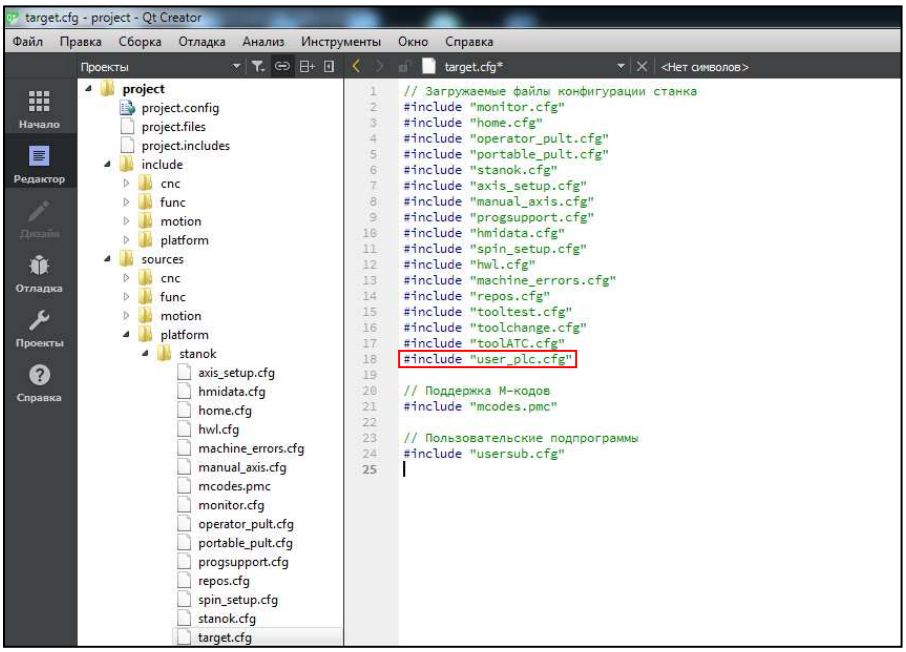

Рис. 2.12. Включение файла программы ПЛК в конфигурационный файл «target.cfg»

В файле «include/platform/имя_проекта/stanok_desc.h» определить идентификатор, соответствующий номеру программы ПЛК.

В рассматриваемом примере: `#define PLC_USER_CTRL 9`.

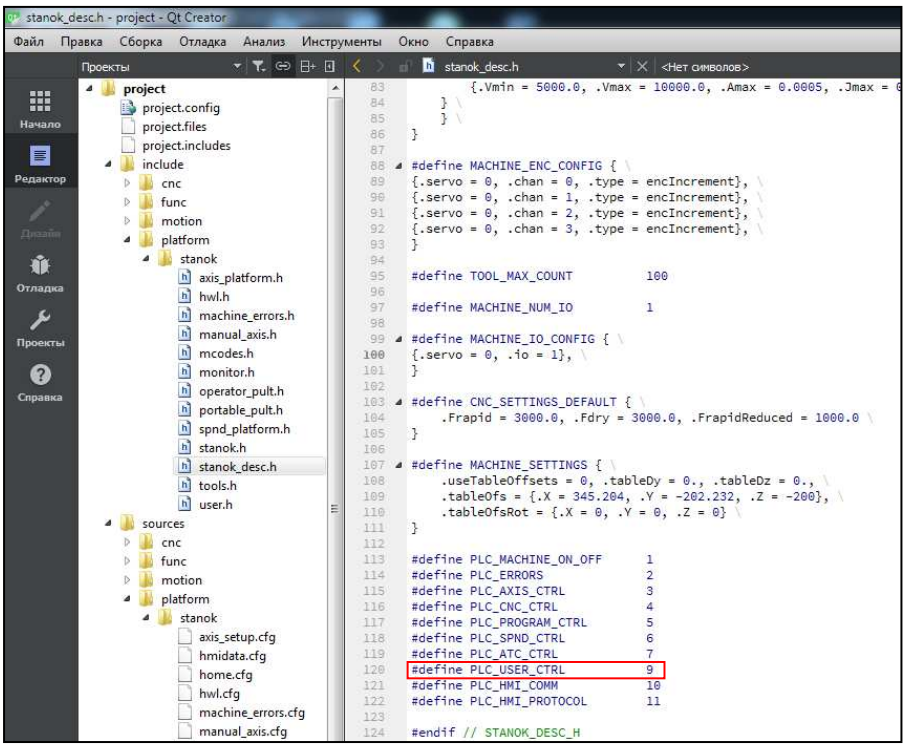

Рис. 2.13. Определение идентификатора программы ПЛК

Для разрешения выполнения программы ПЛК необходимо добавить вызов функции `enablePLC(идентификатор)` в файле «`source/platform/имя_проекта/stanok.cfg`» в функцию `initMachine()`.

В рассматриваемом примере: `enablePLC(PLC_USER_CTRL)`.

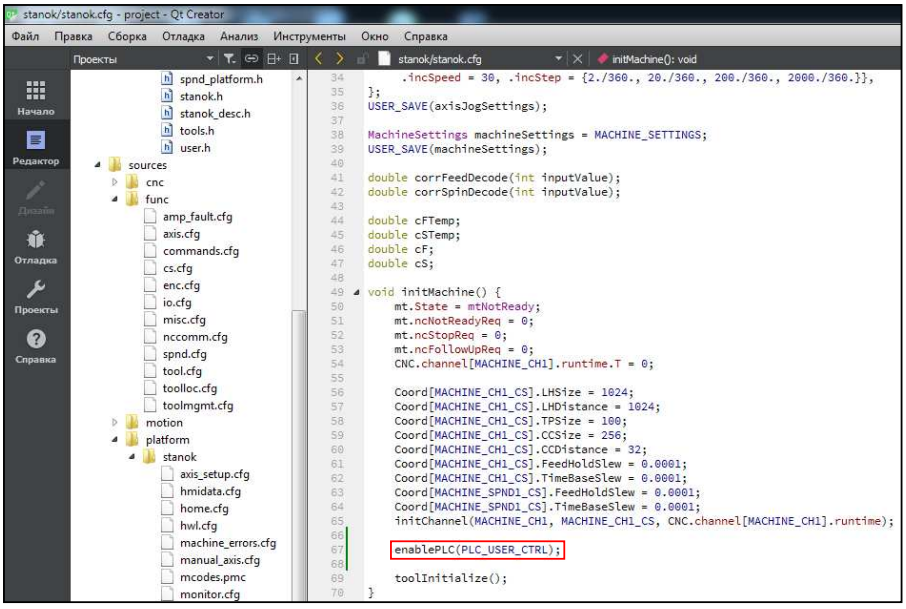

Рис. 2.14. Разрешение выполнения программы ПЛК

2.4 Предопределённые функции

Предопределёнными функциями в системе управления являются:

- `void setup();`
- `void motion_nc();`
- `void motion##()`, где ## – целое число (1, 2, 3, . . .).

Функция `void setup()` – точка старта программы пользователя, определяемая в файле `setup.cfg`. Данной функции передаётся управление после запуска системы.

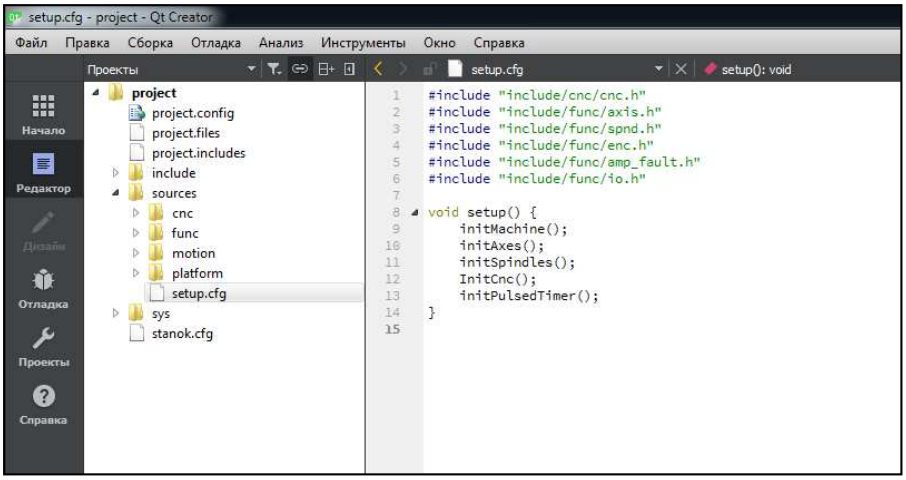

Рис. 2.15. Содержимое файла `setup.cfg`

Функция `void motion_nc()` является пользовательской программой движения №0, функции `void motion##()` – программы движения с номерами ##.

Пример программы движение приведён в **ПРИЛОЖЕНИИ 1** в листинге «Задание программы движения» на стр. 154.

2.5 Загрузка конфигурации в УЧПУ

Загрузка файла конфигурации «config.mcc» в УЧПУ осуществляется по протоколу SCP, предназначенного для защищённого копирования файлов.

Для загрузки файла конфигурации из ОС Linux используется команда «`scp`», которая имеет следующий синтаксис: «`scp source_file_name username@destination_host:destination_folder`». Основная команда SCP без параметров копирует файлы в фоновом режиме. Параметр «-v» команды «`scp`» служит для вывода отладочной информации на экран, что может помочь настроить соединение, аутентификацию и устранить проблемы конфигурации.

Пример использования команды «`scp`» загрузки файла «config.mcc» в УЧПУ: «`scp config.mcc root@192.168.1.90:/root/`».

Загрузка файла конфигурации из ОС Windows выполняется посредством свободно распространяемой (лицензия GNU GPL) программы – графической оболочки-клиента WinSCP.

После запуска программы WinSCP необходимо ввести параметры нового подключения в окне «Вход» (рис. 2.16):

- протокол передачи – SCP;
- имя хоста – IP-адрес УЧПУ, номер порта оставить по умолчанию;
- имя пользователя и пароль (по умолчанию root и 123456).

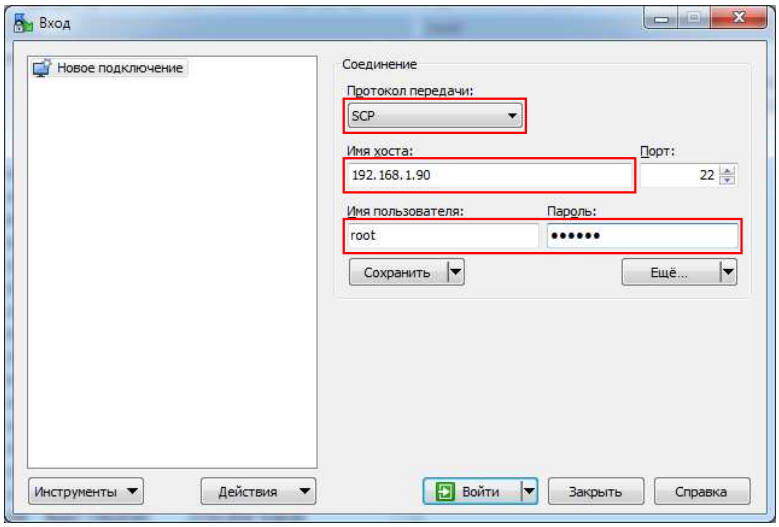

Рис. 2.16. Ввод параметров подключения

Записать введённые параметры, нажав кнопку «Сохранить».

В окне диалоговом окне «Сохранить как новое подключение» оставить настройки сохранения без изменений и нажать кнопку «OK» (рис. 2.17).

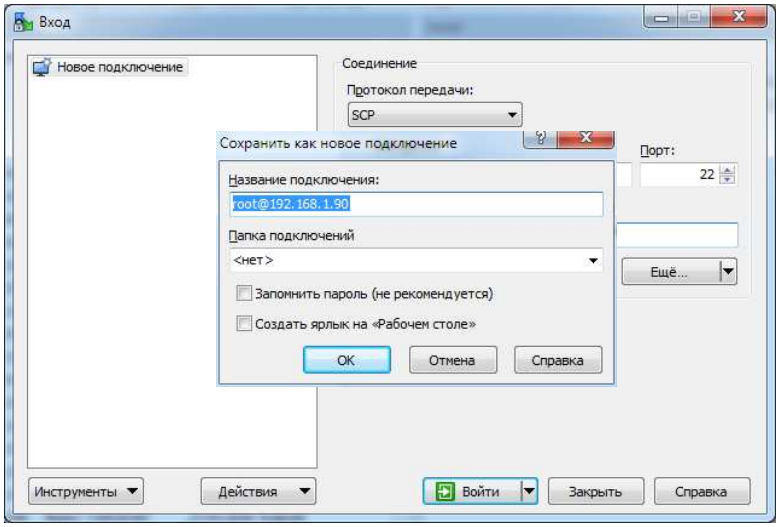

Рис. 2.17. Сохранение параметров подключения

Подключиться к УЧПУ, нажав кнопку «Войти» (рис. 2.18).

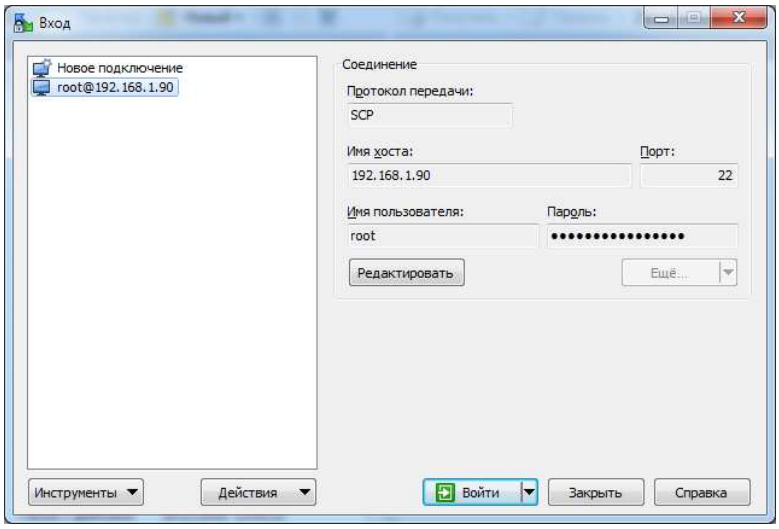

Рис. 2.18. Подключение к УЧПУ

После подключения к УЧПУ в правой панели программы отобразится удалённый каталог УЧПУ для загрузки файлов конфигурации. В левой панели следует перейти в каталог сборки проекта и переписать файл «config.mcc» в каталог УЧПУ `var/lib/motioncore/config/` в правой панели.

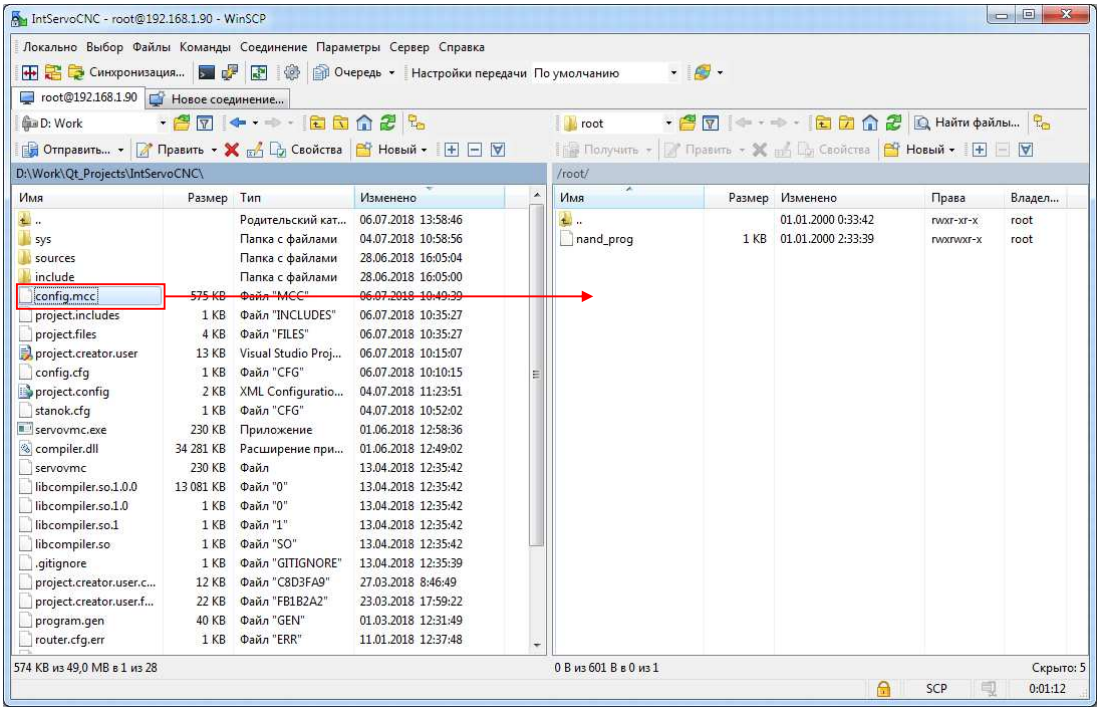

Рис. 2.19. Загрузка файла конфигурации в УЧПУ

Для того, чтобы изменения вступили в силу необходимо перезагрузить УЧПУ командой $$$ через программную оболочку ServoIDE или отключением и включением питания.

3 . Программный интерфейс ПЛК

Программный интерфейс ПЛК – набор типов данных, констант, макросов и функций, предоставляемых системой пользователю для создания программ ПЛК.

3.1 Управление УЧПУ

3.1.1 Типы данных

3.1.1.1 CNCMode

Тип данных: *Перечисление CNCMode*
Файл объявления: *include/cnc/cnc.h*

Перечисление определяет идентификаторы режимов работы УЧПУ.

Таблица 3.1. Перечисление CNCMode

Идентификатор	Описание
cncNull	Режим не определён
cncOff	УЧПУ не активно
cncManual	Ручной режим
cncHome	Режим выезда в нулевую точку
cncHWL	Режим дискретных перемещений
cncAuto	Автоматический режим
cncStep	Пошаговый режим
cncMDI	Режим преднабора
cncVirtual	Виртуальный режим
cncReset	Режим сброса
cncRepos	Режим возврата на контур
cncWaitChangeMode	Ожидание смены режима

3.1.1.2 ChannelStatus

Тип данных: *Перечисление ChannelStatus*
Файл объявления: *include/cnc/cnc.h*

Перечисление определяет идентификаторы состояний канала управления.

Таблица 3.2. Перечисление ChannelStatus

Идентификатор	Описание
channelReset	Готовность
channelInterrupted	Работа прервана
channelActive	Активен

3.1.1.3 ModeState

Тип данных: *Перечисление ModeState*
Файл объявления: *include/cnc/cnc.h*

Перечисление определяет идентификаторы состояний текущего режима УЧПУ.

Таблица 3.3. Перечисление ModeState

Идентификатор	Описание
modeReset	Готовность
modeRunning	Выполнение
modeStopped	Останов

3.1.1.4 ProgramSeekMode

Тип данных: *Перечисление ProgramSeekMode*
Файл объявления: *include/cnc/cnc.h*

Перечисление определяет идентификаторы режимов выполнения УП с произвольного кадра.

Таблица 3.4. Перечисление ProgramSeekMode

Идентификатор	Описание
seekNone	Режим не активен
seekApproach	Выполнение УП с начала выбранного кадра
seekWithoutApproach	Выполнение УП с конца выбранного кадра
seekWithoutCalc	Выполнение УП без расчёта фрагмента программы до выбранного кадра

3.1.1.5 ProgramStatus

Тип данных: *Перечисление ProgramStatus*
Файл объявления: *include/cnc/cnc.h*

Перечисление определяет идентификаторы состояний УП.

Таблица 3.5. Перечисление ProgramStatus

Идентификатор	Описание
programAborted	Выполнение УП прервано и завершено
programInterrupted	Выполнение УП временно прервано для какой-либо операции
programStopped	Выполнение УП остановлено
programRunning	УП выполняется

3.1.1.6 ShutdownState

Тип данных: *Перечисление ShutdownState*
Файл объявления: *include/cnc/cnc.h*

Перечисление определяет идентификаторы состояний автомата выключения УЧПУ и станка.

Таблица 3.6. Перечисление ShutdownState

Идентификатор	Описание
shutdownWaitCommand	Ожидание команды выключения
shutdownWaitAck	Ожидание подтверждения команды выключения

3.1.1.7 ChannelInfo

Тип данных: *Структура ChannelInfo*
Файл объявления: *include/cnc/cnc.h*

Структура определяет данные канала управления.

Таблица 3.7. Структура ChannelInfo

Элемент	Тип	Описание
canLoad	Битовое поле:1	Разрешение загрузки УП
starting	Битовое поле:1	Подготовка к выполнению УП
running	Битовое поле:1	Выполнение УП
holding	Битовое поле:1	УП в процессе останова или возобновления
stopped	Битовое поле:1	УП не выполняется
waitingBlock	Битовое поле:1	Запрос поиска кадра
seekingBlock	Битовое поле:1	Выполнение поиска кадра
virtualStart	Битовое поле:1	Подготовка к выполнению УП в виртуальном режиме
virtualRun	Битовое поле:1	Выполнение УП в виртуальном режиме
canLoadMDI	Битовое поле:1	Разрешение загрузки УП в режиме преднабора
startingMDI	Битовое поле:1	Подготовка к выполнению УП в режиме преднабора
runningMDI	Битовое поле:1	Выполнение УП в режиме преднабора
holdingMDI	Битовое поле:1	УП в режиме преднабора в процессе останова или возобновления
waitingMDI	Битовое поле:2	0 – УП загружена для выполнения в режиме преднабора 1 – запрос загрузки УП для выполнения в режиме преднабора 2 – ошибка загрузки УП для выполнения в режиме преднабора

Продолжение таблицы 3.7.

Элемент	Тип	Описание
mdiReady	Битовое поле:1	УП в режиме преднабора готова к выполнению
switchToRepos	Битовое поле:1	Разрешение перехода в режим возврата на контур
setActual	Битовое поле:8	Младшие 4 бита – команда: 1 – текущая позиция = 0; 2 – текущая позиция = машинная позиция; 3 – текущая позиция = программная позиция Старшие 4 бита – область применения: 0 – все оси; другие значения определяются конфигурацией станка
res	Битовое поле:7	Резерв
Pos[ЧИСЛО_ОСЕЙ]	double	Программная позиция
WorkPos[ЧИСЛО_ОСЕЙ]	double	Программная позиция относительно базового смещения
MachPos[ЧИСЛО_ОСЕЙ]	double	Машинная позиция
TargetPos[ЧИСЛО_ОСЕЙ]	double	Конечная позиция текущего кадра
DistToGo[ЧИСЛО_ОСЕЙ]	double	Остаток пути
ActualPos[ЧИСЛО_ОСЕЙ]	double	Текущая позиция
ActualBase[ЧИСЛО_ОСЕЙ]	double	Базовое смещение текущей позиции
state	**ChannelStatus**	Состояние канала управления
modeState	**ModeState**	Состояние текущего режима УЧПУ
runtime	**ProgramRuntime**	Данные УП
startBlock	unsigned	Начальный блок поиска кадра при выполнении УП с произвольного кадра
blockMode	unsigned	Режим выполнения УП с произвольного кадра
seekCount	unsigned	Номер итерации поиска кадра

3.1.1.8 CNCDesc

Тип данных: *Структура CNCDesc*
Файл объявления: *include/cnc/cnc.h*

Структура определяет данные УЧПУ.

Таблица 3.8. Структура CNCDesc

Элемент	Тип	Описание
mode	**CNCMode**	Текущий режим работы УЧПУ
prevMode	**CNCMode**	Предыдущий режим работы УЧПУ
nextMode	**CNCMode**	Следующий режим работы УЧПУ
Watchdog	int	Счётчик сторожевого таймера
HMIFeedback	int	Флаг обратной связи пульта оператора
HMIFirstStart	int	Флаг включения пульта оператора (до включения пульта оператора равен 1)
hmiTripped	int	Флаг срабатывания сторожевого таймера
HMIWatchdog	**Timer**	Таймер сторожевого таймера связи с пультом оператора
shutdown	**Timer**	Таймер выключения УЧПУ и станка
modeAutoStep	unsigned	Флаг покадровой отработки УП
modeAutoVirtual	unsigned	Флаг отработки УП в виртуальном режиме
modeAutoSkip	unsigned	Флаг программного пропуска кадров при отработке УП
modeAutoOptStop	unsigned	Флаг опционального останова при отработке УП
modeAutoRepos	unsigned	Флаг возврата на контур при возобновлении выполнения УП
alarmCancel	unsigned	Запрос сброса ошибок
modeDryRun	unsigned	Флаг пробной подачи при отработке УП
modeReducedG0	unsigned	Флаг уменьшенной подачи быстрого хода при отработке УП
nodeNoMovement	unsigned	Флаг отработки УП с блокировкой движения
request	**MTCNCRequests**	Текущая исполняемая команда УЧПУ
channel[ЧИСЛО_КАНАЛОВ]	**ChannelInfo**	Данные канала управления
notReadyReq	Битовое поле:1	УЧПУ не готово
startDisableReq	Битовое поле:1	Запрет запуска УП
enablePortablePult	Битовое поле:1	Разрешение работы переносного пульта

Продолжение таблицы 3.8.

Элемент	Тип	Описание
ShutdownHMI	int	Переменная выключения УЧПУ и станка принимает значения: 0x5A при включении УЧПУ, 0xA5 – при получении команды, выключения, 0x55 – при подтверждении команды выключения
ShutdownState	int	Состояние автомата выключения УЧПУ и станка
commands	**CommandQueue**	Очередь команд

3.1.1.9 CNCSettings

Тип данных: *Структура CNCSettings*
Файл объявления: *include/cnc/cnc.h*

Структура определяет значения подачи для различных режимов.

Таблица 3.9. Структура CNCSettings

Элемент	Тип	Описание
Frapid	double	Значение подачи быстрого хода
Fdry	double	Значение пробной подачи
FrapidReduced	double	Значение уменьшенной подачи быстрого хода

3.1.2 Функции

3.1.2.1 InitCnc

Синтаксис: *void InitCnc();*
Аргумент(ы): *Нет*
Файл объявления: *include/cnc/cnc.h*

Функция инициализации УЧПУ.
Является системной.

3.1.2.2 mtIsReady

Синтаксис:	*int mtIsReady();*
Аргумент(ы):	*Нет*
Файл объявления:	*include/cnc/cnc.h*

Функция проверки готовности станка к работе.
Функция возвращает 1, если станок готов, и 0 в противном случае.
Реализуется пользователем.

3.1.2.3 cncSetMode

Синтаксис:	*void cncSetMode(CNCMode mode);*
Аргумент(ы):	***CNCMode** mode – идентификатор режима работы УЧПУ*
Файл объявления:	*include/cnc/cnc.h*

Функция устанавливает режим работы УЧПУ, принимая в качестве аргумента значение одного из идентификаторов перечисления **CNCMode**.
Является системной.

3.1.2.4 cncRequest

Синтаксис:	*void cncRequest (MTCNCRequests request);*
Аргумент(ы):	***MTCNCRequests** request – идентификатор команды управления станком*
Файл объявления:	*include/cnc/cnc.h*

Функция посылает команду УЧПУ, принимая в качестве аргумента значение одного из идентификаторов перечисления **MTCNCRequests**.
Является системной.

3.1.2.5 cncChangeMode

Синтаксис:	*void cncChangeMode (int newMode);*
Аргумент(ы):	*int newMode – идентификатор режима работы УЧПУ*
Файл объявления:	*include/cnc/cnc.h*

Функция выполняет запрос изменения режима работы УЧПУ.
Является системной.

3.1.2.6 channelUpdate

Синтаксис:	*void channelUpdate (int channel);*
Аргумент(ы):	*int channel – номер канала*
Файл объявления:	*include/cnc/cnc.h*

Функция обновляет данные канала, номер которого задаётся в качестве аргумента.
Является системной.

3.1.2.7 cncModeManual

Синтаксис:	*void cncModeManual();*
Аргумент(ы):	*Нет*
Файл объявления:	*include/cnc/cnc.h*

Функция обработки команд в ручном режиме работы УЧПУ.
Является системной.

3.1.2.8 cncModeHome

Синтаксис:	*void cncModeHome();*
Аргумент(ы):	*Нет*
Файл объявления:	*include/cnc/cnc.h*

Функция обработки команд в режиме выезда в нулевую точку УЧПУ.
Является системной.

3.1.2.9 cncModeHandwheel

Синтаксис:	*void cncModeHandwheel();*
Аргумент(ы):	*Нет*
Файл объявления:	*include/cnc/cnc.h*

Функция обработки команд в режиме дискретных перемещений УЧПУ.
Является системной.

3.1.2.10 cncModeAuto

Синтаксис:	*void cncModeAuto();*
Аргумент(ы):	*Нет*
Файл объявления:	*include/cnc/cnc.h*

Функция обработки команд в автоматическом режиме работы УЧПУ.
Является системной.

3.1.2.11 cncModeMDI

Синтаксис:	*void cncModeMDI();*
Аргумент(ы):	*Нет*
Файл объявления:	*include/cnc/cnc.h*

Функция обработки команд в режиме преднабора УЧПУ.
Является системной.

3.1.2.12 cncModeRepos

Синтаксис:	*void cncModeRepos();*
Аргумент(ы):	*Нет*
Файл объявления:	*include/cnc/cnc.h*

Функция обработки команд в режиме возврата на контур УЧПУ.
Является системной.

3.1.2.13 cncManualEnter

Синтаксис:	*void cncManualEnter());*
Аргумент(ы):	*Нет*
Файл объявления:	*include/cnc/cnc.h*

Функция вызывается при установке ручного режима работы УЧПУ. В ней должны определяться действия, выполняемые при входе в данный режим.
Реализуется пользователем.

3.1.2.14 cncManualLeave

Синтаксис:	*int cncManualLeave (CNCMode newMode);*
Аргумент(ы):	***CNCMode** newMode – идентификатор режима работы УЧПУ*
Файл объявления:	*include/cnc/cnc.h*

Функция вызывается при выходе из ручного режима работы УЧПУ. В ней должны определяться действия, выполняемые при выходе из данного режима, а также осуществляться проверка возможности установки нового режима работы УЧПУ, который задаётся аргументом – значением одного из идентификаторов перечисления **CNCMode**.
Возвращаемое значение должно быть отлично от 0 для разрешения нового режима работы.
Реализуется пользователем.

3.1.2.15 cncHwlEnter

Синтаксис:	*void cncHwlEnter();*
Аргумент(ы):	*Нет*
Файл объявления:	*include/cnc/cnc.h*

Функция вызывается при установке режима дискретных перемещений УЧПУ. В ней должны определяться действия, выполняемые при входе в данный режим.
Реализуется пользователем.

3.1.2.16 cncHwlLeave

Синтаксис:	*int cncHwlLeave (CNCMode newMode);*
Аргумент(ы):	***CNCMode** newMode – идентификатор режима работы УЧПУ*
Файл объявления:	*include/cnc/cnc.h*

Функция вызывается при выходе из режима дискретных перемещений УЧПУ. В ней должны определяться действия, выполняемые при выходе из данного режима, а также осуществляться проверка возможности установки нового режима работы УЧПУ, который задаётся аргументом – значением одного из идентификаторов перечисления **CNCMode**.

Возвращаемое значение должно быть отлично от 0 для разрешения нового режима работы.

Реализуется пользователем.

3.1.2.17 cncHomeEnter

Синтаксис:	*void cncHomeEnter();*
Аргумент(ы):	*Нет*
Файл объявления:	*include/cnc/cnc.h*

Функция вызывается при установке режима выезда в нулевую точку УЧПУ. В ней должны определяться действия, выполняемые при входе в данный режим.

Реализуется пользователем.

3.1.2.18 cncHomeLeave

Синтаксис:	*int cncHomeLeave (CNCMode newMode);*
Аргумент(ы):	***CNCMode** newMode – идентификатор режима работы УЧПУ*
Файл объявления:	*include/cnc/cnc.h*

Функция вызывается при выходе из режима выезда в нулевую точку УЧПУ. В ней должны определяться действия, выполняемые при выходе из данного режима, а также осуществляться проверка возможности установки нового режима работы УЧПУ, который задаётся аргументом – значением одного из идентификаторов перечисления **CNCMode**.

Возвращаемое значение должно быть отлично от 0 для разрешения нового режима работы.

Реализуется пользователем.

3.1.2.19 cncAutoEnter

Синтаксис:	*void cncAutoEnter();*
Аргумент(ы):	*Нет*
Файл объявления:	*include/cnc/cnc.h*

Функция вызывается при установке автоматического режима УЧПУ. В ней должны определяться действия, выполняемые при входе в данный режим.

Реализуется пользователем.

3.1.2.20 cncAutoLeave

Синтаксис:	*int cncAutoLeave (CNCMode newMode);*
Аргумент(ы):	***CNCMode** newMode – идентификатор режима работы УЧПУ*
Файл объявления:	*include/cnc/cnc.h*

Функция вызывается при выходе из автоматического режима УЧПУ. В ней должны определяться действия, выполняемые при выходе из данного режима, а также осуществляться проверка возможности установки нового режима работы УЧПУ, который задаётся аргументом – значением одного из идентификаторов перечисления **CNCMode**.

Возвращаемое значение должно быть отлично от 0 для разрешения нового режима работы.

Реализуется пользователем.

3.1.2.21 cncMDIEnter

Синтаксис:	*void cncMDIEnter();*
Аргумент(ы):	*Нет*
Файл объявления:	*include/cnc/cnc.h*

Функция вызывается при установке режима преднабора УЧПУ. В ней должны определяться действия, выполняемые при входе в данный режим.

Реализуется пользователем.

3.1.2.22 cncMDILeave

Синтаксис:	*int cncMDILeave (CNCMode newMode);*
Аргумент(ы):	***CNCMode** newMode – идентификатор режима работы УЧПУ*
Файл объявления:	*include/cnc/cnc.h*

Функция вызывается при выходе из режима преднабора УЧПУ. В ней должны определяться действия, выполняемые при выходе из данного режима, а также осуществляться проверка возможности установки нового режима работы УЧПУ, который задаётся аргументом – значением одного из идентификаторов перечисления **CNCMode**.

Возвращаемое значение должно быть отлично от 0 для разрешения нового режима работы.

Реализуется пользователем.

3.1.2.23 cncReposEnter

Синтаксис:	*void cncReposEnter();*
Аргумент(ы):	*Нет*
Файл объявления:	*include/cnc/cnc.h*

Функция вызывается при установке режима возврата на контур УЧПУ. В ней должны определяться действия, выполняемые при входе в данный режим.

Реализуется пользователем.

3.1.2.24 cncReposLeave

Синтаксис:	*int cncReposLeave (CNCMode newMode);*
Аргумент(ы):	***CNCMode** newMode – идентификатор режима работы УЧПУ*
Файл объявления:	*include/cnc/cnc.h*

Функция вызывается при выходе из режима возврата на контур УЧПУ. В ней должны определяться действия, выполняемые при выходе из данного режима, а также осуществляться проверка возможности установки нового режима работы УЧПУ, который задаётся аргументом – значением одного из идентификаторов перечисления **CNCMode**.

Возвращаемое значение должно быть отлично от 0 для разрешения нового режима работы.

Реализуется пользователем.

3.1.2.25 controlPowerCNC

Синтаксис:	*void controlPowerCNC (int request);*
Аргумент(ы):	*int request – идентификатор команды управления станком*
Файл объявления:	*include/cnc/cnc.h*

Функция обработки запроса выключения УЧПУ и станка. Аргументом функции является значение одного из идентификаторов перечисления **MTCNCRequests**.

Является системной.

3.1.2.26 cncAutoOnProgramExit

Синтаксис:	*void cncAutoOnProgramExit (int channel);*
Аргумент(ы):	*int channel – номер канала*
Файл объявления:	*include/cnc/cnc.h*

Функция вызывается при выходе из автоматического режима УЧПУ. В ней должны определяться действия, выполняемые при выходе из данного режима для канала, номер которого является аргументом функции.

Реализуется пользователем.

3.1.2.27 cncCustomRequestManual

Синтаксис:	*void cncCustomRequestManual (int request);*
Аргумент(ы):	*int request – идентификатор команды пользователя*
Файл объявления:	*include/cnc/cnc.h*

Функция обработки пользовательских команд в ручном режиме УЧПУ. Аргументом функции является команда пользователя.

Реализуется пользователем.

3.1.2.28 cncCustomRequestHome

Синтаксис:	*void cncCustomRequestHome (int request);*
Аргумент(ы):	*int request – идентификатор команды пользователя*
Файл объявления:	*include/cnc/cnc.h*

Функция обработки пользовательских команд в режиме выезда в нулевую точку УЧПУ. Аргументом функции является команда пользователя.

Реализуется пользователем.

3.1.2.29 cncCustomRequestAuto

Синтаксис:	*void cncCustomRequestAuto (int request);*
Аргумент(ы):	*int request – идентификатор команды пользователя*
Файл объявления:	*include/cnc/cnc.h*

Функция обработки пользовательских команд в автоматическом режиме УЧПУ. Аргументом функции является команда пользователя.

Реализуется пользователем.

3.1.2.30 cncCustomRequestMDI

Синтаксис:	*void cncCustomRequestMDI (int request);*
Аргумент(ы):	*int request – идентификатор команды пользователя*
Файл объявления:	*include/cnc/cnc.h*

Функция обработки пользовательских команд в режиме преднабора УЧПУ. Аргументом функции является команда пользователя.

Реализуется пользователем.

3.1.2.31 cncCustomRequestHwl

Синтаксис:	*void cncCustomRequestHwl (int request);*
Аргумент(ы):	*Целое знаковое число*
Аргумент(ы):	*int request – идентификатор команды пользователя*
Файл объявления:	*include/cnc/cnc.h*

Функция обработки пользовательских команд в режиме дискретных перемещений УЧПУ. Аргументом функции является команда пользователя.

Реализуется пользователем.

3.1.2.32 cncCustomRequestRepos

Синтаксис:	*void cncCustomRequestRepos (int request);*
Аргумент(ы):	*int request – идентификатор команды пользователя*
Файл объявления:	*include/cnc/cnc.h*

Функция обработки пользовательских команд в режиме возврата на контур УЧПУ. Аргументом функции является команда пользователя.

Реализуется пользователем.

3.1.2.33 cncManualCanChangeOverride

Синтаксис:	*int cncManualCanChangeOverride();*
Аргумент(ы):	*Нет*
Файл объявления:	*include/cnc/cnc.h*

Функция выполняет запрос на разрешение применения коррекции подачи.

Возвращает 1, если коррекция разрешена, и 0 в противном случае.

Реализуется пользователем.

3.1.2.34 shutdown

Синтаксис:	*void shutdown();*
Аргумент(ы):	*нет*
Файл объявления:	*sys/sys.h*

Функция вызывает выключение УЧПУ.
Является системной.

3.1.2.35 reset

Синтаксис:	*void reset();*
Аргумент(ы):	*нет*
Файл объявления:	*sys/sys.h*

Функция вызывает перезагрузку УЧПУ, которая эквивалентна выключению и последующему включению питания.
Является системной.

3.1.2.36 reinitialize

Синтаксис:	*void reinitialize();*
Аргумент(ы):	*нет*
Файл объявления:	*sys/sys.h*

Функция вызывает сброс параметров УЧПУ до заводских.
Является системной.

3.2 Управление станком

3.2.1 Типы данных

3.2.1.1 MTState

Тип данных:	*Перечисление MTState*
Файл объявления:	*include/cnc/mt.h*

Перечисление определяет идентификаторы состояний станка.

Таблица 3.10. Перечисление MTState

Идентификатор	Описание
mtNotReady	Станок выключен
mtStartOn	Начало включения
mtDriveOn	Включение приводов

Продолжение таблицы 3.10.

Идентификатор	Описание
mtWaitDriveOn	Ожидание включения приводов
mtOthersMotorOn	Включение вспомогательных моторов
mtReady	Станок включен
mtStartOff	Начало выключения
mtOthersMotorOff	Выключение вспомогательных моторов
mtAxisStop	Останов осей и шпинделя
mtAxisWaitStop	Ожидание останова осей и шпинделя
mtDriveOff	Выключение приводов
mtAbort	Аварийное торможение
mtPhaseRef	Фазировка
mtWaitPhaseRef	Ожидание фазировки
mtWaitOff	Ожидание выключения питания станка
mtWaitAbsPos	Ожидание данных от абсолютного ДОС

3.2.1.2 MTCNCRequests

Тип данных: *Перечисление MTCNCRequests*
Файл объявления: *include/cnc/mt.h*

Перечисление определяет идентификаторы команд управления станком.

Начальный номер блока пользовательских команд (mtcncCommandStart) равен 1000, конечный (mtcncCommandEnd) – 1999.

Начальный номер блока пользовательских команд движения (mtcncMoveCommandStart) равен 2000, конечный (mtcncMoveCommandEnd) – 2999.

Таблица 3.11. Перечисление MTCNCRequests

Идентификатор	Описание
mtcncNone	Нет команды
mtcncPowerOn	Включение станка
mtcncPowerOff	Выключение станка
mtcncEmergencyStop	Аварийный останов
mtcncReset	Сброс в начальное состояние
mtcncStart	Запуск операции в текущем режиме
mtcncStop	Останов операции в текущем режиме
mtcncCncOff	Выключение УЧПУ

Продолжение таблицы 3.11.

Идентификатор	Описание
mtcncActivateManual	Включение ручного режима
mtcncActivateHandwheel	Включение режима дискретных перемещений
mtcncActivateRef	Включение режима выезда в нулевую точку
mtcncActivateMDI	Включение режима преднабора
mtcncActivateAuto	Включение автоматического режима
mtcncActivateRepos	Включение режима возврата на контур
mtcncToggleStep	Покадровая отработка УП
mtcncToggleRepos	Возврат на контур
mtcncToggleVirtual	Отработка УП в виртуальном режиме
mtcncToggleOptionalSkip	Отработка УП с программным пропуском кадров
mtcncToggleOptionalStop	Отработка УП с опциональным остановом
mtcncSelectSpeed1	Выбор первой скорости/дискреты безразмерных/дискретных перемещений
mtcncSelectSpeed2	Выбор второй скорости/дискреты безразмерных/дискретных перемещений
mtcncSelectSpeed3	Выбор третьей скорости/дискреты безразмерных/дискретных перемещений
mtcncSelectSpeed4	Выбор четвёртой скорости/дискреты безразмерных/дискретных перемещений
mtcncSelectRapid	Перемещение на скорости быстрого хода
mtcncDryRun	Пробная подача
mtcncReducedRapid	Уменьшенная подача быстрого хода
mtcncMoveLock	Отработка УП с блокировкой движения
mtcncAlarmCancel	Сброс ошибок
mtcncCommandStart	Начальный номер блока пользовательских команд
mtcncCommandEnd	Конечный номер блока пользовательских команд
mtcncMoveCommandStart	Начальный номер блока пользовательских команд движения
mtcncMoveCommandEnd	Конечный номер блока пользовательских команд движения

3.2.1.3 MTDesc

Тип данных: *Структура MTDesc*
Файл объявления: *include/cnc/mt.h*

Структура определяет данные станка.

Таблица 3.12. Структура MTDesc

Элемент	Тип	Описание
State	int	Состояние автомата включения/выключения станка
IN	MTInputs	Входы плат входов
OUT	MTOutputs	Выходы плат реле
PultIn	PultInputs	Входы пульта оператора
PultOut	PultOutputs	Выходы пульта оператора
PortablePultIn	PortablePultInputs	Входы переносного пульта
timerState	**Timer**	Таймер состояния
timerReset	**Timer**	Таймер сброса
timerScan	**Timer**	Таймер выполнения операции
ncNotReadyReq	Битовое поле:1	Запрос готовности системы
ncFollowUpReq	Битовое поле:1	Запрос восстановления после ошибки
ncStopReq	Битовое поле:1	Запрос немедленного останова УП или движения
ncStopAtEndReq	Битовое поле:1	Запрос останова в конце текущего кадра

3.2.2 Функции

3.2.2.1 systemPlcActive

Синтаксис: *int systemPlcActive();*
Аргумент(ы): *Нет*
Файл объявления: *include/cnc/mt.h*

Функция возвращает 1, если нет ошибок программ ПЛК, и 0 в противном случае.
Реализуется пользователем.

3.2.2.2 hasEmergencyStopRequest

Синтаксис: *int hasEmergencyStopRequest();*
Аргумент(ы): *Нет*
Файл объявления: *include/cnc/mt.h*

Функция возвращает 1, если есть запрос аварийного останова, и 0 в противном случае.
Реализуется пользователем.

3.2.2.3 mtControlRequest

Синтаксис:	*void mtControlRequest();*
Аргумент(ы):	*Нет*
Файл объявления:	*include/cnc/mt.h*

Функция добавляет команды в очередь.
Реализуется пользователем.

3.2.2.4 mtUpdateCNCIndication

Синтаксис:	*void mtUpdateCNCIndication();*
Аргумент(ы):	*Нет*
Файл объявления:	*include/cnc/mt.h*

Функция обновляет индикацию пульта оператора.
Реализуется пользователем.

3.3 Обработка ошибок

3.3.1 Типы данных

3.3.1.1 DriveErrors

Тип данных:	*Объединение DriveErrors*
Файл объявления:	*include/cnc/errors.h*

Объединение определяет ошибки и режим работы сервоусилителя.

Таблица 3.13. Объединение DriveErrors

Элемент	Тип	Описание
struct {		
protocol	unsigned:1	Ошибка протокола
ampNotReady	unsigned:1	Нет готовности
ampFault	unsigned:1	Сервоусилитель в состоянии ошибки
i2tFault	unsigned:1	Ошибка i2t
crc	unsigned:1	Ошибка контрольной суммы
igbtFault	unsigned:1	Ошибка IGBT модуля
igbtTempFault	unsigned:1	Превышение температуры IGBT модуля
highDCFault	unsigned:1	Повышенное напряжение в ЗПТ
lowDCFault	unsigned:1	Пониженное напряжение в ЗПТ
linkFault	unsigned:1	Ошибка связи

Продолжение таблицы 3.13.

Элемент	Тип	Описание
brakeOnLowFault	unsigned:1	Сигнал на открытие тормозного транзистора в состоянии L (не в слежении)
brakeOnHighFault	unsigned:1	Сигнал на открытие тормозного транзистора в состоянии H (не в слежении)
brakeFault	unsigned:1	Недостаточная мощность тормозного резистора
currentOutFault	unsigned:1	Измеренный ток в фазе в отсечке
adcFault	unsigned:1	Ошибка АЦП
pwmShortFault	unsigned:1	Период сигнала ШИМ меньше 50 мкс
pwmLongFault	unsigned:1	Период сигнала ШИМ больше 400 мкс
reserved	unsigned:8	Резерв
ampState	unsigned:2	Состояние сервоусилителя
errorCode }	unsigned:4	Текущий код ошибки
errors	unsigned	Переменная, содержащая все битовые поля

Поле `ampState` является 2-битным и содержит коды состояния сервоусилителя:

- 0 – не подано высокое напряжение;
- 1 – подано высокое напряжение;
- 2 – сервоусилитель в слежении.

Поле `errorCode` является 4-битным и и содержит текущий код ошибки сервоусилителя:

- 0 – нет ошибок;
- 1 – ошибка IGBT модуля;
- 2 – превышение температуры IGBT модуля;
- 3 – повышенное напряжение в ЗПТ;
- 4 – пониженное напряжение в ЗПТ;
- 5 – ошибка связи;
- 6 – сигнал на открытие тормозного транзистора в состоянии L (не в слежении);
- 7 – сигнал на открытие тормозного транзистора в состоянии H (не в слежении);
- 8 – недостаточная мощность тормозного резистора;
- 9 – измеренный ток в фазе в отсечке;
- 10 – ошибка АЦП;
- 11 – период сигнала ШИМ меньше 50 мкс;
- 12 – Период сигнала ШИМ больше 400 мкс.

3.3.1.2 EncoderErrors

Тип данных: *Объединение EncoderErrors*
Файл объявления: *include/cnc/errors.h*

Объединение определяет ошибки ДОС.

Таблица 3.14. Объединение EncoderErrors

Элемент	Тип	Описание
struct {		
encFault	unsigned:1	Комбинированная ошибка датчика
decode	unsigned:1	Ошибка декодирования
sumOfSqr	unsigned:1	Неверная сумма квадратов каналов синусно-косинусного датчика
faultN	unsigned:1	Сигнал FAULT_N
adc	unsigned:1	Ошибка АЦП
lineA	unsigned:1	Ошибка канала A
lineB	unsigned:1	Ошибка канала B
lineC	unsigned:1	Ошибка канала C
power	unsigned:1	Ошибка питания
serialDataNotReady	unsigned:1	Ошибка последовательного ДОС
warningBiSS	unsigned:1	Предупреждение ДОС BiSS
faultBiSS	unsigned:1	Ошибка ДОС BiSS
statusEnDat }	unsigned:1	Ошибка статуса ДОС с протоколом EnDat
errors	unsigned	Переменная, содержащая все битовые поля

3.3.1.3 IOErrors

Тип данных: *Объединение IOErrors*
Файл объявления: *include/cnc/errors.h*

Объединение определяет ошибки последовательного интерфейса плат входов/выходов.

Таблица 3.15. Объединение IOErrors

Элемент	Тип	Описание
struct {		
parity	unsigned:1	Ошибка четности
protocol	unsigned:1	Ошибка протокола
crc	unsigned:1	Ошибка контрольной суммы
watchdog }	unsigned:1	Срабатывание сторожевого таймера
errors	unsigned	Переменная, содержащая все битовые поля

3.3.1.4 MotorErrors

Тип данных: *Объединение MotorErrors*
Файл объявления: *include/cnc/errors.h*

Объединение определяет ошибки приводов.

Таблица 3.16. Объединение MotorErrors

Элемент	Тип	Описание
struct {		
phaseref	unsigned:1	Не выполнена фазировка
home	unsigned:1	Не выполнен поиск нулевой точки
homeError	unsigned:1	Произошла ошибка при поиске нулевой точки
openLoop	unsigned:1	Двигатель не в слежении
encoder	unsigned:1	Ошибка ДОС
plusLimit	unsigned:1	Срабатывание аппаратного ограничителя в положительном направлении
minusLimit	unsigned:1	Срабатывание аппаратного ограничителя в отрицательном направлении
swPlusLimit	unsigned:1	Срабатывание программного ограничителя в положительном направлении
swMinusLimit	unsigned:1	Срабатывание программного ограничителя в отрицательном направлении
folError	unsigned:1	Критическая ошибка слежения
folErrorWarning	unsigned:1	Предупредительная ошибка слежения
temperature	unsigned:1	Перегрев двигателя

Продолжение таблицы 3.16.

Элемент	Тип	Описание
tempWarning	unsigned:1	Предупреждение о перегреве двигателя
auxFault	unsigned:1	Внешняя ошибка
pos2Error	unsigned:1	Ошибка рассогласования датчиков положения и скорости
pos2Warning	unsigned:1	Предупреждение рассогласования датчиков положения и скорости
phasePosError	unsigned:1	Ошибка рассогласования датчиков положения и коммутации
phasePosWarning }	unsigned:1	Предупреждение рассогласования датчиков положения и коммутации
errors	unsigned	Переменная, содержащая все битовые поля

3.3.1.5 AxisErrors

Тип данных: *Объединение AxisErrors*
Файл объявления: *include/cnc/errors.h*

Объединение определяет ошибки оси.

Таблица 3.17. Объединение AxisErrors

Элемент	Тип	Описание
struct {		
abortTimeout	unsigned:1	Истекло время операции аварийного торможения
activateTimeout	unsigned:1	Истекло время операции включения в слежение
phaseRefTimeout	unsigned:1	Истекло время операции фазировки
deactivateTimeout }	unsigned:1	Истекло время операции выключения
errors	unsigned	Переменная, содержащая все битовые поля

3.3.1.6 SpindleErrors

Тип данных: *Объединение SpindleErrors*
Файл объявления: *include/cnc/errors.h*

Объединение определяет ошибки шпинделя.

Таблица 3.18. Объединение SpindleErrors

Элемент	Тип	Описание
struct {		
abortTimeout	unsigned:1	Истекло время операции аварийного торможения
activateTimeout	unsigned:1	Истекло время операции включения в слежение
phaseRefTimeout	unsigned:1	Истекло время операции фазировки
deactivateTimeout	unsigned:1	Истекло время операции выключения
speedTimeout	unsigned:1	Истекло время выхода на заданную скорость
stopTimeout	unsigned:1	Истекло время операции останова
homeTimeout	unsigned:1	Истекло время операции поиска нулевой точки
positionTimeout }	unsigned:1	Истекло время выхода в заданное положение
errors	unsigned	Переменная, содержащая все битовые поля

3.3.1.7 ChannelErrors

Тип данных: *Объединение ChannelErrors*
Файл объявления: *include/cnc/errors.h*

Объединение определяет ошибки канала управления.

Таблица 3.19. Объединение ChannelErrors

Элемент	Тип	Описание
struct {		
phaseRefTimeout	unsigned:1	Истекло время операции фазировки
driveOnTimeout	unsigned:1	Истекло время ожидания включения сервоусилителя

Продолжение таблицы 3.19.

Элемент	Тип	Описание
driveOffTimeout	unsigned:1	Истекло время ожидания выключения сервоусилителя
abortTimeout	unsigned:1	Истекло время операции аварийного торможения
stopTimeout	unsigned:1	Истекло время операции останова
homeTimeout	unsigned:1	Истекло время операции поиска нулевой точки
homeError	unsigned:1	Ошибка поиска нулевой точки
startWithoutHome	unsigned:1	Попытка запуска без определения нулевой точки
cannotStart	unsigned:1	Ошибка запуска программы
progStopOk	unsigned:1	УП выполнена
progStopAbort	unsigned:1	УП прервана
progStopSyncError	unsigned:1	Ошибка присвоения в буфере синхронных переменных
progStopBufferError	unsigned:1	Ошибка в буфере программы движения
progStopCCMove	unsigned:1	Неверный кадр в режиме коррекции инструмента
progStopLinToPvt	unsigned:1	Ошибка при преобразовании линейного движения в сплайн или pvt-движение
progStopCCLeadOut	unsigned:1	Неверный кадр при отмене режима коррекции инструмента
progStopCCLeadIn	unsigned:1	Неверный кадр при активации режима коррекции инструмента
progStopCCBufSize	unsigned:1	Недостаточный размер буфера в режиме коррекции инструмента
progStopPvt	unsigned:1	Ошибка расчёта pvt-движения
progStopCCFeed	unsigned:1	Неверное указание подачи в режиме коррекции инструмента
progStopCCDir	unsigned:1	Смена направления движения в режиме коррекции инструмента
progStopNoSolve	unsigned:1	Невозможно рассчитать движение в режиме коррекции инструмента
progStopCC3NdotT	unsigned:1	Ошибка расчёта точки резания в режиме трёхмерной коррекции инструмента
progStopCCDist	unsigned:1	Невозможно предотвратить «перерез» в режиме коррекции инструмента

Продолжение таблицы 3.19.

Элемент	Тип	Описание
progStopCCNoIntersect	unsigned:1	Невозможно найти пересечение траекторий в режиме коррекции инструмента
progStopCCNoMoves	unsigned:1	Между активацией и отменой режима коррекции инструмента кадры без команд движения
progStopRunTime	unsigned:1	Недостаточное время для расчёта движения
progStopInPos	unsigned:1	Истекло время ожидания состояния «в позиции»
progStopSoftLimit	unsigned:1	Срабатывание программного ограничения
progStopRadiusX	unsigned:1	Срабатывание ограничения величины радиальной ошибки в режиме кругового движения
progStopRadiusXX	unsigned:1	Срабатывание ограничения величины радиальной ошибки в режиме кругового движения в расширенной системе координат
progPausedM00	unsigned:1	УП временно остановлена по команде M00 или M01
cycleInvalidArgs	unsigned:1	Неверные аргументы функции постоянного цикла
seekingBlock	unsigned:1	Поиск кадра
seekBlockFound	unsigned:1	Кадр найден
seekBlockNotFound }	unsigned:1	Кадр не найден
errors	unsigned	Переменная, содержащая все битовые поля

3.3.1.8 NCErrors

Тип данных: *Объединение NCErrors*
Файл объявления: *include/cnc/errors.h*

Объединение определяет системные ошибки.

Таблица 3.20. Объединение NCErrors

Элемент	Тип	Описание
struct { factory	unsigned:1	Ошибка загрузки системных параметров, используются параметры по умолчанию
userFactory	unsigned:1	Ошибка загрузки параметров пользователя, пользовательские переменные не определены
swClock	unsigned:1	Отсутствует аппаратный источник частоты
bgWdt	unsigned:1	Срабатывание сторожевого таймера фонового режима
rtWdt	unsigned:1	Срабатывание сторожевого таймера реального времени
sysPlcFault	unsigned:1	Ошибка выполнения системных программ ПЛК
hmiWatchdog }	unsigned:1	Срабатывание сторожевого таймера связи с пультом оператора
errors	unsigned	Переменная, содержащая все битовые поля

3.3.1.9 Errors

Тип данных: *Структура Errors*
Файл объявления: *include/cnc/errors.h*

Структура содержит данные о системных ошибках, об ошибках станка, каналов управления, осей, шпинделей, приводов, ДОС, сервоусилителей и плат входов/выходов.

Таблица 3.21. Структура Errors

Элемент	Тип	Описание
machine	MachineErrors	Ошибки станка
nc	**NCErrors**	Системные ошибки
channel [ЧИСЛО_КАНАЛОВ]	**ChannelErrors**	Ошибки каналов управления
axes [ЧИСЛО_ОСЕЙ]	**AxisErrors**	Ошибки осей
spindles [ЧИСЛО_ШПИНДЕЛЕЙ]	**SpindleErrors**	Ошибки шпинделей
motors [ЧИСЛО_ДВИГАТЕЛЕЙ+1]	**MotorErrors**	Ошибки приводов

Продолжение таблицы 3.21.

Элемент	Тип	Описание
encoders [ЧИСЛО_ДОС]	**EncoderErrors**	Ошибки ДОС
drive [ЧИСЛО_ДВИГАТЕЛЕЙ+1]	**DriveErrors**	Ошибки сервоусилителей
io [ЧИСЛО_ПЛАТ_ВХ/ВЫХ]	**IOErrors**	Ошибки плат входов/выходов

3.3.1.10 ErrorReaction

Тип данных: *Перечисление ErrorReaction*
Файл объявления: *include/cnc/errors.h*

Перечисление определяет идентификаторы типов реакций на ошибки.

Таблица 3.22. Перечисление ErrorReaction

Идентификатор	Описание
reactNone	Нет реакции
reactFollowUp	Восстановление после ошибки
reactStopProgram	Прервано выполнение программы
reactNCNotReady	Нет готовности системы
reactChannelNotReady	Нет готовности канала
reactStartDisable	Запрет запуска программы в канале
reactNeedHome	Необходим повторный поиск нулевой точки для осей в канале
reactShowAlarm	Показать сообщение об ошибке
reactStop	Останов осей
reactStopAtEnd	Останов осей в конце блока
reactAutoOnly	Фиксация ошибки только в автоматическом режиме
reactWarning	Показать предупреждение

3.3.1.11 ErrorClear

Тип данных: *Перечисление ErrorClear*
Файл объявления: *include/cnc/errors.h*

Перечисление определяет идентификаторы типов сброса ошибок.

Самый низкий приоритет имеет автоматический сброс (`clearSelf`), самый высокий приоритет – сброс по включению питания (`clearPowerOn`).

Таблица 3.23. Перечисление ErrorClear

Идентификатор	Описание
clearSelf	Автоматический сброс
clearCancel	Сброс из оболочки, отменой текущего режима работы или перезапуском УП
clearNCStart	Сброс отменой текущего режима работы или перезапуском УП
clearReset	Сброс отменой текущего режима работы
clearNCReset	Сброс перезагрузкой системы
clearPowerOn	Сброс по включению питания

3.3.1.12 DriveErrorReaction

Тип данных: *Перечисление DriveErrorReaction*
Файл объявления: *include/cnc/errors.h*

Перечисление определяет идентификаторы типов реакции на ошибки сервоусилителя.

Таблица 3.24. Перечисление DriveErrorReaction

Идентификатор	Описание
dreactNone	Нет реакции
dreactOFF1	Останов и выключение с задержкой в режиме слежения, иначе выключение
dreactOFF1delayed	Пауза, останов и выключение в режиме слежения, иначе пауза и выключение
dreactOFF2	Выключение
dreactOFF3	Аварийное торможение и выключение с задержкой в режиме слежения, иначе выключение
dreactSTOP2	Аварийное торможение и сохранение режима слежения
dreactIASC_DCBRK	Для синхронного - закоротить обмотки, для асинхронного - торможение постоянным током
dreactENC	Настраивается (по умолчанию dreactOFF2)

3.3.1.13 ErrorDescription

Тип данных: *Структура ErrorDescription*
Файл объявления: *include/cnc/errors.h*

Структура определяет параметры описания ошибки.

Таблица 3.25. Структура ErrorDescription

Элемент	Тип	Описание
id	unsigned	Номер ошибки в категории
reaction	unsigned	Тип реакции
clear	unsigned	Тип сброса

3.3.1.14 ErrorRequests

Тип данных: *Структура ErrorRequests*
Файл объявления: *include/cnc/errors.cfg*

Структура определяет флаги действий системы, которые вызываются согласно реакциям на ошибки в перечислении **ErrorReaction**.

Таблица 3.26. Структура ErrorDescription

Элемент	Тип	Описание
ncNotReady	int:1	Нет готовности системы
ncStop	int:1	Останов осей
ncStopAtEnd	int:1	Останов осей в конце блока
ncFollowUp	int:1	Восстановление после ошибки
channelNotReady	int:1	Нет готовности канала
startDisable	int:1	Запрет запуска программы в канале
needHome	int:1	Необходим повторный поиск нулевой точки для осей в канале

3.3.2 Функции и макросы

3.3.2.1 errorSetScan

Синтаксис:	*int errorSetScan (unsigned curInput, unsigned input, const ErrorDescription &desc, ErrorClear request);*
Аргумент(ы):	*unsigned curInput – флаг ошибки,* *unsigned input – состояние соответствующего входа ошибки,* *const **ErrorDescription** &desc – описание ошибки* ***ErrorClear** request – идентификатор запроса на сброс ошибки*
Файл объявления:	*include/cnc/errors.h*

Функция возвращает 1 (наличие ошибки), если состояние соответствующего входа ошибки отлично от 0.

Если состояние соответствующего входа равно 0 и уровень сброса ошибки в структуре описания ошибки меньше или равен значению идентификатора запроса на сброс ошибки, то функция возвращает 0 (ошибка сброшена).

Если состояние соответствующего входа равно 0 и уровень сброса ошибки в структуре описания ошибки больше значения идентификатора запроса на сброс ошибки, то функция возвращает текущее значение флага ошибки.

Является системной.

3.3.2.2 errorScanSet

Синтаксис:	*errorScanSet (error, input, desc, request)*
Аргумент(ы):	*error – флаг ошибки,* *input – состояние соответствующего входа,* *desc – описание ошибки (переменная типа **ErrorDescription**),* *request – идентификатор запроса на сброс ошибки (переменная типа **ErrorClear**)*
Файл объявления:	*include/cnc/errors.h*

Макрос **errorScanSet** вызывает функцию **errorSetScan** и присваивает возвращаемое значение аргументу `error` (флагу ошибки).

Макрос обновляет флаг выбранной ошибки в зависимости от состояния соответствующего входа и заданного идентификатора запроса на сброс ошибки.

Является системной.

3.3.2.3 errorScanRequest

Синтаксис:	*void errorScanRequest (ErrorClear request);*
Аргумент(ы):	***ErrorClear** request – идентификатор запроса на сброс ошибки*
Файл объявления:	*include/cnc/errors.h*

Функция выполняет вызовы макроса **errorScanSet** для обновления флагов ошибок станка, УЧПУ, каналов управления, осей, шпинделей, приводов, сервоусилителей, ДОС и последовательного интерфейса плат входов/выходов.

Является системной.

3.3.2.4 errorScan

Синтаксис:	*void errorScan();*
Аргумент(ы):	*Нет*
Файл объявления:	*include/cnc/errors.h*

Функция выполняет вызов **errorScanRequest** с аргументом `clearSelf` (см. **ErrorClear**) для обновления флагов ошибок с запросом автоматического сброса ошибок.

Является системной.

3.3.2.5 errorReaction

Синтаксис:	*void errorReaction(unsigned input, const ErrorDescription &desc);*
Аргумент(ы):	*unsigned input – флаг ошибки,* *const **ErrorDescription** &desc – описание ошибки*
Файл объявления:	*include/cnc/errors.h*

Устанавливает флаги действий системы (см. **ErrorRequests**) согласно реакциям на возникшую ошибку.

Является системной.

3.3.2.6 errorsMachineScan

Синтаксис:	*void errorsMachineScan (int request);*
Аргумент(ы):	*int request – идентификатор запроса на сброс ошибки*
Файл объявления:	*include/cnc/errors.h*

Функция выполняет вызовы макроса **errorScanSet** для обновления флагов ошибок станка. Вызывается из **errorScanRequest**.

Реализуется пользователем.

3.3.2.7 errorsMachineReaction

Синтаксис:	*void errorsMachineReaction();*
Аргумент(ы):	*Нет*
Файл объявления:	*include/cnc/errors.h*

Функция выполняет вызовы функции **errorReaction** для ошибок станка.

Реализуется пользователем.

3.3.2.8 encoderScanErrors

Синтаксис:	*void encoderScanErrors(ErrorClear request);*
Аргумент(ы):	***ErrorClear** request – идентификатор типа сброса ошибки*
Файл объявления:	*include/func/enc.h*

Функция выполняет вызовы макроса **errorScanSet** для обновления флагов ошибок ДОС. Вызывается из **errorScanRequest**.

Является системной.

3.3.2.9 encoderErrorsReaction

Синтаксис:	*void encoderErrorsReaction();*
Аргумент(ы):	*Нет*
Файл объявления:	*include/func/enc.h*

Функция выполняет вызовы функции **errorReaction** для ошибок ДОС.

Является системной.

3.3.2.10 ampScanErrors

Синтаксис:	*void ampScanErrors(int motor, int servo, int chan, ErrorClear request, int isaxis, int id);*
Аргумент(ы):	*int motor – номер связанного с осью двигателя,* *int servo – номер платы управления,* *int chan – номер канала,* ***ErrorClear** request – идентификатор типа сброса ошибки,* *int isaxis – флаг оси (1) или шпинделя (0),* *int id – номер оси или шпинделя*
Файл объявления:	*include/func/amp_fault.h*

Функция выполняет вызовы макроса **errorScanSet** для обновления флагов ошибок сервоусилителей. Вызывается из **errorScanRequest**.

Является системной.

3.3.2.11 ampErrorsReaction

Синтаксис:	*void ampErrorsReaction(int motor);*
Аргумент(ы):	*int motor – номер связанного с осью двигателя*
Файл объявления:	*include/func/amp_fault.h*

Функция выполняет вызовы функции **errorReaction** для ошибок сервоусилителей.

Является системной.

3.3.2.12 ioScanErrors

Синтаксис:	*void ioScanErrors(int ioNum, int servo, int io, ErrorClear request)*
Аргумент(ы):	*int ioNum – номер платы входов/выходов,* *int servo – номер платы управления,* *int io – номер входа/выхода,* ***ErrorClear** request – идентификатор типа сброса ошибки*
Файл объявления:	*include/func/io.h*

Функция выполняет вызовы макроса **errorScanSet** для обновления флагов ошибок последовательного интерфейса плат входов/выходов. Вызывается из **errorScanRequest**.

Является системной.

3.3.2.13 ioErrorsReaction

Синтаксис:	*void ioErrorsReaction(int ioNum, int io, int servo);*
Аргумент(ы):	*int ioNum – номер платы входов/выходов,* *int io – номер входа/выхода,* *int servo – номер платы управления*
Файл объявления:	*include/func/io.h*

Функция выполняет вызовы функции **errorReaction** для ошибок последовательного интерфейса плат входов/выходов.

Является системной.

3.4 Управление осями

3.4.1 Типы данных

3.4.1.1 AxisStates

Тип данных:	*Перечисление AxisStates*
Файл объявления:	*include/func/axis.h*

Перечисление определяет идентификаторы состояний оси.

Таблица 3.27. Перечисление AxisStates

Идентификатор	Описание
axisInactive	Ось выключена
axisActive	Ось находится в слежении
axisJoggingPlus	Толчковое движение в положительном направлении
axisJoggingMinus	Толчковое движение в отрицательном направлении
axisJoggingTo	Толчковое движение в заданное положение или на заданное расстояние
axisStopping	Останов оси
axisHomeWaitHW	Ожидание применения аппаратных настроек выезда в нулевую точку
axisHoming	Выезд в нулевую точку
axisIndexWaitHW	Ожидание применения аппаратных настроек поиска индексной метки
axisIndexing	Поиск индексной метки
axisAborting	Аварийное торможение
axisWaitActivate	Ожидание включения

Продолжение таблицы 3.27.

Идентификатор	Описание
axisWaitDeactivate	Ожидание выключения
axisWaitPhaseRef	Ожидание фазировки

3.4.1.2 AxisCommands

Тип данных: *Перечисление AxisCommands*
Файл объявления: *include/func/axis.h*

Перечисление определяет идентификаторы команд управления осями.

Таблица 3.28. Перечисление AxisCommands

Идентификатор	Описание
axisCmdIdle	Нет команды
axisCmdKill	Выключить ось
axisCmdActivate	Включить ось в слежение
axisCmdDeactivate	Выключить ось
axisCmdJogPlus	Выполнить толчковое движение в положительном направлении
axisCmdJogMinus	Выполнить толчковое движение в отрицательном направлении
axisCmdJogStop	Выполнить останов
axisCmdJogRet	Вернуться в сохраненную позицию
axisCmdInc	Выполнить толчковое движение на заданное расстояние
axisCmdAbs	Выполнить толчковое движение в заданное положение
axisCmdHome	Выполнить движение в нулевую точку
axisCmdIndex	Выполнить движение до индексной метки
axisCmdPhaseRef	Выполнить фазировку
axisCmdAbort	Выполнить аварийное выключение

3.4.1.3 AxisAbortMode

Тип данных: *Перечисление AxisAbortMode*
Файл объявления: *include/func/axis.h*

Перечисление определяет идентификаторы действий по команде аварийного останова (ABORT).

Таблица 3.29. Перечисление AxisAbortMode

Идентификатор	Описание
axisAbortStop	Останов категории 2 (аварийно затормозить и оставаться в слежении)
axisAbortAndKill	Останов категории 1 (аварийно затормозить и выключить)
axisAbortKill	Останов категории 0 (выключить)

3.4.1.4 AxisConfig

Тип данных: *Структура AxisConfig*
Файл объявления: *include/func/axis.h*

Структура определяет настройки оси.

Таблица 3.30. Структура AxisConfig

Элемент	Тип	Описание
servo	unsigned	Номер платы управления ($0 \div 3$)
chan	unsigned	Номер канала ($0 \div 7$)
motor	unsigned	Номер связанного с осью двигателя ($0 \div 31$)
homeOrder	unsigned	Порядок выезда в нулевую точку
needDKill	unsigned:1	Требуется задержка перед отключением
needPhaseRef	unsigned:1	Требуется фазировка
needHome	unsigned:1	Требуется выезд в нулевую точку
needIndex	unsigned:1	Требуется позиционирование по индексной метке
hasAbsPos	unsigned:1	Установлен абсолютный датчик

Продолжение таблицы 3.30.

Элемент	Тип	Описание
abortMode	unsigned:2	Реакции на команду аварийного выключения (см. **AxisAbortMode**)
homeCaptCtrl	unsigned:4	Настройка захвата положения для выезда в нулевую точку по входу (флагу)
indexCaptCtrl	unsigned:4	Настройка захвата положения для выезда в нулевую точку по индексной метке ДОС
needPosRef	unsigned:1	Требуется позиционирование при включении станка
refAxis	unsigned:5	Координата для оси
rotaryAxis	unsigned:1	Вращающаяся ось с периодом 360
reserved	unsigned:10	Резерв
homeVel	double	Скорость и направление выезда в нулевую точку
indexVel	double	Скорость и направление поиска индексной метки
homeOffset	double	Смещение нулевой точки относительно позиции ДОС
indexOffset	double	Смещение индексной метки относительно позиции ДОС
homeOfsVel	double	Скорость движения в позицию смещения нулевой точки (не используется)
indexOfsVel	double	Скорость движения в позицию смещения индексной метки (не используется)
minPos	double	Программное ограничение в отрицательном направлении (для абсолютного ДОС настраивается в дискретах датчика, определённых в энкодерной таблице)
maxPos	double	Программное ограничение в положительном направлении (для абсолютного ДОС настраивается в дискретах датчика, определённых в энкодерной таблице)
defaultTa	double	Время в мс ускорения/замедления (при значении больше 0) или коэффициент, обратный величине амплитуды ускорения/замедления (при значении меньше 0) по умолчанию

Продолжение таблицы 3.30.

Элемент	Тип	Описание
defaultTs	double	Время в мс (при значении больше 0) или коэффициент, обратный значению амплитуды рывка (при значении меньше 0), для каждой половины S-кривой профиля ускорения по умолчанию
manualTa	double	Время в мс ускорения/замедления (при значении больше 0) или коэффициент, обратный величине амплитуды ускорения/замедления (при значении меньше 0) в ручном режиме
manualTs	double	Время в мс (при значении больше 0) или коэффициент, обратный значению амплитуды рывка (при значении меньше 0), для каждой половины S-кривой профиля ускорения в ручном режиме
hwlTa	double	Время в мс ускорения/замедления (при значении больше 0) или коэффициент, обратный величине амплитуды ускорения/замедления (при значении меньше 0) в режиме дискретных перемещений
hwlTs	double	Время в мс (при значении больше 0) или коэффициент, обратный значению амплитуды рывка (при значении меньше 0), для каждой половины S-кривой профиля ускорения в режиме дискретных перемещений
homeTa	double	Время в мс ускорения/замедления (при значении больше 0) или коэффициент, обратный величине амплитуды ускорения/замедления (при значении меньше 0) в режиме выезда в нулевую точку
homeTs	double	Время в мс (при значении больше 0) или коэффициент, обратный значению амплитуды рывка (при значении меньше 0), для каждой половины S-кривой профиля ускорения в режиме выезда в нулевую точку

Продолжение таблицы 3.30.

Элемент	Тип	Описание
autoTa	double	Время в мс ускорения/замедления (при значении больше 0) или коэффициент, обратный величине амплитуды ускорения/замедления (при значении меньше 0) в автоматическом режиме
autoTs	double	Время в мс (при значении больше 0) или коэффициент, обратный значению амплитуды рывка (при значении меньше 0), для каждой половины S-кривой профиля ускорения в автоматическом режиме
encRes	double	Число дискрет датчика на оборот

Поля `homeCaptCtrl` и `indexCaptCtrl` являются 4-битными и содержат настройки захвата положения для выезда в нулевую точку:

- биты 0 и 1 определяют тип захвата положения (0 – непосредственный захват, 1 – по индексному сигналу датчика, 2 – захват по флагу, 3 – по флагу и индексному сигналу);
- бит 2 управляет инверсией индексного сигнала ДОС (0 – не инвертировать, 1 – инвертировать);
- бит 3 управляет инверсией флага при захвате положения (0 – не инвертировать, 1 – инвертировать).

3.4.1.5 Axis

Тип данных: *Структура Axis*
Файл объявления: *include/func/axis.h*

Структура определяет состояние, параметры и данные оси.

Таблица 3.31. Структура Axis

Элемент	Тип	Описание
state	**AxisStates**	Текущее состояние
command	**AxisCommands**	Текущая команда
statePreHome	**AxisStates**	Состояние перед выездом в нулевую точку

Продолжение таблицы 3.31.

Элемент	Тип	Описание
followup	unsigned:1	Восстановление после ошибки
phaseRefComplete	unsigned:1	Фазировка выполнена
phaseRefError	unsigned:1	Ошибка фазировки
homeComplete	unsigned:1	Выполнен выезд в нулевую точку
homeErrorFlag	unsigned:1	Ошибка выезда в нулевую точку
posRefComplete	unsigned:1	Позиционирование при включении станка выполнено
timer	**Timer**	Таймер
JogValue	double	Значение заданной позиции для толчкового перемещения
IncStep	double	Значение заданного расстояния для толчкового перемещения
platform	AxisPlatformControl	Пользовательские параметры и переменные оси

Структура `AxisPlatformControl` является пользовательской и служит для введения дополнительных параметров и переменных оси.

Если структура `AxisPlatformControl` задана пользователем, то должен быть определён идентификатор `AXES_PLATFORM_CONTROL_DEFINED`: `#define AXES_PLATFORM_CONTROL_DEFINED`.

3.4.1.6 AxesControl

Тип данных: *Структура AxesControl*
Файл объявления: *include/func/axis.h*

Структура определяет состояние, параметры и данные осей.

Таблица 3.32. Структура AxesControl

Элемент	Тип	Описание
homeState	**HomeStates**	Состояние выезда в нулевую точку
homeComplete	unsigned:1	Выполнен выезд в нулевую точку
homeErrorFlag	unsigned:1	Ошибка выезда в нулевую точку
homeStage	int	Этап выезда в нулевую точку
axis[ЧИСЛО_ОСЕЙ]	**Axis**	Данные осей

Продолжение таблицы 3.32.

Элемент	Тип	Описание
timerHome	**Timer**	Таймер для задержек переключений состояний в режиме выезда в нулевую точку
platform	AxesPlatformControl	Пользовательские параметры и переменные осей
saveSpeed	int	Сохранённая скорость с пульта оператора
activeAxis	int	Номер активной оси

Структура `AxesPlatformControl` является пользовательской и служит для введения дополнительных параметров и переменных осей.

Если структура `AxesPlatformControl` задана пользователем, то должен быть определён идентификатор `AXES_PLATFORM_CONTROL_DEFINED`: `#define AXES_PLATFORM_CONTROL_DEFINED`.

3.4.2 Функции

3.4.2.1 axesForceKill

Синтаксис: *void axesForceKill();*
Аргумент(ы): *Нет*
Файл объявления: *include/func/axis.h*

Функция вызывает принудительное выключение всех осей.

Является системной.

3.4.2.2 axisForceKill

Синтаксис: *void axisForceKill(unsigned axis);*
Аргумент(ы): *unsigned axis – номер оси*
Файл объявления: *include/func/axis.h*

Функция вызывает принудительное выключение оси, номер которой является аргументом функции.

Является системной.

3.4.2.3 axesDeactivate

Синтаксис: *void axesDeactivate();*
Аргумент(ы): *Нет*
Файл объявления: *include/func/axis.h*

Функция вызывает выключение всех осей.

Является системной.

3.4.2.4 axesActivate

Синтаксис:	*void axesActivate();*
Аргумент(ы):	*Нет*
Файл объявления:	*include/func/axis.h*

Функция вызывает включение в слежение всех осей.
Является системной.

3.4.2.5 axesInactive

Синтаксис:	*int axesInactive();*
Аргумент(ы):	*Нет*
Файл объявления:	*include/func/axis.h*

Функция возвращает 1, если хотя бы одна ось не находится в слежении, и 0 в противном случае.
Является системной.

3.4.2.6 axesActive

Синтаксис:	*int axesActive();*
Аргумент(ы):	*Нет*
Файл объявления:	*include/func/axis.h*

Функция возвращает 1, если все оси находятся в слежении, и 0 в противном случае.
Является системной.

3.4.2.7 axesPhaseRefComplete

Синтаксис:	*int axesPhaseRefComplete();*
Аргумент(ы):	*Нет*
Файл объявления:	*include/func/axis.h*

Функция возвращает 1, если фазировка выполнена для всех осей, и 0 в противном случае.
Является системной.

3.4.2.8 axesPhaseRef

Синтаксис:	*int axesPhaseRef();*
Аргумент(ы):	*Нет*
Файл объявления:	*include/func/axis.h*

Функция возвращает 1, если хотя бы одна ось требует фазировки, и 0 в противном случае. Для оси, фазировка которой не выполнена, даётся команда фазировки.
Является системной.

3.4.2.9 axesAborted

Синтаксис:	*int axesAborted();*
Аргумент(ы):	*Нет*
Файл объявления:	*include/func/axis.h*

Функция возвращает 1, если все оси аварийно остановлены, и 0 в противном случае.

Является системной.

3.4.2.10 axisStopped

Синтаксис:	*int axisStopped(unsigned axis);*
Аргумент(ы):	*unsigned axis – номер оси*
Файл объявления:	*include/func/axis.h*

Функция возвращает 1, если ось остановлена (ось в слежении имеет равную нулю заданную скорость и находится в позиции), и 0 в противном случае.

Является системной.

3.4.2.11 axesStopped

Синтаксис:	*int axesStopped();*
Аргумент(ы):	*Нет*
Файл объявления:	*include/func/axis.h*

Функция возвращает 1, если все оси остановлены (оси в слежении имеют равную нулю заданную скорость и находятся в позиции), и 0 в противном случае.

Является системной.

3.4.2.12 axesAbortAll

Синтаксис:	*void axesAbortAll();*
Аргумент(ы):	*Нет*
Файл объявления:	*include/func/axis.h*

Функция вызывает аварийное выключение всех осей.

Является системной.

3.4.2.13 axesStopAll

Синтаксис:	*void axesStopAll();*
Аргумент(ы):	*Нет*
Файл объявления:	*include/func/axis.h*

Функция вызывает останов всех осей при толчковых перемещениях.

Является системной.

3.4.2.14 axesRet

Синтаксис:	*void axesRet();*
Аргумент(ы):	*Нет*
Файл объявления:	*include/func/axis.h*

Функция вызывает перемещение осей в сохраненную позицию при толчковых перемещениях.

Является системной.

3.4.2.15 axisIndexInit

Синтаксис:	*void axisIndexInit(unsigned axis);*
Аргумент(ы):	*unsigned axis – номер оси*
Файл объявления:	*include/func/axis.h*

Функция выполняет инициализацию параметров поиска индексной метки для оси, номер которой является аргументом функции.

Является системной.

3.4.2.16 axisPosition

Синтаксис:	*double axisPosition(unsigned axis);*
Аргумент(ы):	*unsigned axis – номер оси*
Файл объявления:	*include/func/axis.h*

Функция возвращает заданную позицию оси, номер которой является аргументом функции.

Возвращаемое значение измеряется в единицах `encRes` (см. структуру **AxisConfig**). Для вращающихся осей возвращаемое значение – остаток от деления на 360.

Является системной.

3.4.2.17 axesFollowup

Синтаксис:	*void axesFollowup();*
Аргумент(ы):	*Нет*
Файл объявления:	*include/func/axis.h*

Функция устанавливает для всех осей флаг «Восстановление после ошибки» (см. структуру **Axis**).

Является системной.

3.4.2.18 initAxis

Синтаксис:	*void initAxis(int axis);*
Аргумент(ы):	*int axis – номер оси*
Файл объявления:	*include/func/axis.h*

Функция выполняет инициализацию оси, номер которой является аргументом функции, параметрами по умолчанию.

Является системной.

3.4.2.19 initAxes

Синтаксис:	*void initAxes();*
Аргумент(ы):	*Нет*
Файл объявления:	*include/func/axis.h*

Функция выполняет инициализацию осей параметрами по умолчанию.

Является системной.

3.4.2.20 axisInitPlatform

Синтаксис:	*void axisInitPlatform(int axis);*
Аргумент(ы):	*int axis – номер оси*
Файл объявления:	*include/func/axis.h*

Функция выполняет инициализацию параметров оси, номер которой является аргументом функции, пользовательскими значениями, в том числе структуры AxisPlatformControl.

Реализуется пользователем.

3.4.2.21 axesInitPlatform

Синтаксис:	*void axesInitPlatform();*
Аргумент(ы):	*Нет*
Файл объявления:	*include/func/axis.h*

Функция выполняет инициализацию параметров осей пользовательскими значениями, в том числе структуры AxesPlatformControl.

Реализуется пользователем.

3.4.2.22 axesAbsPosRead

Синтаксис:	*void axesAbsPosRead();*
Аргумент(ы):	*Нет*
Файл объявления:	*include/func/axis.h*

Функция выполняет запрос чтения данных абсолютных ДОС для осей.

Является системной.

3.4.2.23 axesAbsPosReadComplete

Синтаксис:	*int axesAbsPosReadComplete();*
Аргумент(ы):	*Нет*
Файл объявления:	*include/func/axis.h*

Функция возвращает 1, если чтение данных абсолютных ДОС для осей завершено, и 0 в противном случае.

Является системной.

3.4.2.24 axisRefPosComplete

Синтаксис:	*int axisRefPosComplete(unsigned axis);*
Аргумент(ы):	*unsigned axis – номер оси*
Файл объявления:	*include/func/axis.h*

Функция возвращает 1, если требуется и выполнено позиционирование оси, номер которой является аргументом функции, при включении станка или запуске программы, и 0 в противном случае.

Является системной.

3.4.2.25 axesRefPosComplete

Синтаксис:	*int axesRefPosComplete();*
Аргумент(ы):	*Нет*
Файл объявления:	*include/func/axis.h*

Функция возвращает 1, если требуется и выполнено позиционирование всех осей при включении станка или запуске программы, и 0 в противном случае.

Является системной.

3.4.2.26 axisAtRefPos

Синтаксис:	*int axisAtRefPos(unsigned axis);*
Аргумент(ы):	*unsigned axis – номер оси*
Файл объявления:	*include/func/axis.h*

Функция возвращает 1, если ось, номер которой является аргументом функции, находится в первой референтной позиции, и 0 в противном случае.

Является системной.

3.4.2.27 axesAtRefPos

Синтаксис:	*int axesAtRefPos();*
Аргумент(ы):	*Нет*
Файл объявления:	*include/func/axis.h*

Функция возвращает 1, если все оси находятся в первой референтной позиции, и 0 в противном случае.

Является системной.

3.5 Управление шпинделями

3.5.1 Типы данных

3.5.1.1 SpindleStates

Тип данных: *Перечисление SpindleStates*
Файл объявления: *include/func/spnd.h*

Перечисление определяет идентификаторы состояний шпинделя.

Таблица 3.33. Перечисление SpindleStates

Идентификатор	Описание
spndInactive	Шпиндель выключен
spndActive	Шпиндель находится в слежении
spndCW	Вращение по часовой стрелке
spndCCW	Вращение против часовой стрелки
spndStopping	Останов шпинделя
spndHomeWaitHW	Ожидание применения аппаратных настроек выезда в нулевую точку
spndHoming	Выезд в нулевую точку
spndIndexWaitHW	Ожидание применения аппаратных настроек поиска индексной метки
spndIndexing	Поиск индексной метки
spndAborting	Аварийное торможение
spndWaitActivate	Ожидание включения
spndWaitDeactivate	Ожидание выключения
spndWaitPhaseRef	Ожидание фазировки

3.5.1.2 SpindleCommands

Тип данных: *Перечисление SpindleCommands*
Файл объявления: *include/func/spnd.h*

Перечисление определяет идентификаторы команд управления шпинделями.

Таблица 3.34. Перечисление SpindleCommands

Идентификатор	Описание
spndCmdIdle	Нет команды
spndCmdKill	Выключить шпиндель
spndCmdActivate	Включить шпиндель в слежение
spndCmdDeactivate	Выключить шпиндель
spndCmdCW	Выполнить вращение по часовой стрелке
spndCmdCCW	Выполнить вращение против часовой стрелки
spndCmdStop	Выполнить останов
spndCmdInc	Выполнить поворот на заданный угол
spndCmdAbs	Выполнить поворот в заданное положение
spndCmdHome	Выполнить движение в нулевую точку
spndCmdIndex	Выполнить движение до индексной метки
spndCmdPhaseRef	Выполнить фазировку
spndCmdAbort	Выполнить аварийное выключение

3.5.1.3 SpindleStage

Тип данных: *Структура SpindleStage*
Файл объявления: *include/func/spnd.h*

Структура определяет настройки ступенчатого разгона шпинделя.

Таблица 3.35. Структура SpindleStage

Элемент	Тип	Описание
Vmin	double	Минимальная скорость
Vmax	doubled	Максимальная скорость
Amax	double	Максимальное ускорение
Jmax	double	Максимальный рывок

3.5.1.4 SpindleConfig

Тип данных: *Структура SpindleConfig*
Файл объявления: *include/func/spnd.h*

Структура определяет настройки шпинделя.

Таблица 3.36. Структура SpindleConfig

Элемент	Тип	Описание
servo	unsigned	Номер платы управления ($0 \div 3$)
chan	unsigned	Номер канала ($0 \div 7$)
motor	unsigned	Номер связанного с осью двигателя ($0 \div 31$)
homeOrder	unsigned	Порядок выезда в нулевую точку
needDKill	Битовое поле:1	Требуется задержка перед отключением
needPhaseRef	Битовое поле:1	Требуется фазировка
needHome	Битовое поле:1	Требуется выезд в нулевую точку
needIndex	Битовое поле:1	Требуется позиционирование по индексной метке
hasAbsPos	Битовое поле:1	Установлен абсолютный датчик
abortMode	Битовое поле:2	Реакции на команду аварийного выключения (см. **AxisAbortMode**)
homeCaptCtrl	Битовое поле:4	Настройка CaptCtrl для выезда в нулевую точку по входу (флагу)
indexCaptCtrl	Битовое поле:4	Настройка CaptCtrl для выезда в нулевую точку по индексной метке
killAfterStop	Битовое поле:1	Выключение после останова
reserved	Битовое поле:16	Резерв
homeVel	double	Скорость выезда в нулевую точку
indexVel	double	Скорость поиска индексной метки
homeOffset	double	Смещение нулевой точки относительно позиции ДОС
indexOffset	double	Смещение индексной метки относительно позиции ДОС
homeOfsVel	double	Скорость движения в позицию смещения нулевой точки (не используется)
indexOfsVel	double	Скорость движения в позицию смещения индексной метки (не используется)
minPos	double	Программное ограничение в отрицательном направлении (для абсолютного ДОС настраивается в дискретах датчика, определённых в энкодерной таблице)

Продолжение таблицы 3.36.

Элемент	Тип	Описание
maxPos	double	Программное ограничение в положительном направлении (для абсолютного ДОС настраивается в дискретах датчика, определённых в энкодерной таблице)
spinEncRes	double	Число дискрет датчика на оборот
atSpeedBand	double	Амплитуда зоны ошибки заданной скорости
stages [ЧИСЛО_СТУПЕНЕЙ_РАЗГОНА]	**SpindleStage**	Параметры ступенчатого разгона

Поля `homeCaptCtrl` и `indexCaptCtrl` являются 4-битными и содержат настройки захвата положения для выезда в нулевую точку:

- биты 0 и 1 определяют тип захвата положения (0 – непосредственный захват, 1 – по индексному сигналу датчика, 2 – захват по флагу, 3 – по флагу и индексному сигналу);
- бит 2 управляет инверсией индексного сигнала ДОС (0 – не инвертировать, 1 – инвертировать);
- бит 3 управляет инверсией флага при захвате положения (0 – не инвертировать, 1 – инвертировать).

3.5.1.5 Spindle

Тип данных: *Структура Spindle*
Файл объявления: *include/func/spnd.h*

Структура определяет состояние, параметры и данные шпинделя.

Таблица 3.37. Структура Spindle

Элемент	Тип	Описание
state	**SpindleStates**	Текущее состояние
command	**SpindleCommands**	Текущая команда
commandAfterActivate	**SpindleCommands**	Команда после включения
followup	Битовое поле:1	Восстановление после ошибки
phaseRefComplete	Битовое поле:1	Фазировка выполнена
phaseRefError	Битовое поле:1	Ошибка фазировки

Продолжение таблицы 3.37.

Элемент	Тип	Описание
homeComplete	Битовое поле:1	Выполнен выезд в нулевую точку
homeErrorFlag	Битовое поле:1	Ошибка выезда в нулевую точку
atSpeed	Битовое поле:1	Заданная скорость достигнута
timer	**Timer**	Таймер
SpeedValue	double	Значение скорости
SpeedOverride	double	Значение корректора скорости
spinStage	unsigned	Номер ступени разгона
platform	SpindlePlatformControl	Пользовательские параметры и переменные шпинделя

Структура `SpindlePlatformControl` является пользовательской и служит для введения дополнительных параметров и переменных шпинделя.

Если структура `SpindlePlatformControl` задана пользователем, то должен быть определён идентификатор `SPINDLE_PLATFORM_CONTROL_DEFINED`: `#define SPINDLE_PLATFORM_CONTROL_DEFINED`.

3.5.1.6 SpindleControl

Тип данных: *Структура SpindleControl*
Файл объявления: *include/func/spnd.h*

Структура определяет состояние, параметры и данные шпинделей.

Таблица 3.38. Структура SpindleControl

Элемент	Тип	Описание
homeState	**HomeStates**	Состояние выезда в нулевую точку
homeComplete	Битовое поле:1	Выполнен выезд в нулевую точку
homeErrorFlag	Битовое поле:1	Ошибка выезда в нулевую точку
homeStage	int	Этап выезда в нулевую точку
spin[ЧИСЛО_ШПИНДЕЛЕЙ]	**Spindle**	Данные шпинделя(-ей)
timerHome	**Timer**	Таймер для задержек переключений состояний в режиме выезда в нулевую точку
platform	SpindlesPlatformControl	Пользовательские параметры и переменные шпинделей

Структура `SpindlesPlatformControl` является пользовательской и служит для введения дополнительных параметров и переменных шпинделей.

Если структура `SpindlesPlatformControl` задана пользователем, то должен быть определён идентификатор `SPINDLE_PLATFORM_CONTROL_DEFINED`: `#define SPINDLE_PLATFORM_CONTROL_DEFINED`.

3.5.2 Функции

3.5.2.1 spinsForceKill

Синтаксис:	*void spinsForceKill();*
Аргумент(ы):	*Нет*
Файл объявления:	*include/func/spnd.h*

Функция вызывает принудительное выключение всех шпинделей.
Является системной.

3.5.2.2 spinForceKill

Синтаксис:	*void spinForceKill(unsigned spin);*
Аргумент(ы):	*unsigned spin – номер шпинделя*
Файл объявления:	*include/func/spnd.h*

Функция вызывает принудительное выключение шпинделя, номер которого является аргументом функции.
Является системной.

3.5.2.3 spinsDeactivate

Синтаксис:	*void spinsDeactivate();*
Аргумент(ы):	*Нет*
Файл объявления:	*include/func/spnd.h*

Функция вызывает выключение всех шпинделей.
Является системной.

3.5.2.4 spinsActivate

Синтаксис:	*void spinsActivate();*
Аргумент(ы):	*Нет*
Файл объявления:	*include/func/spnd.h*

Функция вызывает включение в слежение всех шпинделей.
Является системной.

3.5.2.5 spinsInactive

Синтаксис:	*int spinsInactive();*
Аргумент(ы):	*Нет*
Файл объявления:	*include/func/spnd.h*

Функция возвращает 1, если хотя бы один шпиндель не находится в слежении, и 0 в противном случае.

Является системной.

3.5.2.6 spinsActive

Синтаксис:	*int spinsActive();*
Аргумент(ы):	*Нет*
Файл объявления:	*include/func/spnd.h*

Функция возвращает 1, если все шпиндели находятся в слежении, и 0 в противном случае.

Является системной.

3.5.2.7 spinsPhaseRefComplete

Синтаксис:	*int spinsPhaseRefComplete();*
Аргумент(ы):	*Нет*
Файл объявления:	*include/func/spnd.h*

Функция возвращает 1, если фазировка выполнена для всех шпинделей, и 0 в противном случае.

Является системной.

3.5.2.8 spinsPhaseRef

Синтаксис:	*int spinsPhaseRef();*
Аргумент(ы):	*Нет*
Файл объявления:	*include/func/spnd.h*

Функция возвращает 1, если хотя бы один шпиндель требует фазировки, и 0 в противном случае. Для шпинделя, фазировка которого не выполнена, даётся команда фазировки.

Является системной.

3.5.2.9 spinAborted

Синтаксис:	*int spinAborted(unsigned spin);*
Аргумент(ы):	*unsigned spin – номер шпинделя*
Файл объявления:	*include/func/spnd.h*

Функция возвращает 1, если шпиндель, номер которого является аргументом функции, аварийно остановлен, и 0 в противном случае.

Является системной.

3.5.2.10 spinsAborted

Синтаксис:	*int spinsAborted();*
Аргумент(ы):	*Нет*
Файл объявления:	*include/func/spnd.h*

Функция возвращает 1, если все шпиндели аварийно остановлены, и 0 в противном случае.

Является системной.

3.5.2.11 spinIsStopped

Синтаксис:	*int spinIsStopped(unsigned spin);*
Аргумент(ы):	*unsigned spin – номер шпинделя*
Файл объявления:	*include/func/spnd.h*

Функция возвращает 1, если шпиндель, номер которого является аргументом функции, остановлен (в слежении имеет равную нулю заданную скорость), и 0 в противном случае.

Является системной.

3.5.2.12 spinsStopped

Синтаксис:	*int spinsStopped();*
Аргумент(ы):	*Нет*
Файл объявления:	*include/func/spnd.h*

Функция возвращает 1, если все шпиндели остановлены (в слежении имеют равную нулю заданную скорость), и 0 в противном случае.

Является системной.

3.5.2.13 spinsAbortAll

Синтаксис:	*void spinsAbortAll();*
Аргумент(ы):	*Нет*
Файл объявления:	*include/func/spnd.h*

Функция вызывает аварийное выключение всех шпинделей.

Является системной.

3.5.2.14 spinsStopAll

Синтаксис:	*void spinsStopAll();*
Аргумент(ы):	*Нет*
Файл объявления:	*include/func/spnd.h*

Функция вызывает останов всех шпинделей при толчковых перемещениях.

Является системной.

3.5.2.15 spinAtSpeed

Синтаксис:	*int spinAtSpeed(unsigned spin);*
Аргумент(ы):	*unsigned spin – номер шпинделя*
Файл объявления:	*include/func/spnd.h*

Функция возвращает 1, если шпиндель, номер которого является аргументом функции, имеет скорость равную заданной, и 0 в противном случае.

Является системной.

3.5.2.16 spinPosition

Синтаксис:	*double spinPosition(unsigned spin);*
Аргумент(ы):	*unsigned spin – номер шпинделя*
Файл объявления:	*include/func/spnd.h*

Функция возвращает заданную позицию шпинделя, номер которого является аргументом функции.

Возвращаемое значение измеряется в единицах `spinEncRes` (см. структуру **SpindleConfig**).

Является системной.

3.5.2.17 spinSpeedCommand

Синтаксис:	*void spinSpeedCommand(unsigned spin, double speed, int direction);*
Аргумент(ы):	*unsigned spin – номер шпинделя,* *double speed – скорость,* *int direction – направление вращения*
Файл объявления:	*include/func/spnd.h*

Функция задаёт скорость и направление вращения шпинделя, номер которого является аргументом функции.

Является системной.

3.5.2.18 spinCurStage

Синтаксис:	*unsigned spinCurStage(unsigned spin);*
Аргумент(ы):	*unsigned spin – номер шпинделя*
Файл объявления:	*include/func/spnd.h*

Функция возвращает номер ступени разгона шпинделя, номер которого является аргументом функции.

Является системной.

3.5.2.19 spinNeedChangeStage

Синтаксис:	*int spinNeedChangeStage(unsigned spin);*
Аргумент(ы):	*unsigned spin – номер шпинделя*
Файл объявления:	*include/func/spnd.h*

Функция возвращает 1, если требуется смена ступени разгона шпинделя, номер которого является аргументом функции, и 0 в противном случае.

Является системной.

3.5.2.20 spinsFollowup

Синтаксис:	*void spinsFollowup();*
Аргумент(ы):	*Нет*
Файл объявления:	*include/func/axis.h*

Функция устанавливает для всех шпинделей флаг «Восстановление после ошибки» (см. структуру **Spindle**).

Является системной.

3.5.2.21 initSpindle

Синтаксис:	*void initSpindle(int spin);*
Аргумент(ы):	*int spin – номер шпинделя*
Файл объявления:	*include/func/axis.h*

Функция выполняет инициализацию шпинделя, номер которого является аргументом функции, параметрами по умолчанию.

Является системной.

3.5.2.22 initSpindles

Синтаксис:	*void initSpindles();*
Аргумент(ы):	*Нет*
Файл объявления:	*include/func/axis.h*

Функция выполняет инициализацию шпинделей параметрами по умолчанию.

Является системной.

3.5.2.23 spinInitPlatform

Синтаксис:	*void spinInitPlatform(int spin);*
Аргумент(ы):	*int spin – номер шпинделя*
Файл объявления:	*include/func/axis.h*

Функция выполняет инициализацию параметров шпинделя, номер которого является аргументом функции, пользовательскими значениями, в том числе структуры SpindlePlatformControl.

Реализуется пользователем.

3.5.2.24 spinsInitPlatform

Синтаксис:	*void spinsInitPlatform();*
Аргумент(ы):	*Нет*
Файл объявления:	*include/func/axis.h*

Функция выполняет инициализацию параметров шпинделей пользовательскими значениями, в том числе структуры SpindlesPlatformControl.

Реализуется пользователем.

3.6 Датчики обратной связи

3.6.1 Типы данных

3.6.1.1 EncType

Тип данных:	*Перечисление EncType*
Файл объявления:	*include/func/enc.h*

Перечисление определяет идентификаторы типов датчиков обратной связи.

Таблица 3.39. Перечисление EncType

Идентификатор	Описание
encNone	Нет
encIncrement	Инкрементальный ДОС
encSinCos	Синусно-косинусный ДОС
encEnDat	ДОС с интерфейсом EnDat 2.2
encBiSS	ДОС с интерфейсом BiSS

3.6.1.2 EncConfig

Тип данных:	*Структура EncConfig*
Файл объявления:	*include/func/enc.h*

Структура определяет параметры датчика.

Таблица 3.40. Структура EncConfig

Элемент	Тип	Описание
servo	unsigned	Номер платы (0÷3)
chan	unsigned	Номер канала (0÷7)
type	unsigned	Тип датчика (см. **EncType**)

3.7 Реферирование осей

3.7.1 Типы данных

3.7.1.1 HomeStates

Тип данных: *Перечисление HomeStates*
Файл объявления: *include/func/home.h*

Перечисление определяет идентификаторы состояний выезда в нулевую точку.

Таблица 3.41. Перечисление HomeStates

Идентификатор	Описание
homeReady	Выезд в ноль не выполнен
homeStart	Начало выезда в нулевую точку
homeWaitStage	Ожидание этапа выезда в нулевую точку
homeComplete	Выезд в нулевую точку выполнен
homeError	Ошибка выезда в нулевую точку

3.7.2 Функции

3.7.2.1 isHomeComplete

Синтаксис: *int isHomeComplete();*
Аргумент(ы): *Нет*
Файл объявления: *include/func/home.h*

Функция возвращает 1, если завершено реферирование осей и шпинделей, и 0 в противном случае.

Реализуется пользователем.

3.7.2.2 isHoming

Синтаксис:	*int isHoming();*
Аргумент(ы):	*Нет*
Файл объявления:	*include/func/home.h*

Функция возвращает 1, если выполняется реферирование осей и шпинделей, и 0 в противном случае.

Реализуется пользователем.

3.7.2.3 startHoming

Синтаксис:	*void startHoming();*
Аргумент(ы):	*Нет*
Файл объявления:	*include/func/home.h*

Функция инициирует начало реферирование осей и шпинделей.

Реализуется пользователем.

3.7.2.4 isHomingError

Синтаксис:	*int isHomingError();*
Аргумент(ы):	*Нет*
Файл объявления:	*include/func/home.h*

Функция возвращает 1, если произошла ошибка при реферировании осей или шпинделей, и 0 в противном случае.

Реализуется пользователем.

3.7.2.5 homeCancel

Синтаксис:	*void homeCancel();*
Аргумент(ы):	*Нет*
Файл объявления:	*include/func/home.h*

Функция выполняет останов реферирования осей и шпинделей.

Реализуется пользователем.

3.8 Состояние управляющей программы

3.8.1 Функции

3.8.1.1 csProgramRunning

Синтаксис:	*int csProgramRunning(int cs);*
Аргумент(ы):	*int cs – номер координатной системы*
Файл объявления:	*include/func/cs.h*

Функция возвращает 1, если выполняется УП, и 0 в противном случае.
Является системной.

3.8.1.2 csProgramHolding

Синтаксис:	*int csProgramHolding(int cs);*
Аргумент(ы):	*int cs – номер координатной системы*
Файл объявления:	*include/func/cs.h*

Функция возвращает 1, если произведён приостанов подачи, и 0 в противном случае.
Является системной.

3.8.1.3 csProgramStarting

Синтаксис:	*int csProgramStarting(int cs);*
Аргумент(ы):	*int cs – номер координатной системы*
Файл объявления:	*include/func/cs.h*

Функция возвращает 1, если УП начинает выполняться, и 0 в противном случае.
Является системной.

3.8.1.4 csProgramPaused

Синтаксис:	*int csProgramPaused(int cs);*
Аргумент(ы):	*int cs – номер координатной системы*
Файл объявления:	*include/func/cs.h*

Функция возвращает 1, если УП временно приостановлена, и 0 в противном случае.
Является системной.

3.8.1.5 csProgramStopped

Синтаксис:	*int csProgramStopped(int cs);*
Аргумент(ы):	*int cs – номер координатной системы*
Файл объявления:	*include/func/cs.h*

Функция возвращает 1, если УП остановлена, и 0 в противном случае.
Является системной.

3.9 Очередь команд

3.9.1 Типы данных

3.9.1.1 CommandRequest

Тип данных: *Структура CommandRequest*
Файл объявления: *include/func/commands.h*

Структура определяет параметры команды.

Таблица 3.42. Структура CommandRequest

Элемент	Тип	Описание
command	unsigned	Идентификатор команды
prio	unsigned	Приоритет
next	int	Номер следующей команды в очереди

3.9.1.2 CommandQueue

Тип данных: *Структура CommandQueue*
Файл объявления: *include/func/commands.h*

Структура определяет параметры очереди команд.

Таблица 3.43. Структура CommandQueue

Элемент	Тип	Описание
used	int	Первый используемый номер команды в очереди
free	int	Первый неиспользуемый номер команды в очереди
queue[РАЗМЕР_ОЧЕРЕДИ_КОМАНД]	**CommandRequest**	Массив команд

3.9.2 Функции

3.9.2.1 commandsInit

Синтаксис:	*void commandsInit(struct CommandQueue &queue);*
Аргумент(ы):	*struct **CommandQueue** &queue – очередь команд*
Файл объявления:	*include/func/commands.h*

Функция инициализирует очередь команд.

Является системной.

3.9.2.2 commandPush

Синтаксис:	*int commandPush(struct CommandQueue &queue,* *unsigned command, unsigned prio);*
Аргумент(ы):	*struct **CommandQueue** &queue – очередь команд,* *unsigned command – идентификатор команды,* *unsigned prio – приоритет команды*
Файл объявления:	*include/func/commands.h*

Функция помещает команду с учётом заданного приоритета в очередь команд.

Функция возвращает номер команды, если она помещена в очередь, и -1 в случае ошибки.

Является системной.

3.9.2.3 commandPop

Синтаксис:	*unsigned commandPop(struct CommandQueue &queue);*
Аргумент(ы):	*struct **CommandQueue** &queue – очередь команд*
Файл объявления:	*include/func/commands.h*

Функция возвращает идентификатор команды, которая должна быть выполнена с учётом приоритета, из очереди команд.

Является системной.

3.9.2.4 commandFlush

Синтаксис:	*void commandFlush(struct CommandQueue &queue);*
Аргумент(ы):	*struct **CommandQueue** &queue – очередь команд*
Файл объявления:	*include/func/commands.h*

Функция очищает очередь команд.

Является системной.

3.10 Управление движением

3.10.1 Типы данных

3.10.1.1 Axes

Тип данных: *Перечисление Axes*
Файл объявления: *sys/sys.h*

Перечисление определяет идентификаторы осей.

Таблица 3.44. Перечисление Axes

Идентификатор	Идентификатор	Идентификатор	Идентификатор
axA	axXA	axXM	axXV
axB	axXB	axXN	axXW
axC	axXC	axXO	axXX
axU	axXD	axXP	axXY
axV	axXE	axXQ	axXZ
axW	axXF	axXR	axABC
axX	axXG	axXS	axUVW
axY	axXH	axXT	axXYZ
axZ	axXL	axXU	axAll

3.10.1.2 Vectors

Тип данных: *Перечисление Vectors*
Файл объявления: *sys/sys.h*

Перечисление определяет идентификаторы векторов.

Таблица 3.45. Перечисление Axes

Идентификатор	Идентификатор	Идентификатор	Идентификатор
vI	vJ	vK	vXI
vXJ	vXK	vIJK	vXIJK

3.10.1.3 XYZ

Тип данных: *Структура XYZ*
Файл объявления: *sys/sys.h*

Структура определяет координаты по осям декартовой системы координат.

Таблица 3.46. Структура XYZ

Элемент	Тип	Описание
X	double	Координата по оси X
Y	double	Координата по оси Y
Z	double	Координата по оси Z

3.10.1.4 SpindleTimeBase

Тип данных: *Перечисление SpindleTimeBase*
Файл объявления: *sys/sys.h*

Перечисление определяет идентификаторы временной развёртки шпинделя.

Таблица 3.47. Перечисление SpindleTimeBase

Идентификатор	Описание
spinUseCSTimebase	Временная развёртка указанной координатной системы
spinUseCS0TimeBase	Временная развёртка координатной системы № 0
spinUseFixedTimeBase	100% фиксированная временная развёртка

3.10.1.5 MotorDefinition

Тип данных: *Структура MotorDefinition*
Файл объявления: *sys/sys.h*

Структура определяет параметры привязки двигателя к оси координатной системы.

Таблица 3.48. Структура MotorDefinition

Элемент	Тип	Описание
A, B, C, U, V, W, X, Y, Z XA, XB, XC, XD, XE, XF, XG, XH XL, XM, XN, XO, XP, XQ, XR, XS XT, XU, XV, XW, XX, XY, XZ	double	Масштабные коэффициенты, связывающие положение двигателя и координаты осей (число дискрет перемещения для двигателя на одну единицу величины перемещения по оси)
Ofs	double	Смещение между нулевой точкой двигателя и нулевой позицией оси

3.10.1.6 Pos

Тип данных: *Объединение Pos*
Файл объявления: *sys/sys.h*

Объединение определяет данные перемещения для различных режимов движения.

Таблица 3.49. Объединение Pos

Элемент	Тип	Описание
struct { A, B, C, U, V, W, X, Y, Z I, J, K R }	 double double double	 Координаты по осям Компоненты вектора Радиус
struct { Axis[9] Vec[6] }	 double double	 Координаты по осям Компоненты вектора, радиус

3.10.1.7 JogTarget

Тип данных: *Структура JogTarget*
Файл объявления: *sys/sys.h*

Структура определяет координаты и смещения для толчковых перемещений.

Таблица 3.50. Структура JogTarget

Элемент	Тип	Описание
pos[32]	double	Координаты
offset[32]	double	Смещения

3.10.1.8 Vec

Тип данных: *Объединение Vec*
Файл объявления: *sys/sys.h*

Объединение определяет компоненты вектора.

Таблица 3.51. Объединение Pos

Элемент	Тип	Описание
struct { I, J, K }	double	Компоненты вектора
V[3]	double	Компоненты вектора

3.10.2 Функции и макросы

3.10.2.1 cout

Синтаксис: *int cout(int motor, double level);*
Аргумент(ы): *int motor – номер двигателя,*
double level – значение тока двигателя в % от максимального тока
Файл объявления: *sys/sys.h*

Функция вызывает замыкание контура тока/момента двигателя с величиной задания в % от максимального тока.

Первый аргумент функции `motor` – номер двигателя (0÷31). Второй аргумент `level` -– значение тока двигателя в % от максимального тока. Знак задания определяет направление вращения вала двигателя, величина задания должна находиться в диапазон от 0 до 100.

Возвращаемое значение равно 0 при отсутствии ошибок и отлично от 0 в противном случае.

Является системной.

3.10.2.2 kill

Синтаксис:	*int kill(int motor);*
Аргумент(ы):	*int motor – номер двигателя*
Файл объявления:	*sys/sys.h*

Функция вызывает снятие управления и полное отключение двигателя, номер которого определяется аргументом функции, с последующим остановом в режиме свободного выбега (категория останова 0).

Возвращаемое значение равно 0 при отсутствии ошибок и отлично от 0 в противном случае.

Является системной.

3.10.2.3 killMulti

Синтаксис:	*int killMulti(int motors);*
Аргумент(ы):	*int motors – номера двигателей*
Файл объявления:	*sys/sys.h*

Функция снятие управления и полное отключение двигателей, номера которых определяются аргументом функции, с последующим остановом в режиме свободного выбега (категория останова 0).

Аргумент функции – битовое поле, в котором номера установленных битов (значения которых равны 1) соответствуют номерам отключаемых двигателей.

Возвращаемое значение равно 0 при отсутствии ошибок и отлично от 0 в противном случае.

Является системной.

3.10.2.4 dkill

Синтаксис:	*int dkill(int motor);*
Аргумент(ы):	*int motor – номер двигателя*
Файл объявления:	*sys/sys.h*

Функция вызывает снятие управления и полное отключение двигателя, номер которого определяется аргументом функции, с задержкой на включение тормоза (категория останова 0).

Возвращаемое значение равно 0 при отсутствии ошибок и отлично от 0 в противном случае.

Является системной.

3.10.2.5 dkillMulti

Синтаксис:	*int dkillMulti(int motors);*
Аргумент(ы):	*int motors – номера двигателей*
Файл объявления:	*sys/sys.h*

Функция вызывает снятие управления и полное отключение двигателей, номера которых определяются аргументом функции, с задержкой на включение тормоза (категория останова 0).

Аргумент функции – битовое поле, в котором номера установленных битов (значения которых равны 1) соответствуют номерам отключаемых двигателей.

Возвращаемое значение равно 0 при отсутствии ошибок и отлично от 0 в противном случае.

Является системной.

3.10.2.6 abortMotor

Синтаксис:	*int abortMotor(int motor);*
Аргумент(ы):	*int motor – номер двигателя*
Файл объявления:	*sys/sys.h*

Функция выполняет управляемый аварийный останов двигателя, номер которого определяется аргументом функции. После останова двигатель либо выключается (категория останова 1) либо остается в слежении (категория останова 2).

Возвращаемое значение равно 0 при отсутствии ошибок и отлично от 0 в противном случае.

Является системной.

3.10.2.7 abortMotorMulti

Синтаксис:	*int abortMotorMulti(int motors);*
Аргумент(ы):	*int motors – номера двигателей*
Файл объявления:	*sys/sys.h*

Функция выполняет управляемый аварийный останов двигателей, номера которых определяются аргументом функции. После останова двигатели либо выключаются (категория останова 1) либо остаются в слежении (категория останова 2).

Аргумент функции – битовое поле, в котором номера установленных битов (значения которых равны 1) соответствуют номерам останавливаемых двигателей.

Возвращаемое значение равно 0 при отсутствии ошибок и отлично от 0 в противном случае.

Является системной.

3.10.2.8 adisableMotor

Синтаксис:	*int adisableMotor(int motor);*
Аргумент(ы):	*int motor – номер двигателя*
Файл объявления:	*sys/sys.h*

Функция выполняет управляемый аварийный останов двигателя, номер которого определяется аргументом функции, с последующим отключением с задержкой на включение тормоза (категория останова 1).

Возвращаемое значение равно 0 при отсутствии ошибок и отлично от 0 в противном случае.

Является системной.

3.10.2.9 adisableMotorMulti

Синтаксис:	*int adisableMotorMulti(int motors);*
Аргумент(ы):	*int motors – номера двигателей*
Файл объявления:	*sys/sys.h*

Функция выполняет управляемый аварийный останов двигателей, номера которых определяются аргументом функции, с последующим их отключением с задержкой на включение тормоза (категория останова 1).

Аргумент функции – битовое поле, в котором номера установленных битов (значения которых равны 1) соответствуют номерам останавливаемых двигателей.

Возвращаемое значение равно 0 при отсутствии ошибок и отлично от 0 в противном случае.

Является системной.

3.10.2.10 assignMotor

Синтаксис:	*int assignMotor(int motor, const MotorDefinition &def);*
Аргумент(ы):	*int motor – номер двигателя,* *const **MotorDefinition** &def – параметры привязки двигателя к оси*
Файл объявления:	*sys/sys.h*

Функция выполняет привязку двигателя, номер которого определяется аргументом функции, к оси координатной системы.

Возвращаемое значение равно 0 при отсутствии ошибок и отлично от 0 в противном случае.

Является системной.

3.10.2.11 assignMotorInverse

Синтаксис:	*int assignMotorInverse(int motor);*
Аргумент(ы):	*int motor – номер двигателя*
Файл объявления:	*sys/sys.h*

Функция выполняет привязку двигателя, номер которого определяется аргументом функции, к оси инверсной кинематики.

Возвращаемое значение равно 0 при отсутствии ошибок и отлично от 0 в противном случае.

Является системной.

3.10.2.12 assignMotorSpindle

Синтаксис:	*int assignMotorSpindle(int motor, SpindleTimeBase mode);*
Аргумент(ы):	*int motor – номер двигателя,*
	***SpindleTimeBase** mode – идентификатор временной развёртки*
Файл объявления:	*sys/sys.h*

Функция выполняет привязку двигателя, номер которого определяется аргументом функции, к шпиндельной оси.

Возвращаемое значение равно 0 при отсутствии ошибок и отлично от 0 в противном случае.

Является системной.

3.10.2.13 unassignMotor

Синтаксис:	*int unassignMotor(int motor);*
Аргумент(ы):	*int motor – номер двигателя*
Файл объявления:	*sys/sys.h*

Функция выполняет отвязку двигателя, номер которого определяется аргументом функции, от оси (обнуляет масштабирующие коэффициенты, связывающие положение двигателя и координаты осей).

Возвращаемое значение равно 0 при отсутствии ошибок и отлично от 0 в противном случае.

Является системной.

3.10.2.14 phaseref

Синтаксис:	*int phaseref(int motor);*
Аргумент(ы):	*int motor – номер двигателя*
Файл объявления:	*sys/sys.h*

Функция вызывает выполнение фазировки двигателем, номер которого определяется аргументом функции.

Возвращаемое значение равно 0 при отсутствии ошибок и отлично от 0 в противном случае.

Является системной.

3.10.2.15 phaserefMulti

Синтаксис:	*int phaserefMulti(int motors);*
Аргумент(ы):	*int motors – номера двигателей*
Файл объявления:	*sys/sys.h*

Функция вызывает выполнение фазировки двигателями, номера которых определяются аргументом функции.

Аргумент функции – битовое поле, в котором номера установленных битов (значения которых равны 1) соответствуют номерам двигателей.

Возвращаемое значение равно 0 при отсутствии ошибок и отлично от 0 в противном случае.

Является системной.

3.10.2.16 home

Синтаксис:	*int home(int motor);*
Аргумент(ы):	*int motor – номер двигателя*
Файл объявления:	*sys/sys.h*

Функция вызывает выполнение поиска нулевой точки двигателем, номер которого определяется аргументом функции.

Возвращаемое значение равно 0 при отсутствии ошибок и отлично от 0 в противном случае.

Является системной.

3.10.2.17 homeMulti

Синтаксис:	*int homeMulti(int motors);*
Аргумент(ы):	*int motors – номера двигателей*
Файл объявления:	*sys/sys.h*

Функция вызывает выполнение поиска нулевой точки двигателями, номера которых определяются аргументом функции.

Аргумент функции – битовое поле, в котором номера установленных битов (значения которых равны 1) соответствуют номерам двигателей.

Возвращаемое значение равно 0 при отсутствии ошибок и отлично от 0 в противном случае.

Является системной.

3.10.2.18 homez

Синтаксис:	*int homez(int motor);*
Аргумент(ы):	*int motor – номер двигателя*
Файл объявления:	*sys/sys.h*

Функция вызывает установку новой позиции нулевой точки для двигателя, номер которого определяется аргументом функции.

Возвращаемое значение равно 0 при отсутствии ошибок и отлично от 0 в противном случае.

Является системной.

3.10.2.19 homezMulti

Синтаксис:	*int homezMulti(int motors);*
Аргумент(ы):	*int motors – номера двигателей*
Файл объявления:	*sys/sys.h*

Функция вызывает установку новой позиции нулевой точки для двигателей, номера которых определяются аргументом функции.

Аргумент функции – битовое поле, в котором номера установленных битов (значения которых равны 1) соответствуют номерам двигателей.

Возвращаемое значение равно 0 при отсутствии ошибок и отлично от 0 в противном случае.

Является системной.

3.10.2.20 jogPlus

Синтаксис:	*int jogPlus(int motor);*
Аргумент(ы):	*int motor – номер двигателя*
Файл объявления:	*sys/sys.h*

Функция вызывает толчковое перемещение в положительном направлении двигателем, номер которого определяется аргументом функции.

Возвращаемое значение равно 0 при отсутствии ошибок и отлично от 0 в противном случае.

Является системной.

3.10.2.21 jogMotorsPlus

Синтаксис:	*int jogMotorsPlus(int motors);*
Аргумент(ы):	*int motors – номера двигателей*
Файл объявления:	*sys/sys.h*

Функция вызывает толчковое перемещение в положительном направлении двигателями, номера которых определяются аргументом функции.

Аргумент функции – битовое поле, в котором номера установленных битов (значения которых равны 1) соответствуют номерам двигателей.

Возвращаемое значение равно 0 при отсутствии ошибок и отлично от 0 в противном случае.

Является системной.

3.10.2.22 jogMinus

Синтаксис:	*int jogMinus(int motor);*
Аргумент(ы):	*int motor – номер двигателя*
Файл объявления:	*sys/sys.h*

Функция вызывает толчковое перемещение в отрицательном направлении двигателем, номер которого определяется аргументом функции.

Возвращаемое значение равно 0 при отсутствии ошибок и отлично от 0 в противном случае.

Является системной.

3.10.2.23 jogMotorsMinus

Синтаксис:	*int jogMotorsMinus(int motors);*
Аргумент(ы):	*int motors – номера двигателей*
Файл объявления:	*sys/sys.h*

Функция вызывает толчковое перемещение в отрицательном направлении двигателями, номера которых определяются аргументом функции.

Аргумент функции – битовое поле, в котором номера установленных битов (значения которых равны 1) соответствуют номерам двигателей.

Возвращаемое значение равно 0 при отсутствии ошибок и отлично от 0 в противном случае.

Является системной.

3.10.2.24 jogStop

Синтаксис:	*int jogStop(int motor);*
Аргумент(ы):	*int motor – номер двигателя*
Файл объявления:	*sys/sys.h*

Функция вызывает останов толчкового перемещения двигателя, номер которого определяется аргументом функции.

Возвращаемое значение равно 0 при отсутствии ошибок и отлично от 0 в противном случае.

Является системной.

3.10.2.25 jogMotorsStop

Синтаксис:	*int jogMotorsStop(int motors);*
Аргумент(ы):	*int motors – номера двигателей*
Файл объявления:	*sys/sys.h*

Функция вызывает останов толчкового перемещения двигателей, номера которых определяются аргументом функции.

Аргумент функции – битовое поле, в котором номера установленных битов (значения которых равны 1) соответствуют номерам двигателей.

Возвращаемое значение равно 0 при отсутствии ошибок и отлично от 0 в противном случае.

Является системной.

3.10.2.26 jogTo

Синтаксис:	*int jogTo(int motor, double target);*
Аргумент(ы):	*int motor – номер двигателя,* *double target – заданная позиция*
Файл объявления:	*sys/sys.h*

Функция вызывает толчковое движение в заданную позицию относительно нулевой точки двигателя, номер которого определяется аргументом функции.

Возвращаемое значение равно 0 при отсутствии ошибок и отлично от 0 в противном случае.

Является системной.

3.10.2.27 jogMotorsTo

Синтаксис:	*int jogMotorsTo(JogTarget target);*
Аргумент(ы):	***JogTarget** target – заданные позиции*
Файл объявления:	*sys/sys.h*

Функция вызывает толчковое движение в заданные позиции относительно нулевой точки двигателей, номера которых определяются аргументом функции.

Аргумент функции – структура **JogTarget**, в которой номера ячеек массива со значениями, отличными от `NAN`, соответствуют номерам двигателей, а сами значения ячеек являются заданными позициями.

Возвращаемое значение равно 0 при отсутствии ошибок и отлично от 0 в противном случае.

Является системной.

3.10.2.28 jogRelToCmd

Синтаксис:	*int jogRelToCmd(int motor, double target);*
Аргумент(ы):	*int motor – номер двигателя,* *double target – заданное расстояние*
Файл объявления:	*sys/sys.h*

Функция вызывает толчковое движение на заданное расстояние относительно текущей программной позиции двигателя, номер которого определяется аргументом функции.

Возвращаемое значение равно 0 при отсутствии ошибок и отлично от 0 в противном случае.

Является системной.

3.10.2.29 jogMotorsRelToCmd

Синтаксис:	*int jogMotorsRelToCmd(JogTarget target);*
Аргумент(ы):	***JogTarget** target – заданные расстояния*
Файл объявления:	*sys/sys.h*

Функция вызывает толчковое движение на заданные расстояния относительно текущей программной позиции двигателей, номера которых определяются аргументом функции.

Аргумент функции – структура **JogTarget**, в которой номера ячеек массива со значениями, отличными от `NAN`, соответствуют номерам двигателей, а сами значения ячеек являются заданными расстояниями.

Возвращаемое значение равно 0 при отсутствии ошибок и отлично от 0 в противном случае.

Является системной.

3.10.2.30 jogRelToAct

Синтаксис:	*int jogRelToAct(int motor, double target);*
Аргумент(ы):	*int motor – номер двигателя,*
	double target – заданное расстояние
Файл объявления:	*sys/sys.h*

Функция вызывает толчковое движение на заданное расстояние относительно текущей фактической позиции двигателя, номер которого определяется аргументом функции.

Возвращаемое значение равно 0 при отсутствии ошибок и отлично от 0 в противном случае.

Является системной.

3.10.2.31 jogMotorsRelToAct

Синтаксис:	*int jogMotorsRelToAct(JogTarget target);*
Аргумент(ы):	***JogTarget** target – заданные расстояния*
Файл объявления:	*sys/sys.h*

Функция вызывает толчковое движение на заданные расстояния относительно текущей фактической позиции двигателей, номера которых определяются аргументом функции.

Аргумент функции – структура **JogTarget**, в которой номера ячеек массива со значениями, отличными от `NAN`, соответствуют номерам двигателей, а сами значения ячеек являются заданными расстояниями.

Возвращаемое значение равно 0 при отсутствии ошибок и отлично от 0 в противном случае.

Является системной.

3.10.2.32 jogRet

Синтаксис:	*int jogRet(int motor);*
Аргумент(ы):	*int motor – номер двигателя*
Файл объявления:	*sys/sys.h*

Функция вызывает толчковое движение в сохранённую позицию двигателем, номер которого определяется аргументом функции.

Возвращаемое значение равно 0 при отсутствии ошибок и отлично от 0 в противном случае.

Является системной.

3.10.2.33 jogMotorsRet

Синтаксис:	*int jogMotorsRet(int motors);*
Аргумент(ы):	*int motors – номера двигателей*
Файл объявления:	*sys/sys.h*

Функция вызывает толчковое движение в сохранённую позицию двигателями, номера которых определяются аргументом функции.

Аргумент функции – битовое поле, в котором номера установленных битов (значения которых равны 1) соответствуют номерам двигателей.

Возвращаемое значение равно 0 при отсутствии ошибок и отлично от 0 в противном случае.

Является системной.

3.10.2.34 jogToSave

Синтаксис:	*int jogToSave(int motor, double target);*
Аргумент(ы):	*int motor – номер двигателя,* *double target – заданная позиция*
Файл объявления:	*sys/sys.h*

Функция вызывает толчковое движение в заданную позицию относительно нулевой точки двигателя, номер которого определяется аргументом функции, и сохранение значения конечного положения.

Возвращаемое значение равно 0 при отсутствии ошибок и отлично от 0 в противном случае.

Является системной.

3.10.2.35 jogMotorsToSave

Синтаксис:	*int jogMotorsToSave(JogTarget target);*
Аргумент(ы):	***JogTarget** target – заданные позиции*
Файл объявления:	*sys/sys.h*

Функция вызывает толчковое движение в заданные позиции относительно нулевой точки двигателей, номера которых определяются аргументом функции, и сохранение значений конечного положения.

Аргумент функции – структура **JogTarget**, в которой номера ячеек массива со значениями, отличными от `NAN`, соответствуют номерам двигателей, а сами значения ячеек являются заданными позициями.

Возвращаемое значение равно 0 при отсутствии ошибок и отлично от 0 в противном случае.

Является системной.

3.10.2.36 absAxes

Синтаксис:	*int absAxes(unsigned axes);*
Аргумент(ы):	*unsigned axes – номера осей*
Файл объявления:	*sys/sys.h*

Функция устанавливает абсолютный режим перемещений для осей, номера которых определяются аргументом функции. В данном режиме программируется величина конечного положения.

Аргумент функции – битовое поле, в котором номера установленных битов (значения которых равны 1) соответствуют номерам осей.

Возвращаемое значение равно 0 при отсутствии ошибок и отлично от 0 в противном случае.

Является системной.

3.10.2.37 incAxes

Синтаксис:	*int incAxes(unsigned axes);*
Аргумент(ы):	*unsigned axes – номера осей*
Файл объявления:	*sys/sys.h*

Функция устанавливает относительный режим перемещений для осей, номера которых определяются аргументом функции. В данном режиме программируется величина перемещения от текущего положения.

Аргумент функции – битовое поле, в котором номера установленных битов (значения которых равны 1) соответствуют номерам осей.

Возвращаемое значение равно 0 при отсутствии ошибок и отлично от 0 в противном случае.

Является системной.

3.10.2.38 absVectors

Синтаксис:	*int absVectors(unsigned vectors);*
Аргумент(ы):	*unsigned vectors – номера векторов*
Файл объявления:	*sys/sys.h*

Функция устанавливает абсолютный режим для задающих центр окружности компонент вектора, номера которых определяются аргументом функции. В данном режиме компоненты I, J, K, II, JJ, KK параллельные осям X, Y, Z, XX, XY, XZ соответственно, определяют расстояние от начала координат до центра окружности.

Аргумент функции – битовое поле, в котором номера установленных битов (значения которых равны 1) соответствуют номерам векторов.

Возвращаемое значение равно 0 при отсутствии ошибок и отлично от 0 в противном случае.

Является системной.

3.10.2.39 incVectors

Синтаксис:	*int incVectors(unsigned vectors);*
Аргумент(ы):	*unsigned vectors – номера векторов*
Файл объявления:	*sys/sys.h*

Функция устанавливает относительный режим для задающих центр окружности компонент вектора, номера которых определяются аргументом функции. В данном режиме компоненты I, J, K, II, JJ, KK параллельные осям X, Y, Z, XX, XY, XZ соответственно, определяют расстояние от начальной точки перемещения до центра окружности.

Аргумент функции – битовое поле, в котором номера установленных битов (значения которых равны 1) соответствуют номерам векторов.

Возвращаемое значение равно 0 при отсутствии ошибок и отлично от 0 в противном случае.

Является системной.

3.10.2.40 frax

Синтаксис:	*int frax(unsigned axes);*
Аргумент(ы):	*unsigned axes – номера осей*
Файл объявления:	*sys/sys.h*

Функция определяет, какие оси должны быть задействованы в расчёте подачи в основной декартовой системы координат (X/Y/Z).

Аргумент функции – битовое поле, в котором номера установленных битов (значения которых равны 1) соответствуют номерам осей.

Возвращаемое значение равно 0 при отсутствии ошибок и отлично от 0 в противном случае.

Является системной.

3.10.2.41 frax2

Синтаксис:	*int frax2(unsigned axes);*
Аргумент(ы):	*unsigned axes – номера осей*
Файл объявления:	*sys/sys.h*

Функция определяет, какие оси должны быть задействованы в расчёте подачи в расширенной декартовой системы координат (XX/XY/XZ).

Аргумент функции – битовое поле, в котором номера установленных битов (значения которых равны 1) соответствуют номерам осей.

Возвращаемое значение равно 0 при отсутствии ошибок и отлично от 0 в противном случае.

Является системной.

3.10.2.42 nofrax

Синтаксис:	*int nofrax();*
Аргумент(ы):	*нет*
Файл объявления:	*sys/sys.h*

Функция отменяет выбор осей, задействованных в расчёте подачи в основной декартовой системы координат (X/Y/Z).

Возвращаемое значение равно 0 при отсутствии ошибок и отлично от 0 в противном случае.

Является системной.

3.10.2.43 nofrax2

Синтаксис:	*int nofrax2();*
Аргумент(ы):	*нет*
Файл объявления:	*sys/sys.h*

Функция отменяет выбор осей, задействованных в расчёте подачи в расширенной декартовой системы координат (XX/XY/XZ).

Возвращаемое значение равно 0 при отсутствии ошибок и отлично от 0 в противном случае.

Является системной.

3.10.2.44 delay

Синтаксис:	*int delay(double time);*
Аргумент(ы):	*double time – время останова*
Файл объявления:	*sys/sys.h*

Функция останавливает движение всех осей в координатной системе, в которой выполняется УП, на заданное время (удержание программной позиции в течение заданного времени).

Время останова, измеряемое в мс, включает в себя половину времени торможения и ускорения, не прерывает расчеты в буфере опережающего просмотра и масштабируется в зависимости от временной развёртки (например при увеличении значения временной развертки на 50% фактическое время останова в 2 раза превысит заданное).

Возвращаемое значение равно 0 при отсутствии ошибок и отлично от 0 в противном случае.

Является системной.

3.10.2.45 dwell

Синтаксис:	*int dwell(double time);*
Аргумент(ы):	*double time – время задержки*
Файл объявления:	*sys/sys.h*

Функция останавливает движение всех осей в координатной системе, в которой выполняется УП, на заданное время (удержание программной позиции в течение заданного времени).

Время задержки, измеряемое в мс, не учитывает время торможения и ускорения, прерывает расчеты в буфере опережающего просмотра и не зависит от временной развёртки.

Возвращаемое значение равно 0 при отсутствии ошибок и отлично от 0 в противном случае.

Является системной.

3.10.2.46 setF

Синтаксис:	*int setF(double feedrate);*
Аргумент(ы):	*double feedrate – величина скорости подачи*
Файл объявления:	*sys/sys.h*

Функция устанавливает скорость подачи, величина которой является аргументом функции.

Возвращаемое значение равно 0 при отсутствии ошибок и отлично от 0 в противном случае.

Является системной.

3.10.2.47 setS

Синтаксис:	*int setS(double spindle);*
Аргумент(ы):	*double spindle – величина скорости шпинделя*
Файл объявления:	*sys/sys.h*

Функция устанавливает скорость шпинделя, величина которой является аргументом функции.

Возвращаемое значение равно 0 при отсутствии ошибок и отлично от 0 в противном случае.

Является системной.

3.10.2.48 ta

Синтаксис:	*int ta(double time);*
Аргумент(ы):	*double time – время ускорения*
Файл объявления:	*sys/sys.h*

Функция устанавливает время заданного ускорения для программных линейных или круговых движений (время ускорения S-кривой), величина которого является аргументом функции. Оно используется как время начального ускорения после останова в начале последовательности сопряжённых перемещений и при переходах между последовательными перемещениями.

Если данное время больше, чем заданное функцией `int ts(double time)`, то общее время ускорения будет равно сумме этих времён.

Если данное время меньше, чем заданное функцией `int ts(double time)`, то общее время ускорения (торможения) будет равно удвоенному значению аргумента функции `int ts(double time)`.

Возвращаемое значение равно 0 при отсутствии ошибок и отлично от 0 в противном случае.

Является системной.

3.10.2.49 td

Синтаксис:	*int td(double time);*
Аргумент(ы):	*double time – время торможения*
Файл объявления:	*sys/sys.h*

Функция устанавливает время заданного ускорения торможения для программных линейных или круговых движений (время торможения S-кривой), величина которого является аргументом функции. Оно используется как время конечного ускорения торможения до останова в конце последовательности сопряжённых перемещений.

Если данное время больше, чем заданное функцией `int ts(double time)`, то общее время ускорения будет равно сумме этих времён.

Если данное время меньше, чем заданное функцией `int ts(double time)`, то общее время ускорения (торможения) будет равно удвоенному значению аргумента функции `int ts(double time)`.

Возвращаемое значение равно 0 при отсутствии ошибок и отлично от 0 в противном случае.

Является системной.

3.10.2.50 tm

Синтаксис:	*int tm(double time);*
Аргумент(ы):	*double time – время или модуль вектора скорости подачи*
Файл объявления:	*sys/sys.h*

Функция устанавливает время или модуль вектора скорости подачи для линейных или круговых движений.

Если значение аргумента больше нуля, то оно определяет время движения в мс. При этом скорость движения будет такой, чтобы перемещение было выполнено за указанное время.

Если значение аргумента меньше нуля, то оно определяет модуль вектора скорости. При этом время движения будет таким, чтобы перемещение было выполнено с указанной скоростью.

Возвращаемое значение равно 0 при отсутствии ошибок и отлично от 0 в противном случае.

Является системной.

3.10.2.51 ts

Синтаксис:	*int ts(double time);*
Аргумент(ы):	*double time – время разгона/торможения S-кривой*
Файл объявления:	*sys/sys.h*

Функция устанавливает для каждой половины заданной S-кривой время ускорения для программных линейных или круговых движений. Оно используется как время начального ускорения после останова в начале последовательности сопряжённых перемещений, при переходах между последовательными перемещениями и конечного ускорения торможения до останова в конце последовательности.

Возвращаемое значение равно 0 при отсутствии ошибок и отлично от 0 в противном случае.

Является системной.

3.10.2.52 abort

Синтаксис:	*int abort(int cs);*
Аргумент(ы):	*int cs – номер координатной системы*
Файл объявления:	*sys/sys.h*

Функция выполняет прерывание программы движения для координатной системы, номер которой определяется аргументом функции, а также управляемый аварийный останов двигателей в заданной координатной системе. После останова двигатели либо выключаются (категория останова 1) либо остаются в слежении (категория останова 2).

Возвращаемое значение равно 0 при отсутствии ошибок и отлично от 0 в противном случае.

Является системной.

3.10.2.53 abortMulti

Синтаксис:	*int abortMulti(int cs);*
Аргумент(ы):	*int cs – номера координатных систем*
Файл объявления:	*sys/sys.h*

Функция выполняет прерывание программ движения для заданных координатных систем, а также управляемый аварийный останов соответствующих двигателей. После останова двигатели либо выключаются (категория останова 1) либо остаются в слежении (категория останова 2).

Аргумент функции – битовое поле, в котором номера установленных битов (значения которых равны 1) соответствуют номерам координатных систем.

Возвращаемое значение равно 0 при отсутствии ошибок и отлично от 0 в противном случае.

Является системной.

3.10.2.54 adisable

Синтаксис:	*int adisable(int cs);*
Аргумент(ы):	*int cs – номер координатной системы*
Файл объявления:	*sys/sys.h*

Функция выполняет прерывание программы движения для координатной системы, номер которой определяется аргументом функции, а также управляемый аварийный останов двигателей в заданной координатной системе с последующим их отключением с задержкой на включение тормоза (категория останова 1).

Возвращаемое значение равно 0 при отсутствии ошибок и отлично от 0 в противном случае.

Является системной.

3.10.2.55 adisableMulti

Синтаксис:	*int adisableMulti(int cs);*
Аргумент(ы):	*int cs – номера координатных систем*
Файл объявления:	*sys/sys.h*

Функция выполняет прерывание программ движения для заданных координатных систем, а также управляемый аварийный останов соответствующих двигателей с последующим их отключением с задержкой на включение тормоза (категория останова 1).

Аргумент функции – битовое поле, в котором номера установленных битов (значения которых равны 1) соответствуют номерам координатных систем.

Возвращаемое значение равно 0 при отсутствии ошибок и отлично от 0 в противном случае.

Является системной.

3.10.2.56 disable

Синтаксис:	*int disable(int cs);*
Аргумент(ы):	*int cs – номер координатной системы*
Файл объявления:	*sys/sys.h*

Функция выполняет прерывание программы движения для координатной системы, номер которой определяется аргументом функции, а также снятие управления и полное отключение двигателей в заданной координатной системе с их последующим остановом в режиме свободного выбега (категория останова 0).

Возвращаемое значение равно 0 при отсутствии ошибок и отлично от 0 в противном случае.

Является системной.

3.10.2.57 disableMulti

Синтаксис:	*int disableMulti(int cs);*
Аргумент(ы):	*int cs – номера координатных систем*
Файл объявления:	*sys/sys.h*

Функция выполняет прерывание программ движения для заданных координатных систем, а также снятие управления и полное отключение соответствующих двигателей с их последующим остановом в режиме свободного выбега (категория останова 0).

Аргумент функции – битовое поле, в котором номера установленных битов (значения которых равны 1) соответствуют номерам координатных систем.

Возвращаемое значение равно 0 при отсутствии ошибок и отлично от 0 в противном случае.

Является системной.

3.10.2.58 ddisable

Синтаксис:	*int ddisable(int cs);*
Аргумент(ы):	*int cs – номер координатной системы*
Файл объявления:	*sys/sys.h*

Функция выполняет прерывание программы движения для координатной системы, номер которой определяется аргументом функции, а также снятие управления и полное отключение двигателей в заданной координатной системе с задержкой на включение тормоза (категория останова 0).

Возвращаемое значение равно 0 при отсутствии ошибок и отлично от 0 в противном случае.

Является системной.

3.10.2.59 ddisableMulti

Синтаксис:	*int ddisableMulti(int cs);*
Аргумент(ы):	*int cs – номера координатных систем*
Файл объявления:	*sys/sys.h*

Функция выполняет прерывание программ движения для заданных координатных систем, а также снятие управления и полное отключение соответствующих двигателей с задержкой на включение тормоза (категория останова 0).

Аргумент функции – битовое поле, в котором номера установленных битов (значения которых равны 1) соответствуют номерам координатных систем.

Возвращаемое значение равно 0 при отсутствии ошибок и отлично от 0 в противном случае.

Является системной.

3.10.2.60 enable

Синтаксис:	*int enable(int cs);*
Аргумент(ы):	*int cs – номер координатной системы*
Файл объявления:	*sys/sys.h*

Функция выполняет включение двигателей в слежение (включение и замыкание контура положения) в координатной системе, номер которой определяется аргументом функции. В слежение будут включены двигатели, которые находятся в отключенном состоянии или работающие в режиме контура тока/момента.

Возвращаемое значение равно 0 при отсутствии ошибок и отлично от 0 в противном случае.

Является системной.

3.10.2.61 enableMulti

Синтаксис:	*int enableMulti(int cs);*
Аргумент(ы):	*int cs – номера координатных систем*
Файл объявления:	*sys/sys.h*

Функция выполняет включение двигателей в слежение (включение и замыкание контура положения) в заданных координатных системах. В слежение будут включены двигатели, которые находятся в отключенном состоянии или работающие в режиме контура тока/момента.

Аргумент функции – битовое поле, в котором номера установленных битов (значения которых равны 1) соответствуют номерам координатных систем.

Возвращаемое значение равно 0 при отсутствии ошибок и отлично от 0 в противном случае.

Является системной.

3.10.2.62 hold

Синтаксис:	*int hold(int cs);*
Аргумент(ы):	*int cs – номер координатной системы*
Файл объявления:	*sys/sys.h*

Функция приостанавливает выполнение УП в координатной системе, номер которой определяется аргументом функции, уменьшая значение временной развертки координатной системы до 0.

Возвращаемое значение равно 0 при отсутствии ошибок и отлично от 0 в противном случае.

Является системной.

3.10.2.63 holdMulti

Синтаксис:	*int holdMulti(int cs);*
Аргумент(ы):	*int cs – номера координатных систем*
Файл объявления:	*sys/sys.h*

Функция приостанавливает выполнение УП в заданных координатных системах, уменьшая значение их временной развертки до 0.

Аргумент функции – битовое поле, в котором номера установленных битов (значения которых равны 1) соответствуют номерам координатных систем.

Возвращаемое значение равно 0 при отсутствии ошибок и отлично от 0 в противном случае.

Является системной.

3.10.2.64 pause

Синтаксис:	*int pause(int cs);*
Аргумент(ы):	*int cs – номер координатной системы*
Файл объявления:	*sys/sys.h*

Функция временно останавливает выполнение УП в координатной системе, номер которой определяется аргументом функции, в конце последнего вычисленного перемещения.

Возвращаемое значение равно 0 при отсутствии ошибок и отлично от 0 в противном случае.

Является системной.

3.10.2.65 pauseMulti

Синтаксис:	*int pauseMulti(int cs);*
Аргумент(ы):	*int cs – номера координатных систем*
Файл объявления:	*sys/sys.h*

Функция временно останавливает выполнение УП в заданных координатных системах в конце последнего рассчитанного перемещения.

Аргумент функции – битовое поле, в котором номера установленных битов (значения которых равны 1) соответствуют номерам координатных систем.

Возвращаемое значение равно 0 при отсутствии ошибок и отлично от 0 в противном случае.

Является системной.

3.10.2.66 resume

Синтаксис:	*int resume(int cs);*
Аргумент(ы):	*int cs – номер координатной системы*
Файл объявления:	*sys/sys.h*

Функция возобновляет выполнение временно остановленных УП в координатной системе, номер которой определяется аргументом функции, начиная с точки останова.

Возвращаемое значение равно 0 при отсутствии ошибок и отлично от 0 в противном случае.

Является системной.

3.10.2.67 resumeMulti

Синтаксис:	*int resumeMulti(int cs);*
Аргумент(ы):	*int cs – номера координатных систем*
Файл объявления:	*sys/sys.h*

Функция возобновляет выполнение временно остановленных УП в заданных координатных системах, начиная с точки останова.

Аргумент функции – битовое поле, в котором номера установленных битов (значения которых равны 1) соответствуют номерам координатных систем.

Возвращаемое значение равно 0 при отсутствии ошибок и отлично от 0 в противном случае.

Является системной.

3.10.2.68 run

Синтаксис:	*int run(int cs);*
Аргумент(ы):	*int cs – номер координатной системы*
Файл объявления:	*sys/sys.h*

Функция вызывает выполнение УП в координатной системе, номер которой определяется аргументом функции. Если выполнение УП было остановлено с помощью функций **hold**, **pause** или **step**, УП начнёт выполняться с той точки, где она была остановлена. Для перехода в начало УП следует предварительно вызвать функцию **begin**.

Возвращаемое значение равно 0 при отсутствии ошибок и отлично от 0 в противном случае.

Является системной.

3.10.2.69 runMulti

Синтаксис:	*int runMulti(int cs);*
Аргумент(ы):	*int cs – номера координатных систем*
Файл объявления:	*sys/sys.h*

Функция вызывает выполнение УП в заданных координатных системах. Если выполнение УП было остановлено с помощью функций **holdMulti**, **pauseMulti** или **stepMulti**, УП начнёт выполняться с той точки, где она была остановлена. Для перехода в начало УП следует предварительно вызвать функцию **beginMulti**.

Аргумент функции – битовое поле, в котором номера установленных битов (значения которых равны 1) соответствуют номерам координатных систем.

Возвращаемое значение равно 0 при отсутствии ошибок и отлично от 0 в противном случае.

Является системной.

3.10.2.70 begin

Синтаксис:	*int begin(int cs, double prog);*
Аргумент(ы):	*int cs – номер координатной системы,*
	double prog – номер программы движения
Файл объявления:	*sys/sys.h*

Функция устанавливает программу движения для координатной системы, номер которой определяется аргументом функции, и вызывает переход в начало заданной программы.

Возвращаемое значение равно 0 при отсутствии ошибок и отлично от 0 в противном случае.

Является системной.

3.10.2.71 beginMulti

Синтаксис:	*int beginMulti(int cs, double prog);*
Аргумент(ы):	*int cs – номера координатных систем,*
	double prog – номер программы движения
Файл объявления:	*sys/sys.h*

Функция устанавливает программу движения для координатных систем, номера которых определяются аргументом функции, и вызывает переход в начало заданной программы.

Первый аргумент функции `cs` – битовое поле, в котором номера установленных битов (значения которых равны 1) соответствуют номерам координатных систем. Второй аргумент `prog` – номер программы движения.

Возвращаемое значение равно 0 при отсутствии ошибок и отлично от 0 в противном случае.

Является системной.

3.10.2.72 start

Синтаксис:	*int start(int cs, double prog);*
Аргумент(ы):	*int cs – номер координатной системы,*
	double prog – номер программы движения
Файл объявления:	*sys/sys.h*

Функция устанавливает программу движения для координатной системы, номер которой определяется аргументом функции, вызывает переход в начало заданной программы и последующее её выполнение.

Возвращаемое значение равно 0 при отсутствии ошибок и отлично от 0 в противном случае.

Является системной.

3.10.2.73 startMulti

Синтаксис:	*int startMulti(int cs, double prog);*
Аргумент(ы):	*int cs – номера координатных систем,*
	double prog – номер программы движения
Файл объявления:	*sys/sys.h*

Функция устанавливает программу движения для координатных систем, номера которых определяются аргументом функции, вызывает переход в начало заданной программы и последующее её выполнение.

Первый аргумент функции `cs` – битовое поле, в котором номера установленных битов (значения которых равны 1) соответствуют номерам координатных систем. Второй аргумент `prog` – номер программы движения.

Возвращаемое значение равно 0 при отсутствии ошибок и отлично от 0 в противном случае.

Является системной.

3.10.2.74 step

Синтаксис:	*int step(int cs);*
Аргумент(ы):	*int cs – номер координатной системы*
Файл объявления:	*sys/sys.h*

Функция вызывает пошаговое выполнение УП в координатной системе, номер которой определяется аргументом функции.

Возвращаемое значение равно 0 при отсутствии ошибок и отлично от 0 в противном случае.

Является системной.

3.10.2.75 stepMulti

Синтаксис:	*int stepMulti(int cs);*
Аргумент(ы):	*int cs – номера координатных систем*
Файл объявления:	*sys/sys.h*

Функция вызывает пошаговое выполнение УП в заданных координатных системах.

Аргумент функции – битовое поле, в котором номера установленных битов (значения которых равны 1) соответствуют номерам координатных систем.

Возвращаемое значение равно 0 при отсутствии ошибок и отлично от 0 в противном случае.

Является системной.

3.10.2.76 stop

Синтаксис:	*int stop(int cs);*
Аргумент(ы):	*int cs – номер координатной системы*
Файл объявления:	*sys/sys.h*

Функция вызывает останов выполнения УП в координатной системе, номер которой определяется аргументом функции, позволяя завершить уже рассчитанные перемещения. После останова программы выполняется переход в её начало.

Возвращаемое значение равно 0 при отсутствии ошибок и отлично от 0 в противном случае.

Является системной.

3.10.2.77 stopMulti

Синтаксис:	*int stopMulti(int cs);*
Аргумент(ы):	*int cs – номера координатных систем*
Файл объявления:	*sys/sys.h*

Функция вызывает останов выполнения УП в заданных координатных системах, позволяя завершить уже рассчитанные перемещения. После останова программы выполняется переход в её начало.

Аргумент функции – битовое поле, в котором номера установленных битов (значения которых равны 1) соответствуют номерам координатных систем.

Возвращаемое значение равно 0 при отсутствии ошибок и отлично от 0 в противном случае.

Является системной.

3.10.2.78 suspend

Синтаксис:	*int suspend(int cs);*
Аргумент(ы):	*int cs – номер координатной системы*
Файл объявления:	*sys/sys.h*

Функция является аналогичной **pause**.

Является системной.

3.10.2.79 suspendMulti

Синтаксис:	*int suspendMulti(int cs);*
Аргумент(ы):	*int cs – номера координатных систем*
Файл объявления:	*sys/sys.h*

Функция является аналогичной **pauseMulti**.

Является системной.

3.10.2.80 bstart

Синтаксис:	*int bstart();*
Аргумент(ы):	*нет*
Файл объявления:	*sys/sys.h*

Функция указывает начало части программы, которая должна быть выполнена за один «шаг». Выполнение будет продолжаться до вызова функции **bstop**.

Возвращаемое значение равно 0 при отсутствии ошибок и отлично от 0 в противном случае.

Является системной.

3.10.2.81 bstop

Синтаксис:	*int bstop();*
Аргумент(ы):	*нет*
Файл объявления:	*sys/sys.h*

Функция указывает окончание части программы, которая должна быть выполнена за один «шаг».

Возвращаемое значение равно 0 при отсутствии ошибок и отлично от 0 в противном случае.

Является системной.

3.10.2.82 dread

Синтаксис:	*int dread(int cs, double *p);*
Аргумент(ы):	*int cs – номер координатной системы*
	*double *p – указатель на массив*
Файл объявления:	*sys/sys.h*

Функция выполняет расчёт и запись в массив заданной позиции для активных (имеющих определение) осей в координатной системе, номер которой определяется аргументом функции. Заданная позиция оси рассчитывается на основе заданной позиции двигателя, выражения определения оси и действующей матрицы преобразований.

Первый аргумент функции `cs` – номер координатной системы. Второй аргумент `*p` -– указатель на массив типа `double`, который должен содержать не менее 32 значений.

Возвращаемое значение равно 0 при отсутствии ошибок и отлично от 0 в противном случае.

Является системной.

3.10.2.83 pread

Синтаксис:	*int pread(int cs, double *p);*
Аргумент(ы):	*int cs – номер координатной системы*
	*double *p – указатель на массив*
Файл объявления:	*sys/sys.h*

Функция выполняет расчёт и запись в массив текущей позиции для активных (имеющих определение) осей в координатной системе, номер которой определяется аргументом функции. Текущая позиция оси рассчитывается на основе текущей позиции двигателя, выражения определения оси и действующей матрицы преобразований.

Первый аргумент функции `cs` – номер координатной системы. Второй аргумент `*p` -– указатель на массив типа `double`, который должен содержать не менее 32 значений.

Возвращаемое значение равно 0 при отсутствии ошибок и отлично от 0 в противном случае.

Является системной.

3.10.2.84 tread

Синтаксис:	*int tread(int cs, double *p);*
Аргумент(ы):	*int cs – номер координатной системы*
	*double *p – указатель на массив*
Файл объявления:	*sys/sys.h*

Функция выполняет расчёт и запись в массив конечной позиции выполняемого перемещения (выполняемого кадра) для активных (имеющих определение) осей в координатной системе, номер которой определяется аргументом функции.

Первый аргумент функции `cs` – номер координатной системы. Второй аргумент `*p` -– указатель на массив типа `double`, который должен содержать не менее 32 значений.

Возвращаемое значение равно 0 при отсутствии ошибок и отлично от 0 в противном случае.

Является системной.

3.10.2.85 dtogread

Синтаксис:	*int dtogread(int cs, double *p);*
Аргумент(ы):	*int cs – номер координатной системы*
	*double *p – указатель на массив*
Файл объявления:	*sys/sys.h*

Функция выполняет расчёт и запись в массив остатка пути выполняемого перемещения (выполняемого кадра) для активных (имеющих определение) осей в координатной системе, номер которой определяется аргументом функции. Остаток пути рассчитывается как разность конечной позиции выполняемого перемещения и текущего значения заданной позиции оси. Значение заданной позиции оси рассчитывается на основе заданной позиции двигателя, выражения определения оси и действующей матрицы преобразований. При активной коррекции инструмента значение конечной позиции выполняемого перемещения смещается на величину коррекции.

Первый аргумент функции `cs` – номер координатной системы. Второй аргумент `*p` -– указатель на массив типа `double`, который должен содержать не менее 32 значений.

Возвращаемое значение равно 0 при отсутствии ошибок и отлично от 0 в противном случае.

Является системной.

3.10.2.86 fread

Синтаксис:	*int fread(int cs, double *f);*
Аргумент(ы):	*int cs – номер координатной системы*
	*double *f – указатель на массив*
Файл объявления:	*sys/sys.h*

Функция выполняет расчёт и запись в массив ошибки слежения для активных (имеющих определение) осей в координатной системе, номер которой определяется аргументом функции. Ошибка слежения оси рассчитывается на основе ошибки слежения двигателя, выражения определения оси и действующей матрицы преобразований.

Первый аргумент функции `cs` – номер координатной системы. Второй аргумент `*f` -– указатель на массив типа `double`, который должен содержать не менее 32 значений.

Возвращаемое значение равно 0 при отсутствии ошибок и отлично от 0 в противном случае.

Является системной.

3.10.2.87 vread

Синтаксис:	*int vread(int cs, double *v);*
Аргумент(ы):	*int cs – номер координатной системы*
	*double *f – указатель на массив*
Файл объявления:	*sys/sys.h*

Функция выполняет расчёт и запись в массив текущей скорости (усреднённой за 16 сервоциклов) для активных (имеющих определение) осей в координатной системе, номер которой определяется аргументом функции. Скорость измеряется в единицах измерения перемещения по оси за мс.

Первый аргумент функции `cs` – номер координатной системы. Второй аргумент `*v` -– указатель на массив типа `double`, который должен содержать не менее 32 значений.

Возвращаемое значение равно 0 при отсутствии ошибок и отлично от 0 в противном случае.

Является системной.

3.10.2.88 pset

Синтаксис:	*int pset(const Pos &pos);*
Аргумент(ы):	*const **Pos** &pos – данные позиций*
Файл объявления:	*sys/sys.h*

Функция переопределяет текущую позицию осей и связанных с ними двигателей. Значения текущих заданных позиций становятся равными указанным значениям, поэтому никакого движения не происходит. Функция изменяет позицию нулевой точки двигателей (вводит смещение) и, таким образом, программные пределы перемещения и таблицу компенсаций.

Аргумент функции – структура **Pos**, в которой номера ячеек массива со значениями, отличными от NAN, соответствуют номерам осей, а сами значения ячеек являются текущими позициями.

Возвращаемое значение равно 0 при отсутствии ошибок и отлично от 0 в противном случае.

Является системной.

3.10.2.89 pstore

Синтаксис:	*int pstore();*
Аргумент(ы):	*нет*
Файл объявления:	*sys/sys.h*

Функция сохраняет смещения, которые вызваны **pset**.

Возвращаемое значение равно 0 при отсутствии ошибок и отлично от 0 в противном случае.

Является системной.

3.10.2.90 pload

Синтаксис:	*int pload();*
Аргумент(ы):	*нет*
Файл объявления:	*sys/sys.h*

Функция загружает смещения, которые сохранены **pstore**.

Действие функции аналогично **pset**.

Возвращаемое значение равно 0 при отсутствии ошибок и отлично от 0 в противном случае.

Является системной.

3.10.2.91 pclear

Синтаксис:	*int pclear();*
Аргумент(ы):	*нет*
Файл объявления:	*sys/sys.h*

Функция устанавливает текущую позицию осей и связанных с ними двигателей, равной 0. Значения текущих заданных позиций становятся равными указанным значениям, поэтому никакого движения не происходит. Функция изменяет позицию нулевой точки двигателей (вводит смещение) и, таким образом, программные пределы перемещения и таблицу компенсаций.

Действие функции аналогично **pset** с аргументами 0 или **homez** для двигателей.

Возвращаемое значение равно 0 при отсутствии ошибок и отлично от 0 в противном случае.

Является системной.

3.10.2.92 pmatch

Синтаксис:	*int pmatch();*
Аргумент(ы):	*нет*
Файл объявления:	*sys/sys.h*

Функция вызывает расчёт начальных позиций осей в координатной системе, чтобы они соответствовали текущим заданным позициям двигателей. Расчёт производится посредством выражений, обратных выражениям определений осей, или прямых кинематических преобразований.

Возвращаемое значение равно 0 при отсутствии ошибок и отлично от 0 в противном случае.

Является системной.

3.10.2.93 move

Синтаксис:	*int move(const Pos &pos);*
Аргумент(ы):	*const **Pos** &pos – данные перемещения*
Файл объявления:	*sys/sys.h*

Функция выполняет перемещение в указанное аргументом функции положение в установленном режиме движения.

Возвращаемое значение равно 0 при отсутствии ошибок и отлично от 0 в противном случае.

Является системной.

3.10.2.94 rapidmove

Синтаксис:	*int rapidmove(const Pos &pos);*
Аргумент(ы):	*const **Pos** &pos – данные перемещения*
Файл объявления:	*sys/sys.h*

Функция выполняет быстрое перемещение в указанное аргументом функции положение.

Возвращаемое значение равно 0 при отсутствии ошибок и отлично от 0 в противном случае.

Является системной.

3.10.2.95 rapid

Синтаксис:	*int rapid();*
Аргумент(ы):	*Нет*
Файл объявления:	*sys/sys.h*

Функция возвращает 1, если активен режим быстрых перемещений, и 0 в противном случае.

Является системной.

3.10.2.96 linearmove

Синтаксис:	*int linearmove(const Pos &pos);*
Аргумент(ы):	*const **Pos** &pos – данные перемещения*
Файл объявления:	*sys/sys.h*

Функция выполняет линейное перемещение в указанное аргументом функции положение.

Возвращаемое значение равно 0 при отсутствии ошибок и отлично от 0 в противном случае.

Является системной.

3.10.2.97 linear

Синтаксис:	*int linear();*
Аргумент(ы):	*Нет*
Файл объявления:	*sys/sys.h*

Функция возвращает 1, если активен режим линейной интерполяции, и 0 в противном случае.

Является системной.

3.10.2.98 cir1move

Синтаксис:	*int cir1move(const Pos &pos);*
Аргумент(ы):	*const **Pos** &pos – данные перемещения*
Файл объявления:	*sys/sys.h*

Функция выполняет круговое перемещение по часовой стрелке в указанное аргументом функции положение.

Возвращаемое значение равно 0 при отсутствии ошибок и отлично от 0 в противном случае.

Является системной.

3.10.2.99 cir2move

Синтаксис:	*int cir2move(const Pos &pos);*
Аргумент(ы):	*const **Pos** &pos – данные перемещения*
Файл объявления:	*sys/sys.h*

Функция выполняет круговое перемещение против часовой стрелки в указанное аргументом функции положение.

Возвращаемое значение равно 0 при отсутствии ошибок и отлично от 0 в противном случае.

Является системной.

3.10.2.100 circle1

Синтаксис:	*int circle1();*
Аргумент(ы):	*нет*
Файл объявления:	*sys/sys.h*

Функция устанавливает режим круговой интерполяции по часовой стрелке для основной декартовой системы координат (X/Y/Z).

Возвращаемое значение равно 0 при отсутствии ошибок и отлично от 0 в противном случае.

Является системной.

3.10.2.101 circle2

Синтаксис:	*int circle2();*
Аргумент(ы):	*нет*
Файл объявления:	*sys/sys.h*

Функция устанавливает режим круговой интерполяции против часовой стрелки для основной декартовой системы координат (X/Y/Z).

Возвращаемое значение равно 0 при отсутствии ошибок и отлично от 0 в противном случае.

Является системной.

3.10.2.102 circle3

Синтаксис:	*int circle3();*
Аргумент(ы):	*нет*
Файл объявления:	*sys/sys.h*

Функция устанавливает режим круговой интерполяции по часовой стрелке для расширенной декартовой системы координат (XX/XY/XZ).

Возвращаемое значение равно 0 при отсутствии ошибок и отлично от 0 в противном случае.

Является системной.

3.10.2.103 circle4

Синтаксис:	*int circle4();*
Аргумент(ы):	*нет*
Файл объявления:	*sys/sys.h*

Функция устанавливает режим круговой интерполяции против часовой стрелки для расширенной декартовой системы координат (XX/XY/XZ).

Возвращаемое значение равно 0 при отсутствии ошибок и отлично от 0 в противном случае.

Является системной.

3.10.2.104 pvt

Синтаксис:	*int pvt(double time);*
Аргумент(ы):	*double time – время перемещения*
Файл объявления:	*sys/sys.h*

Функция устанавливает режим движения с заданными положением, скоростью и временем (pvt-движение). Если данный режим уже установлен, то изменяется время перемещения. Если установлен другой режим движения (линейная или круговая интерполяция, быстрые перемещения, сплайновая интерполяция), то будет выполнен выход из него. Время перемещения измеряется в мс.

Возвращаемое значение равно 0 при отсутствии ошибок и отлично от 0 в противном случае.

Является системной.

3.10.2.105 spline

Синтаксис:	*int spline(double time);*
Аргумент(ы):	*double time – время сегмента перемещения*
Файл объявления:	*sys/sys.h*

Функция устанавливает режим сплайновой интерполяции. Если данный режим уже установлен, то изменяется время сегмента перемещения. Если установлен другой режим движения (линейная или круговая интерполяция, быстрые перемещения, pvt-движение), то будет выполнен выход из него. Время сегмента перемещения измеряется в мс.

Возвращаемое значение равно 0 при отсутствии ошибок и отлично от 0 в противном случае.

Является системной.

3.10.2.106 ccmode1

Синтаксис:	*int ccmode1();*
Аргумент(ы):	*нет*
Файл объявления:	*sys/sys.h*

Функция отменяет двухмерную и трёхмерную коррекцию радиуса инструмента, уменьшая её постепенно на последующем линейном перемещении. Является эквивалентом G40.

Возвращаемое значение равно 0 при отсутствии ошибок и отлично от 0 в противном случае.

Является системной.

3.10.2.107 ccmode2

Синтаксис:	*int ccmode2();*
Аргумент(ы):	*нет*
Файл объявления:	*sys/sys.h*

Функция включает двухмерную коррекцию радиуса инструмента влево, вводя её постепенно на последующем линейном перемещении. Является эквивалентом G41.

Возвращаемое значение равно 0 при отсутствии ошибок и отлично от 0 в противном случае.

Является системной.

3.10.2.108 ccmode3

Синтаксис: *int ccmode3();*
Аргумент(ы): *нет*
Файл объявления: *sys/sys.h*

Функция включает двухмерную коррекцию радиуса инструмента вправо, вводя её постепенно на последующем линейном перемещении. Является эквивалентом G42.

Возвращаемое значение равно 0 при отсутствии ошибок и отлично от 0 в противном случае.

Является системной.

3.10.2.109 ccmode4

Синтаксис: *int ccmode4();*
Аргумент(ы): *нет*
Файл объявления: *sys/sys.h*

Функция включает трёхмерную коррекцию радиуса инструмента, вводя её постепенно на последующем линейном перемещении.

Возвращаемое значение равно 0 при отсутствии ошибок и отлично от 0 в противном случае.

Является системной.

3.10.2.110 ccr

Синтаксис: *int ccr(double r);*
Аргумент(ы): *double r – радиус инструмента*
Файл объявления: *sys/sys.h*

Функция задаёт величину радиуса инструмента для двухмерной коррекции. Траектория центра инструмента будет смещена на данное расстояние перпендикулярно запрограммированной траектории в заданной плоскости коррекции.

Возвращаемое значение равно 0 при отсутствии ошибок и отлично от 0 в противном случае.

Является системной.

3.10.2.111 txyz

Синтаксис: *int txyz(Vec v);*
Аргумент(ы): ***Vec** v – вектор ориентации инструмента*
Файл объявления: *sys/sys.h*

Функция задаёт вектор ориентации инструмента для трёхмерной коррекции. Компоненты вектора I, J, K параллельны осям X, Y, Z соответственно. Геометрическая сумма компонент определяет направление вектора (от основания к концу или от конца к основанию), длина вектора не имеет значения.

Возвращаемое значение равно 0 при отсутствии ошибок и отлично от 0 в противном случае.

Является системной.

3.10.2.112 txyzScale

Синтаксис:	*int txyzScale(double s);*
Аргумент(ы):	*double s – масштабный коэффициент*
Файл объявления:	*sys/sys.h*

Функция задаёт масштабный коэффициент для вектора подачи и радиуса инструмента при двухмерной коррекции, отличный от коэффициентов масштабирования матрицы преобразования. Предназначен для сохранения подачи и радиуса в непреобразованных единицах измерения перемещения по осям.

Возвращаемое значение равно 0 при отсутствии ошибок и отлично от 0 в противном случае.

Является системной.

3.10.2.113 nxyz

Синтаксис:	*int nxyz(const Vec &v);*
Аргумент(ы):	*const **Vec** v – вектор нормали к поверхности*
Файл объявления:	*sys/sys.h*

Функция задаёт вектор нормали к поверхности для трёхмерной коррекции. Компоненты вектора I, J, K параллельны осям X, Y, Z соответственно. Геометрическая сумма компонент определяет направление вектора от поверхности детали к инструменту, длина вектора не имеет значения. Вектор должен быть определен в базовых машинных координатах.

Возвращаемое значение равно 0 при отсутствии ошибок и отлично от 0 в противном случае.

Является системной.

3.10.2.114 normal

Синтаксис:	*int normal(const Vec &v);*
Аргумент(ы):	*const **Vec** v – вектор нормали к рабочей плоскости*
Файл объявления:	*sys/sys.h*

Функция задаёт вектор нормали к рабочей плоскости (перпендикулярный рабочей плоскости) для круговой интерполяции, двухмерной коррекции. Компоненты вектора I, J, K параллельны осям X, Y, Z соответственно. Геометрическая сумма компонент определяет направление вектора и, следовательно, положение рабочей плоскости. От

ориентации вектора зависит направление перемещения по дуге окружности и коррекции инструмента (используется правосторонняя система координат). Длина вектора не имеет значения.

Возвращаемое значение равно 0 при отсутствии ошибок и отлично от 0 в противном случае.

Является системной.

3.10.2.115 tsel

Синтаксис:	*int tsel(int id);*
Аргумент(ы):	*int id – номер матрицы преобразования*
Файл объявления:	*sys/sys.h*

Функция задаёт номер активной матрицы преобразования для координатной системы УП. Диапазон действительных номеров – 0÷255. Значение номера, равное -1, отменяет выбор всех матриц преобразования.

Возвращаемое значение равно 0 при отсутствии ошибок и отлично от 0 в противном случае.

Является системной.

3.10.2.116 wait

Синтаксис:	*void wait();*
Аргумент(ы):	*нет*
Файл объявления:	*sys/sys.h*

Функция приостанавливает расчеты до следующего прерывания реального времени. Предназначена для защиты от срабатывания сторожевого таймера.

Является системной.

3.11 Переменные и буфера

3.11.1 Функции

3.11.1.1 detectEdgeRise

Синтаксис:	*int detectEdgeRise(int &detector, int input);*
Аргумент(ы):	*int &detector – сохранённое входное значение*
	int input – текущее входное значение
Файл объявления:	*include/func/misc.h*

Функция служит для детектирования изменения с 0 на 1 (детектирования фронта) входной величины.

Функция возвращает 0, если входное значение не изменилось и осталось равным 0, и 1, если входное значение стало отличным от 0.

Является системной.

3.11.1.2 detectEdgeFall

Синтаксис:	*int detectEdgeFall(int &detector, int input);*
Аргумент(ы):	*int &detector – сохранённое входное значение*
	int input – текущее входное значение
Файл объявления:	*include/func/misc.h*

Функция служит для детектирования изменения с 1 на 0 (детектирования спада) входной величины.

Функция возвращает 0, если входное значение не изменилось и осталось равным 1, и 1, если входное значение стало равным 0.

Является системной.

3.11.1.3 syncset

Синтаксис:	*void syncset(void *ptr, int value);*
Аргумент(ы):	*void *ptr – указатель на переменную*
	int value – присваиваемое значение
Файл объявления:	*sys/sys.h*

Функция выполняет синхронное присваивание значения типа `int`, адресуемой указателем переменной. Синхронное присваивание осуществляется в момент начала следующего перемещения.

Первый аргумент функции `*ptr` – указатель, ссылающийся на переменную. Второй аргумент `value` -– присваиваемое значение.

Является системной.

3.11.1.4 syncsetf

Синтаксис:	*void syncsetf(void *ptr, float value);*
Аргумент(ы):	*void *ptr – указатель на переменную*
	float value – присваиваемое значение
Файл объявления:	*sys/sys.h*

Синхронное присваивание значения типа `float`, адресуемой указателем переменной. Синхронное присваивание осуществляется в момент начала следующего перемещения.

Первый аргумент функции `*ptr` – указатель, ссылающийся на переменную. Второй аргумент `value` -– присваиваемое значение.

Является системной.

3.11.1.5 syncsetd

Синтаксис:	*void syncsetd(void *ptr, double value);*
Аргумент(ы):	*void *ptr – указатель на переменную*
	double value – присваиваемое значение
Файл объявления:	*sys/sys.h*

Синхронное присваивание значения типа `double`, адресуемой указателем переменной. Синхронное присваивание осуществляется в момент начала следующего перемещения.

Первый аргумент функции `*ptr` – указатель, ссылающийся на переменную. Второй аргумент `value` -– присваиваемое значение.

Является системной.

3.11.1.6 usave

Синтаксис:	*void usave(void *ptr);*
Аргумент(ы):	*void *ptr – указатель на переменную*
Файл объявления:	*sys/sys.h*

Функция выполняет сохранение пользовательской переменной, на которую ссылается указатель. Пользовательская переменная должна быть объявлена посредством макроса `#define USER_SAVE(name)`.

Является системной.

3.11.1.7 clearGather

Синтаксис:	*int clearGather();*
Аргумент(ы):	*нет*
Файл объявления:	*sys/sys.h*

Функция очищает буфер данных сервопрерываний.

Является системной.

3.11.1.8 clearPhaseGather

Синтаксис:	*int clearPhaseGather();*
Аргумент(ы):	*нет*
Файл объявления:	*sys/sys.h*

Функция очищает буфер данных фазных прерываний.

Является системной.

3.12 Таймеры

3.12.1 Типы данных

3.12.1.1 Timer

Тип данных:	*Структура Timer*
Файл объявления:	*sys/sys.h*

Структура определяет параметры таймера.

Таблица 3.52. Структура Timer

Элемент	Тип	Описание
start	int	Начальное значение счётчика таймера
timeout	int	Интервал

3.12.2 Функции и макросы

3.12.2.1 timerStart

Синтаксис: *timerStart(timer, timeoutVal);*
Аргумент(ы): *timer – переменная типа **Timer***
timeoutVal – интервал срабатывания
Файл объявления: *sys/sys.h*

Макрос запускает таймер, инициализируя переменную `timer`: полю timer.start присваивается текущее значение системного счётчика, полю timer.timeout – значение интервала срабатывания.

Интервал срабатывания таймера задаётся в периодах сервоцикла (1 период сервоцикла равен 400 мс). Так, например, 1 с соответствует значению интервала равному 2500.

Является системной.

3.12.2.2 timerTimeout

Синтаксис: *timerTimeout(timer);*
Аргумент(ы): *timer – переменная типа **Timer***
Файл объявления: *sys/sys.h*

Макрос возвращает 0, если не истёк заданный интервал срабатывания, и значение, отличное от 0, в противном случае.

Является системной.

3.12.2.3 timerLeft

Синтаксис: *timerLeft(timer);*
Аргумент(ы): *timer – переменная типа **Timer***
Файл объявления: *sys/sys.h*

Макрос возвращает число периодов сервоцикла, оставшихся до срабатывания таймера.

Является системной.

3.12.2.4 timerPassed

Синтаксис:	*timerPassed(timer);*
Аргумент(ы):	*timer – переменная типа **Timer***
Файл объявления:	*sys/sys.h*

Макрос возвращает число периодов сервоцикла, прошедших с момента запуска таймера.

Является системной.

3.12.2.5 initPulsedTimer

Синтаксис:	*void initPulsedTimer();*
Аргумент(ы):	*Нет*
Файл объявления:	*include/func/misc.h*

Функция инициализации периодического (импульсного) таймера.

Является системной.

3.12.2.6 timerSc

Синтаксис:	*int timerSc(int period);*
Аргумент(ы):	*int period – период таймера*
Файл объявления:	*include/func/misc.h*

Функция периодического (импульсного) таймера – таймера, выходное значение которого периодически переключается с 0 на 1 и обратно через интервал, равный половине периода таймера. Период таймера задаётся в периодах сервоцикла (1 период сервоцикла равен 400 мс). Так, например, интервал 1 с соответствует значению периода таймера равному 2500.

Функция возвращает 1, если с момента переключения таймера с 1 на 0 истёк интервал, больший или равный половине периода, и 0 в противном случае.

Является системной.

3.13 Управление программами ПЛК

3.13.1 Функции

3.13.1.1 enablePLC

Синтаксис:	*int enablePLC(int plc);*
Аргумент(ы):	*int plc – номер программы ПЛК*
Файл объявления:	*sys/sys.h*

Функция вызывает выполнение программы ПЛК, номер которой (от 0 до 31) определяется аргументом функции. Выполнение стартует с начала программы.

Возвращаемое значение равно 0 при отсутствии ошибок и отлично от 0 в противном случае.

Является системной.

3.13.1.2 enablePLCs

Синтаксис:	*int enablePLCs(int plc);*
Аргумент(ы):	*int plc – номера программ ПЛК*
Файл объявления:	*sys/sys.h*

Функция вызывает выполнение программ ПЛК, номера которых (от 0 до 31) определяются аргументом функции. Аргумент функции – битовое поле, в котором номера установленных битов (значения которых равны 1) соответствуют номерам программ ПЛК. Выполнение стартует с начала программы.

Возвращаемое значение равно 0 при отсутствии ошибок и отлично от 0 в противном случае.

Является системной.

3.13.1.3 pausePLC

Синтаксис:	*int pausePLC(int plc);*
Аргумент(ы):	*int plc – номер программы ПЛК*
Файл объявления:	*sys/sys.h*

Функция вызывает временный останов программы ПЛК, номер которой (от 0 до 31) определяется аргументом функции.

Возвращаемое значение равно 0 при отсутствии ошибок и отлично от 0 в противном случае.

Является системной.

3.13.1.4 pausePLCs

Синтаксис:	*int pausePLCs(int plc);*
Аргумент(ы):	*int plc – номера программ ПЛК*
Файл объявления:	*sys/sys.h*

Функция вызывает временный останов программ ПЛК, номера которых (от 0 до 31) определяются аргументом функции. Аргумент функции – битовое поле, в котором номера установленных битов (значения которых равны 1) соответствуют номерам программ ПЛК.

Возвращаемое значение равно 0 при отсутствии ошибок и отлично от 0 в противном случае.

Является системной.

3.13.1.5 resumePLC

Синтаксис:	*int resumePLC(int plc);*
Аргумент(ы):	*int plc – номер программы ПЛК*
Файл объявления:	*sys/sys.h*

Функция вызывает возобновление выполнения программы ПЛК, номер которой (от 0 до 31) определяется аргументом функции.

Возвращаемое значение равно 0 при отсутствии ошибок и отлично от 0 в противном случае.

Является системной.

3.13.1.6 resumePLCs

Синтаксис:	*int resumePLCs(int plc);*
Аргумент(ы):	*int plc – номера программ ПЛК*
Файл объявления:	*sys/sys.h*

Функция вызывает возобновление выполнения программ ПЛК, номера которых (от 0 до 31) определяются аргументом функции. Аргумент функции – битовое поле, в котором номера установленных битов (значения которых равны 1) соответствуют номерам программ ПЛК.

Возвращаемое значение равно 0 при отсутствии ошибок и отлично от 0 в противном случае.

Является системной.

3.13.1.7 disablePLC

Синтаксис:	*int disablePLC(int plc);*
Аргумент(ы):	*int plc – номер программы ПЛК*
Файл объявления:	*sys/sys.h*

Функция вызывает отмену выполнения программы ПЛК, номер которой (от 0 до 31) определяется аргументом функции. Возвращаемое значение равно 0 при отсутствии ошибок и отлично от 0 в противном случае.

Является системной.

3.13.1.8 disablePLCs

Синтаксис:	*int disablePLCs(int plc);*
Аргумент(ы):	*int plc – номера программ ПЛК*
Файл объявления:	*sys/sys.h*

Функция вызывает отмену выполнения программ ПЛК, номера которых (от 0 до 31) определяются аргументом функции. Аргумент функции – битовое поле, в котором номера установленных битов (значения которых равны 1) соответствуют номерам программ ПЛК.

Возвращаемое значение равно 0 при отсутствии ошибок и отлично от 0 в противном случае.

Является системной.

3.13.1.9 stepPLC

Синтаксис:	*int stepPLC(int plc);*
Аргумент(ы):	*int plc – номер программы ПЛК*
Файл объявления:	*sys/sys.h*

Функция вызывает пошаговое выполнение программы ПЛК, номер которой (от 0 до 31) определяется аргументом функции.

Возвращаемое значение равно 0 при отсутствии ошибок и отлично от 0 в противном случае.

Является системной.

3.13.1.10 stepPLCs

Синтаксис:	*int stepPLCs(int plc);*
Аргумент(ы):	*int plc – номера программ ПЛК*
Файл объявления:	*sys/sys.h*

Функция вызывает пошаговое выполнение программ ПЛК, номера которых (от 0 до 31) определяются аргументом функции. Аргумент функции – битовое поле, в котором номера установленных битов (значения которых равны 1) соответствуют номерам программ ПЛК.

Возвращаемое значение равно 0 при отсутствии ошибок и отлично от 0 в противном случае.

Является системной.

4 . Реализация программ ПЛК

4.1 Таймеры

4.1.1 Таймер однократного запуска

Таймер однократного запуска (не периодический) – таймер, выходное значение которого становится равным 1 по истечении заданного интервала и не меняется до повторного запуска (таймер, отмеряющий заданный интервал времени с момента запуска).

Для использования таймера однократного запуска необходимо объявить переменную типа **Timer**.

Запуск таймера осуществляется вызовом макроса **timerStart**, аргументами которого являются переменная типа **Timer** и величина интервала срабатывания в периодах сервоцикла.

Для проверки срабатывания таймера предназначен макрос **timerTimeout**, который возвращает значение, отличное от 0, если истёк заданный интервал срабатывания.

Перезапуск таймера осуществляется повторным вызовом макроса **timerStart**.

Временная диаграмма таймера представлена на рис. 4.1.

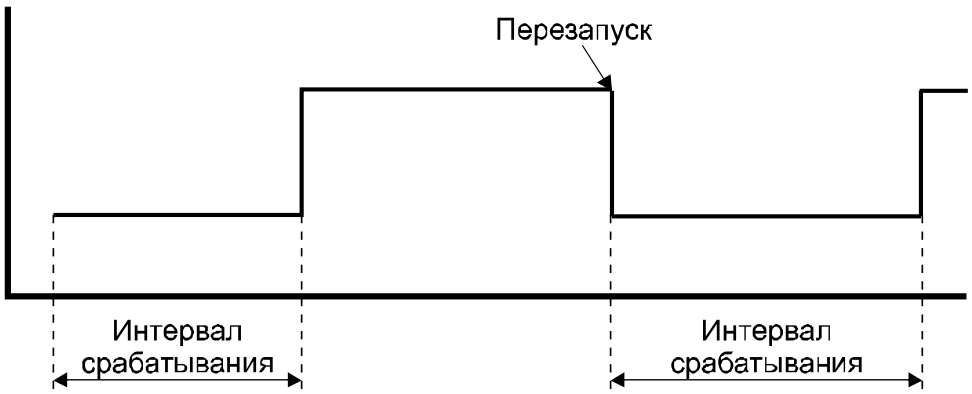

Рис. 4.1. Таймер однократного запуска

Листинг 4.4 на стр. 144 иллюстрирует использование таймера однократного запуска.

4.1.2 Периодический таймер

Периодический (импульсный) таймер – таймер, выходное значение которого периодически переключается с 0 на 1 и обратно через интервал, равный половине заданного периода таймера.

Функция **timerSc** возвращает 1, если с момента переключения таймера с 1 на 0 истёк интервал, больший или равный половине периода таймера, и 0 в противном случае.

Временная диаграмма таймера представлена на рис. 4.1.

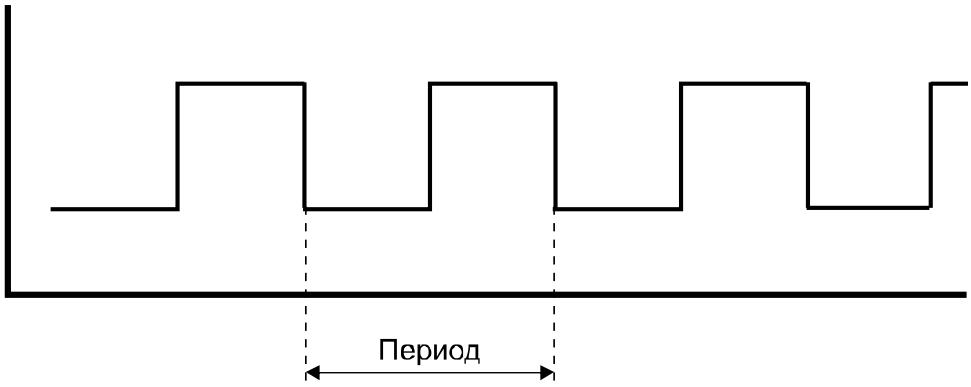

Рис. 4.2. Периодический таймер

Периодический таймер используется для организации равных промежутков времени, например, для мигающей индикации или разного рода фильтров.

Листинг 4.1 показывает применение периодического таймера для индикации пульта оператора.

Переменная `homeLed` (строка 5) определяет состояние индикатора реферирования осей. Если не выполнен выезд в 0 или не выполнено позиционирование всех осей при включении станка, то переменной `homeLed` будет присваиваться периодически 0 или 1. Значение `homeLed` записывается в `mt.PultOut.modeHome` (строка 19) и индикатор реферирования осей пульта оператора будет мигать.

Листинг 4.1. Применение периодического таймера

```
1 void mtUpdateCNCIndication() { // Функция обновления индикации пульта
2     mt.PultOut.PultLed[0] = 0; // Обнуление регистров индикации пульта
3     mt.PultOut.PultLed[1] = 0;
4   ...
5   // Если не завершено реферирование или не выполнено позиционирование –
6     // мигание индикатора
7     int homeLed = (!isHomeComplete() || !axesRefPosComplete()) &&
        timerSc(TIMER_HOME_INCOMPLETE);
```

```
8     int resetLed = (mt.ncNotReadyReq || CNC.notReadyReq) &&
        timerSc(TIMER_ERRORS);
9     int startLed = 0;
10    int stopLed = 0;
11    int mode = CNC.mode;
12
13    if (mode == cncWaitChangeMode) {
14        mode = CNC.prevMode;
15    }
16
17    switch (mode) {
18  // Обновления индикации пульта в ручном режиме
19    case cncManual:
20        mt.PultOut.modeManual = 1;
21        mt.PultOut.modeAuto = 0;
22        mt.PultOut.modeHome = homeLed;
23        mt.PultOut.modeHWL = 0;
24        mt.PultOut.modeMDI = 0;
25        mt.PultOut.modeReset = resetLed;
26        mt.PultOut.modeStep = 0;
27        mt.PultOut.modeReposToContour = 0;
28        mt.PultOut.speed1 = axesControl.platform.speedSelect == 0;
29        mt.PultOut.speed2 = axesControl.platform.speedSelect == 1;
30        mt.PultOut.speed3 = axesControl.platform.speedSelect == 2;
31        mt.PultOut.speed4 = axesControl.platform.speedSelect == 3;
32        mt.PultOut.rapid  = axesControl.platform.speedSelect == -1;
33        break;
34
35  // Обновления индикации пульта в режиме выезда в нулевую точку
36  case cncHome:
37        mt.PultOut.modeManual  = 0;
38        mt.PultOut.modeAuto    = 0;
39        mt.PultOut.modeHome    = 1;
40        mt.PultOut.modeHWL     = 0;
41        mt.PultOut.modeMDI     = 0;
42        mt.PultOut.modeReset   = resetLed;
43        mt.PultOut.modeStep    = 0;
44        mt.PultOut.modeReposToContour = 0;
45        startLed = isHoming();
46        break;
47  ...
48  }
49 }
```

4.2 Входы/выходы

Обращение ко входам и выходам осуществляется через 4-х байтовые переменные Servo[i].IO[j].DataIn[k] и Servo[i].IO[j].DataOut[k] соответственно, где i – номер платы управления, j – номер порта платы управления, k – номер регистра выходных/выходных данных.

Переменные Servo[i].IO[j].DataIn[k] представляют собой набор 1-но битных полей, каждое из которых содержит состояние отдельного входа.

Переменные Servo[i].IO[j].DataOut[k] представляют собой набор 1-но битных полей, в каждое из которых записывается состояние отдельного выхода.

Для работы со входами и выходами пользователем объявляются объединения:

- MTInputs – выходные сигналы электрооборудования станка (путь по умолчанию: include\platform\имя_проекта\stanok.h);
- MTOutputs – входные сигналы электрооборудования станка (путь по умолчанию: include\platform\имя_проекта\stanok.h);
- PultInputs – выходные сигналы пульта оператора (путь по умолчанию: include\platform\имя_проекта\operator_pult.h);
- PultOutputs – входные сигналы пульта оператора (путь по умолчанию: include\platform\имя_проекта\operator_pult.h);
- PortablePultInputs – выходные сигналы переносного пульта (путь по умолчанию: include\platform\имя_проекта\portable_pult.h).

Для указания функционального назначения отдельного входа или выхода следует определить имена идентификаторов соответствующего битового поля в объявлении объединений.

Листинг 4.2 показывает пример определения имён идентификаторов в объявлении объединений `MTInputs` и `MTOutputs`.

Листинг 4.2. Пример определения имён идентификаторов

```
union MTInputs {
  struct {
   // Первая плата входов
   unsigned chillerLubeLevelLow:1;  // Вход 0 – низкий уровень масла в охладителе
      шпинделя
   unsigned clearCoolantOverload:1; // Вход 1 – перегрузка очистителя СОЖ
   unsigned onFanSPND:1; // Вход 2 – вентилятор шпинделя включён
   unsigned overloadPumpA:1;  // Вход 3 – перегрузка насоса СОЖ А
   unsigned mtOn:1;  // Вход 4 – станок включён
   unsigned reserved1:26;
   unsigned controlServiceDoor:1;  // Вход 31 – контроль сервисной двери
   //Вторая плата входов
   unsigned overloadChipConv:1;  // Вход 0  – перегрузка щипкового конвеера
```

```
    unsigned lubeLevelLow:1;   // Вход 1  – низкий уровень масла
    unsigned coolantFilterUnclean:1; // Вход 2 – фильтр СОЖ засорён
    unsigned unclampingAxisC:1;   // Вход 3  – ось C разжата
    unsigned reserved2:27;
    unsigned coolantLevelLow:1;   // Вход 31 – СОЖ низкий уровень
   };
   int Inputs[2];
};

union MTOutputs {
   struct {
    // Первая плата реле 24 выхода
    unsigned clearCoolantOn:1;   // Выход 0 – включение очистки СОЖ от масла
    unsigned chipConvOn:1;   // Выход 1 – включение конвейера стружки
    unsigned autoLubeOn:1; // Выход 2 платы – включение автоматической смазки ШВП
    unsigned workingLight:1; // Выход 3 – включение рабочего освещения
    unsigned workpieceBlast:1;   // Выход 4 – обдув рабочей зоны
    unsigned reserved1:18;
    unsigned operatorDoorOpen:1; // Выход 23 – открытие двери оператора
    // Вторая плата реле 8 выходов
    unsigned clampingAxisA:1;   // Выход 0 – зажим оси A
    unsigned pumpA:1;   // Выход 1  – включение насоса A подачи СОЖ
    unsigned screwAugerCW:1; // Выход 2 платы  – вращение шнекового транспортёра по часовой стрелке
    unsigned screwAugerCCW:1; // Выход 3 платы – вращение шнекового транспортёра против часовой стрелки
    unsigned pumpA:1;   // Выход 4  – включение насоса A подачи СОЖ
    unsigned spindleChiller:1;   // Выход 5 – включение охлаждения шпинделя
    unsigned reserved2:1;
    unsigned unclampingAxisA:1;   // Выход 7 – разжим оси A
   };
   int Outputs[2];
};
```

Плата входов имеет возможность подключения 32 входных сигналов, плата выходов имеет возможность вывода до 24 выходных сигналов.

Перечисленные выше объединения являются полями структуры **MTDesc**.

Пользователь должен объявить переменную `mt` типа **MTDesc** для возможности работы со входами и выходами.

Перед чтением состояний входов необходимо проверить корректность полученных данных: 1-й бит переменной `Servo[i].IO[j].Status` должен быть установлен (равен 1).

Листинг 4.3 показывает обращение ко входам и выходам. В данном примере пульт оператора подключён к порту №0, плата входов/выходов для управления электроавтоматикой станка – к порту №1.

Листинг 4.3. Обращение ко входам и выходам

```
MTDesc mt;   // Объявление структуры с данными станка
...
void readInputs() {     // Функция чтения данных входных регистров
    if (Servo[0].IO[0].Status & 1) { // Проверка корректности данных
    // Чтение регистров пульта оператора
        mt.PultIn.PultBtn[0] = Servo[0].IO[0].DataIn[0];
        mt.PultIn.PultBtn[1] = Servo[0].IO[0].DataIn[1];
        mt.PultIn.PultBtn[2] = Servo[0].IO[0].DataIn[2];
        mt.PultIn.PultBtn[3] = Servo[0].IO[0].DataIn[3];
    // Обнуление числа ошибок соединения с пультом оператора
        countErrorLinkOperatorPult = 0;
    } else {
        countErrorLinkOperatorPult++; // Инкрементирование числа ошибок
        if (countErrorLinkOperatorPult >= 100)
      // Ошибка соединения с пультом оператора
            errorSet(systemErrors.machine.linkOperatorPult);
    }

    if (Servo[0].IO[1].Status & 1) { // Проверка корректности данных
    //Чтение регистров платы входов/выходов
        mt.IN.Inputs[0] = Servo[0].IO[1].DataIn[0];
    // Обнуление числа ошибок соединения с платой входов
        countErrorLinkIntIO = 0;
    } else {
        countErrorLinkIntIO++; // Инкрементирование числа ошибок
        if (countErrorLinkIntIO >= 100)
      // Ошибка соединения с платой входов
            errorSet(systemErrors.machine.linkIntIO);
    }
}

void writeOutputs() {  // Функция записи данных в выходные регистры
    // Запись в регистры пульта оператора
    Servo[0].IO[0].DataOut[0] = mt.PultOut.PultLed[0];
    Servo[0].IO[0].DataOut[1] = mt.PultOut.PultLed[1];
    Servo[0].IO[0].DataOut[2] = mt.PultOut.PultLed[2];
    // Запись в регистры платы входов/выходов
    Servo[0].IO[1].DataOut[0] = mt.OUT.Outputs[0];
}
```

4.3 Программирование алгоритмов управления

Для программирования алгоритмов управления используются конечные автоматы.

Конечные автоматы – конструкции, которые описываются ограниченным набором возможных состояний, набором сигналов (событий) и условиями переходов из одного состояния в другое. Последующее состояние автомата определяется текущим состоянием и входными сигналами.

Листинг 4.4 показывает фрагмент реализации конечного автомата включения/выключения станка с помощью оператора множественного выбора `switch-case`. Полностью автомат приведён в **ПРИЛОЖЕНИИ 1** в листинге «Программа включения/выключения станка» на стр. 161.

Листинг 4.4. Фрагмент реализации конечного автомата управления станком

```
switch (mt.State) {
case mtNotReady: {    // Ожидание включения главного пускателя
  // Проверка наличия команды сброса в начальное состояние
  if (CNC.request == mtcncReset) { mtReset(); } // Сброс в начальное состояние
  // Станок включен и система готова?
  if (mtIsOn() && !mt.ncNotReadyReq) mt.State=mtStartOn;
  break;
}

case mtStartOn: {    // Начало включения
  // Проверка наличия команды сброса в начальное состояние
  if (CNC.request == mtcncReset) { mtReset(); }
  // Проверка готовности системы
  if (mt.ncNotReadyReq) { mtAbortRequest(); break; } // Аварийное выключение
  mt.State = mtDriveOn;
  // Фазировка выполнена?
  if (!axesPhaseRefComplete() || !spinsPhaseRefComplete()) mt.State = mtPhaseRef;
  break;
}

case mtPhaseRef: {    // Фазировка
  // Проверка наличия команды сброса в начальное состояние
  if (CNC.request == mtcncReset) { mtReset(); } // Сброс в начальное состояние
  // Проверка готовности системы
  if (mt.ncNotReadyReq) { mtAbortRequest(); break; } // Аварийное выключение
  if (axesPhaseRef() || spinsPhaseRef()) { // Требуется фазировка?
    mt.State = mtWaitPhaseRef;
```

```
28      // Запуск таймера фазировки
29      timerStart(mt.timerState, MT_TIME_DRIVE_PHASE_REF);
30    } else {
31      mt.State = mtDriveOn; // Фазировка выполнена
32    }
33    break;
34  }
35
36  case mtWaitPhaseRef: {    // Ожидание окончания фазировки
37    // Проверка наличия команды сброса в начальное состояние
38    if (CNC.request == mtcncReset) { mtReset(); } // Сброс в начальное состояние
39    // Проверка готовности системы
40    if (mt.ncNotReadyReq) { mtAbortRequest(); break; } // Аварийное выключение
41    if (timerTimeout(mt.timerState)) {
42      // Ошибка: истекло время фазировки
43      errorSet(systemErrors.channel[0].phaseRefTimeout);
44      break;
45    }
46    // Фазировка выполнена?
47    if (axesPhaseRefComplete() && spinsPhaseRefComplete()) {
48      mt.State=mtDriveOn;
49    }
50    break;
51  }
52
53  case mtDriveOn: {    // Включение приводов
54    // Проверка наличия команды сброса в начальное состояние
55    if (CNC.request == mtcncReset) { mtReset(); } // Сброс в начальное состояние
56    // Проверка готовности системы
57    if (mt.ncNotReadyReq) { mtAbortRequest(); break; } // Аварийное выключение
58    axesActivate(); // Включение в слежение всех осей
59    // Запуск таймера включения приводов
60    timerStart(mt.timerState, MT_TIME_DRIVE_ON);
61    mt.State=mtWaitDriveOn;
62    break;
63  }
64
65  case mtWaitDriveOn: {    // Ожидание включения приводов
66    // Проверка наличия команды сброса в начальное состояние
67    if (CNC.request == mtcncReset) { mtReset(); } // Сброс в начальное состояние
68    // Проверка готовности системы
69    if (mt.ncNotReadyReq) { mtAbortRequest(); break; } // Аварийное выключение
70    if (timerTimeout(mt.timerState)) {
```

```
71      // Ошибка: истекло время включения приводов
72      errorSet(systemErrors.channel[0].driveOnTimeout);
73      break;
74    }
75    // Все оси находятся в слежении ?
76    if (axesActive()) {
77      mt.State=mtOthersMotorOn; // Включение вспомогательных двигателей
78    }
79    break;
80  }
```

4.4 Обработка аварийных ситуаций и ошибок электрооборудования станка

Обработка аварийных ситуаций и ошибок электрооборудования станка выполняется с помощью следующих программных средств:

- объединение `MachineErrors` – список аварийных ситуаций и ошибок;
- макрос `errorSet` – установка ошибки (соответствующему битовому полю объединения `MachineErrors` присваивается 1);
- макрос `DEFINE_ERROR` – создание и инициализация переменной типа **ErrorDescription**;
- пользовательская функция **errorsMachineScan** – набор вызовов макроса `errorScanSet`;
- пользовательская функция **errorsMachineReaction** – набор вызовов функции **errorReaction**.

В объединении `MachineErrors` (путь по умолчанию: include\platform\имя_проекта\machine_error.h) пользователем определяются аварийные ситуации и ошибки электрооборудования станка в виде битовых полей. Оно является полем структуры **Errors**.

Листинг 4.5 показывает пример объявления объединения `MachineErrors`.

Листинг 4.5. Пример объявления объединения `MachineErrors`

```
1  #ifndef MACHINE_ERRORS_H
2  #define MACHINE_ERRORS_H
3
4  #define MACHINE_ERRORS_DEFINED
5
6  union MachineErrors {
7      struct {
```

```
        unsigned emergencyStop: 1; // Аварийный останов
        unsigned lubeError: 1; // Ошибка смазки
        unsigned spinChillerError: 1; // Ошибка охлаждения шпинделя
        unsigned toolNotFound: 1; // Инструмент не найден
        unsigned overloadPumpA: 1; // Перегрузка насоса А СОЖ
        unsigned coolantLevelHigh: 1; // Высокий уровень СОЖ
        unsigned overloadChipConv: 1; // Перегрузка конвейера стружки
        unsigned linkOperatorPult: 1; // Ошибка связи с пультом оператора
    };
    unsigned errors;
};

#endif // MACHINE_ERRORS_H
```

Листинг 4.6 показывает пример использования макроса `errorSet` в функции контроля СОЖ.

Если насос А включен и произошла его перегрузка (проверка соответствующего выхода и входа в строке 7), то выключается подача СОЖ (строка 8) и выставляется соответствующий бит ошибки (строка 9).

Если уровень СОЖ высокий (проверка соответствующего входа в строке 13) и станок включен (строка 14), то выключается подача СОЖ (строка 15) и выставляется соответствующий бит ошибки (строка 17).

Системная переменная `systemErrors` имеет тип **Errors**.

Листинг 4.6. Пример использования макроса `errorSet`

```
#include "sys/sys.h"
#include "include/platform/stanok/stanok.h"
#include "include/platform/stanok/coolant.h"

void checkCoolant() {
    // Контроль двигателя насоса А
  // Насоса А включен и перегружен?
    if (mt.OUT.pumpA == 1 && mt.IN.overloadPumpA == 1) {
        coolantOff(); // выключение подачи СОЖ
        // Ошибка: перегрузка насоса А СОЖ
        errorSet(systemErrors.machine.overloadPumpA);
    }

    // Контроль уровня СОЖ
    if (mt.IN.coolantLevelHigh == 1) { // Высокий уровень СОЖ?
        if (mt.State == mtReady) { // Станок включен?
```

```
            coolantOff(); // Выключение подачи СОЖ
        }
        // Ошибка: высокий уровень СОЖ
        errorSet(systemErrors.machine.coolantLevelHigh);
    }
}

// Выключение подачи СОЖ
void coolantOff() {
    mt.OUT.pumpA = 0; // Отключение насоса A
    mt.OUT.pumpB = 0; // Отключение насоса B
    mt.OUT.pumpC = 0; // Отключение насоса C
}
```

Макрос `DEFINE_ERROR(name, code, react, cl)` создаёт переменную типа **ErrorDescription** с именем descError{name} и инициализирует её поля аргументами descError{name}.id=code, descError{name}.reaction=react, descError{name}.clear=cl.

Поле descError{name}.id – приоритет ошибки.

Поле descError{name}.reaction – идентификаторы перечисления **ErrorReaction**, определяющие реакцию на ошибку.

Поле descError{name}.clear – идентификаторы перечисления **ErrorClear**, определяющие тип сброса ошибки.

Листинг 4.7 показывает вызовы макроса `DEFINE_ERROR`, которые создают переменные descErrorMachineEmergencyStop и descErrorMachineLubeError.

Листинг 4.7. Пример вызова макроса `DEFINE_ERROR`

```
DEFINE_ERROR(MachineEmergencyStop, 0, reactNCNotReady | reactStartDisable |
  reactShowAlarm | reactStop, clearNCReset);

DEFINE_ERROR(MachineLubeError, 1, reactStartDisable | reactStopAtEnd |
  reactShowAlarm, clearSelf);
```

Функция **errorsMachineScan** должна быть реализована пользователем. В ней вызывается макрос **errorScanSet** для ошибок, определенных в объединении `MachineErrors` и макросом `DEFINE_ERROR`.

Агрументы `errorScanSet`:

- error – битовое поле объединения `MachineErrors`;
- input – значение, которое возвращает функция контроля соответствующего параметра (состояние соответствующего входа);
- desc – переменная типа **ErrorDescription**;
- request – переменная типа **ErrorClear**.

Функция **errorsMachineReaction** должна быть реализована пользователем. В ней вызывается функция **errorReaction** для ошибок, определенных в объединении `MachineErrors` и макросом `DEFINE_ERROR`.

Листинг 4.8 показывает пример реализации файла обработки аварийных ситуаций и ошибок электрооборудования станка (путь по умолчанию: include\platform\имя_проекта\machine_error.cfg).

Листинг 4.8. Пример обработки аварийных ситуаций и ошибок электрооборудования станка

```
#include "include/cnc/errors.h"
#include "include/platform/stanok/cool_spin.h"
#include "include/platform/stanok/coolant.h"
#include "include/platform/stanok/lube.h"

// Создания списка ошибок
DEFINE_ERROR(MachineEmergencyStop, 0, reactNCNotReady | reactStartDisable |
  reactShowAlarm | reactStop, clearNCReset);
DEFINE_ERROR(MachineLubeError, 1, reactStartDisable | reactStopAtEnd |
  reactShowAlarm, clearSelf);
DEFINE_ERROR(MachineSpinChillerError, 1, reactStartDisable | reactStopAtEnd |
  reactShowAlarm, clearSelf);
DEFINE_ERROR(MachineToolNotFound, 1, reactStartDisable | reactStop |
  reactShowAlarm, clearNCStart);
DEFINE_ERROR(MachineOverloadPumpA, 1, reactStartDisable | reactStop |
  reactShowAlarm, clearNCStart);
DEFINE_ERROR(MachineCoolantLevelHigh, 1, reactStartDisable | reactStop |
  reactShowAlarm, clearNCStart);
DEFINE_ERROR(MachineOverloadChipConv, 1, reactStartDisable | reactStop |
  reactShowAlarm, clearNCReset);
DEFINE_ERROR(MachineLinkOperatorPult, 1, reactNCNotReady | reactStartDisable |
  reactShowAlarm | reactStop, clearNCReset);

// Функция обновления флагов ошибок
void errorsMachineScan(int request)
{
    // аварийный останов
    errorScanSet(systemErrors.machine.emergencyStop, hasEmergencyStopRequest(),
      descErrorMachineEmergencyStop, request);
    // ошибка смазки
    errorScanSet(systemErrors.machine.lubeError, hasLubeError(),
      descErrorMachineLubeError, request);
    // ошибка охлаждения шпинделя
```

```
24      errorScanSet(systemErrors.machine.spinChillerError, hasSpinChillerError(),
          descErrorMachineSpinChillerError, request);
25      // инструмент не найден
26      errorScanSet(systemErrors.machine.toolNotFound, 0,
          descErrorMachineToolNotFound, request);
27      // перегрузка мотора насоса А (обмывочная СОЖ)
28      errorScanSet(systemErrors.machine.overloadPumpA, 0,
          descErrorMachineOverloadPumpA, request);
29      // высокий уровень СОЖ
30      errorScanSet(systemErrors.machine.coolantLevelHigh, 0,
          descErrorMachineCoolantLevelHigh, request);
31      // перегрузка конвейера стружки
32      errorScanSet(systemErrors.machine.overloadChipConv, 0,
          descErrorMachineOverloadChipConv, request);
33      // нет связи связи с пультом оператора
34      errorScanSet(systemErrors.machine.linkOperatorPult, 0,
          descErrorMachineLinkOperatorPult, request);
35  }
36
37  // Функция установки флагов действий системы согласно реакциям на ошибки
38  void errorsMachineReaction()
39  {
40    // аварийное выключение
41    errorReaction(systemErrors.machine.emergencyStop,
        descErrorMachineEmergencyStop);
42    // ошибка смазки направляющих
43    errorReaction(systemErrors.machine.lubeError, descErrorMachineLubeError);
44    // ошибка системы охлаждения шпинделя
45    errorReaction(systemErrors.machine.spinChillerError,
        descErrorMachineSpinChillerError);
46    // инструмент не найден
47    errorReaction(systemErrors.machine.toolNotFound, descErrorMachineToolNotFound);
48    // перегрузка мотора помпы А (обмывочная СОЖ)
49    errorReaction(systemErrors.machine.overloadPumpA,
        descErrorMachineOverloadPumpA);
50    // высокий уровень СОЖ
51    errorReaction(systemErrors.machine.coolantLevelHigh,
        descErrorMachineCoolantLevelHigh);
52    // перегрузка конвеера стружки
53    errorReaction(systemErrors.machine.overloadChipConv,
        descErrorMachineOverloadChipConv);
54    // потеря связи с пультом оператора
```

```
55    errorReaction(systemErrors.machine.linkOperatorPult,
        descErrorMachineLinkOperatorPult);
56  }
```

Листинг 4.9 показывает пример реализации функции контроля смазки направляющих `hasLubeError()`, которая вызывается в строке 21 листинга 4.8.

Листинг 4.9. Пример реализации функции контроля смазки направляющих

```
1  #include "sys/sys.h"
2  #include "include/platform/stanok/stanok.h"
3  #include "include/platform/stanok/lube.h"
4
5  // контроль смазки направляющих
6  int hasLubeError() {
7      if (mt.IN.lubeLevelLow == 0)  return 1;
8      else  return 0;
9  }
10 // включение смазки направляющих
11 void lubeOn() {
12     mt.OUT.autoLubeOn = 1;
13 }
14 // выключение смазки направляющих
15 void lubeOff() {
16     mt.OUT.autoLubeOn = 0;
17 }
```

ПРИЛОЖЕНИЕ 1

Примеры программ ПЛК

Управление толчковыми перемещениям

```
enum Direction { // Направление движения или останов
  dirMinus = 0,
  dirPlus,
  dirStop
};

enum JogSpeed { // Скорость толчкового движения
  jogSpeed0 = 0,
  jogSpeed1 = 1,
  jogSpeed2 = 2,
  jogSpeed3 = 4
};

PLC(1, jogPlc) // Объявление программы ПЛК

void setup() {
    enablePLC(1); // Разрешение выполнения программы ПЛК
}

void jogPlc()
{
   int dir, data;

   // Чтение входных данных:
   // первые 4 бита определяют направление движения, следующие 3 бита – скорость
   data = Servo[0].IO[3].DataIn[0];

   // Если данные нулевые, то выполняется останов
```

```
    if (data==0) dir = dirStop;
    // Если данные не нулевые, то первый бит определяет направление движения
    else dir = data & 1;

    // Сдвиг данных на 4 бита
    data = (data >> 4) & 7;

    // Запись скорости толчкового движения в переменную Motor[0].Move.JogSpeed
    switch (data) {
    case jogSpeed0:
        Motor[0].Move.JogSpeed = 0;
        break;
    case jogSpeed1:
        Motor[0].Move.JogSpeed = 10;
        break;
    case jogSpeed2:
         Motor[0].Move.JogSpeed = 25;
         break;
    case jogSpeed3:
         Motor[0].Move.JogSpeed = 50;
         break;
     default:
         Motor[0].Move.JogSpeed = 0;
    }

    if (dir == dirPlus)
    {
      // Толчковое перемещение в положительном направлении двигателем 0
        jogPlus(0);
    }
    else if (dir == dirStop)
    {
      // Останов толчкового перемещения двигателя 0
        jogStop(0);
    }
    else if (dir == dirMinus)
    {
      // Толчковое перемещение в отрицательном направлении двигателем 0
        jogMinus(0);
    }
}
```

Задание программы движения

```
#include "sys/sys.h"
#define NUMBER_OF_MOTORS 3

enum MODE { // Режимы работы
    mdNull,
    mdHome,
    mdAuto,
    mdWait,
};

enum COMMANDS { // Команды
    cmdNull,
    cmdStop,
    cmdStart,
};

union Button { // Кнопки пульта оператора
    struct{
        unsigned modeManual:1;      // Ручной режим
        unsigned modeHome:1;    // Ручной выезда в нулевую точку
        unsigned reserved0:4;
        unsigned modeAuto:1;    // Автоматический режим
        unsigned modeStep:1;    // Пошаговый режим
        unsigned reserved1:6;
        unsigned modeStart:1;   // Старт
        unsigned modeStop:1;    // Стоп
        unsigned reserved2:16;
        unsigned reserved3:32;
        unsigned reserved4:32;
        unsigned reserved5:25;
        unsigned cncOff: 1;     // Вход от ключа
        unsigned reserved6:5;
    };
    int PultBtn[4];
};

int mode;
double programmSpeed = 10;
double speedFirstStage = 700;
double speedSecondStage = 1000;
```

```
int paused = 200;
int numStep = 10;
int angleStep = 100;
Button button, doubleButton, light;

PLC(1, stend) // Объявление программы ПЛК

void setup()
{
    enablePLC(1); // Разрешение выполнения программы ПЛК
    Local.coord = 1; // Задание активной координатной системы
  // Число сегментов движения в буфере опережающего просмотра
    Coord[1].LHSize = 1024;
  // Включение функции буферизации опережающего просмотра для координатной
  // системы и задание дистанции опережающего просмотра
    Coord[1].LHDistance = 1024;
    assignMotor(0, {.X = 10000./360.}); // Привязка двигателя 0 к оси X
    assignMotor(1, {.Y = 10000./360.}); // Привязка двигателя 1 к оси Y
    assignMotor(2, {.Z = 10000./360.}); // Привязка двигателя 2 к оси Z
}

void moveMotors(int a) // Перемещение в заданное положение
{
  // Перемещение в заданное положение в установленном режиме движения
    move({.X = a, .Y = a, .Z = a});
}

int isHomeComplete(){ // Проверка выезда в нулевую точку
    int hmcmplt = 0;
    for (int i = 0; i <NUMBER_OF_MOTORS; i++){
        if (Motor[i].Trig.HomeComplete != 1){
            hmcmplt++;
        }
    }
    if (hmcmplt) return 0;
    else return 1;
}

int isClosedLoop() // Проверка режима слежения
{
    int snps = 0;
    for (int i = 0; i <NUMBER_OF_MOTORS; i++){
        if (Motor[i].Servo.ClosedLoop == 1){
```

```
84              snps++;
85          }
86      }
87      return snps;
88  }
89
90  // Программа движения
91  void motion_nc(){
92      //Счетчик угла поворота
93      static double angle = 0;
94
95      //Включить индикацию работы программы
96      light.modeStart = 1;
97      //Использовать абсолютные перемещения для моторов осей X, Y и Z
98      absAxes(axXYZ);
99      //Использовать линейные интерполированные перемещения для всех осей
100     linear();
101     //Сбросить счетчик угла поворота
102     angle = 0;
103     //Задать скорость 100 оборотов в минуту для линейно интерполированных осей
104     setF(100*360*sqrt(3));
105     //Перемещение в заданное положение всеми двигателем
106     moveMotors(angle);
107
108     // Бесконечный цикл. Программа работает в два этапа.
109     // Каждый этап имеет максимальные скорости: speedFirstStage и
        speedSecondStage
110     while (1) {
111         //Цикл разгона
112         for (int i = 1; i < numStep; i++){
113
114             //Расчитать новую скорость(чем больше i,  тем больше скорость)
115             programmSpeed = speedFirstStage*i/numStep;
116             //Задание новой скорости
117             setF(programmSpeed*360*sqrt(3));
118             //Рассчитать перемещение
119             angle = angle + angleStep;
120             //Перемещение в заданное положение всеми двигателем
121             moveMotors(angle);
122         }
123
124         //Присвоить скорость speedFirstStage
125         programmSpeed = speedFirstStage;
```

```
126         //Задание максимальной скорости для первого этапа
127         setF(programmSpeed*360*sqrt(3));
128         //Рассчитать перемещение (отработать 50 оборотов)
129         angle = angle+360*50;
130         //Перемещение в заданное положение всеми двигателем
131         moveMotors(angle);
132 
133         //Цикл торможения (аналогичен разгону, но скорость уменьшается)
134         for (int i = 1; i < numStep; i++){
135             programmSpeed = speedFirstStage/i;
136       //Задание новой скорости
137             setF(programmSpeed*360*sqrt(3));
138             angle = angle + angleStep;
139       //Перемещение в заданное положение всеми двигателем
140             moveMotors(angle);
141         }
142 
143         //Рассчитать скорость второго этапа
144         programmSpeed = speedSecondStage;
145         //Задание скорости второго этапа
146         setF(programmSpeed*360);
147         //Приостанов движения всех осей
148         dwell(3*paused);
149         //Рассчитать перемещение (полоборота в плюс)
150         angle=angle+180;
151         //Перемещение двигателем X в заданное положение
152         move({.X = angle});
153         dwell(paused);
154         //Перемещение двигателем Y в заданное положение
155         move({.Y = angle});
156         dwell(paused);
157         //Перемещение двигателем Z в заданное положение
158         move({.Z = angle});
159         dwell(paused);
160          //Рассчитать перемещение (полоборота в минус)
161         angle=angle-180;
162          //Перемещение двигателем Z в заданное положение
163         move({.Z = angle});
164         dwell(paused);
165         //Перемещение двигателем Y в заданное положение
166         move({.Y = angle});
167         dwell(paused);
```

```
168         //Перемещение двигателем X в заданное положение
169         move({.X = angle});
170         dwell(paused);
171         //Задание новой скорости
172         setF(programmSpeed*360*sqrt(3));
173         //Рассчитать перемещение (один оборот в плюс)
174         angle = angle+360;
175         //Перемещение в заданное положение всеми двигателем
176         moveMotors(angle);
177         dwell(3*paused);
178         //Рассчитать перемещение (один оборот в минус)
179         angle = angle-360;
180         //Перемещение в заданное положение всеми двигателем
181         moveMotors(angle);
182         dwell(3*paused);
183     }
184 }
185 // Программа ПЛК
186 void stend()
187 {
188     button.PultBtn[0] = Servo[0].IO[0].DataIn[0];
189     button.PultBtn[3] = Servo[0].IO[0].DataIn[3];
190 
191     if (!button.cncOff){ // Ключ в положении "Включено"
192       // Кнопка "Выезд в нулевую точку" нажата?
193         if (button.modeHome != doubleButton.modeHome){
194             // Нажата кнопка "Выезд в нулевую точку"
195             mode = mdHome;
196             light.modeHome = 1;
197             light.modeAuto = 0;
198         }
199     // Кнопка "Автоматический режим" нажата?
200         if (button.modeAuto != doubleButton.modeAuto){
201             // Нажата кнопка "Автоматический режим"
202             mode = mdAuto;
203             light.modeHome = 0;
204             light.modeAuto = 1;
205         }
206         switch (mode) {
207         case mdHome: // Режим выезда в нулевую точку
208             // Кнопка "Старт" нажата?
209             if (button.modeStart !=doubleButton.modeStart){
210                 // Нажата кнопка "Старт" – начать выезд в нулевую точку
```

```
211                 home(2); // Начать выезд в нулевую точку двигателем Z
212                 home(1); // Начать выезд в нулевую точку двигателем Y
213                 home(0); // Начать выезд в нулевую точку двигателем X
214                 light.modeStart = 1;
215             }
216             if (isHomeComplete()){ // Выезд в нулевую точку завершён?
217                 light.modeStart = 0;
218             }
219         // Кнопка "Стоп" нажата?
220             if (button.modeStop != doubleButton.modeStop){
221           // Нажата кнопка "Стоп"
222                 for (int i = 0; i < NUMBER_OF_MOTORS; i++){
223                     jogStop(i); // Останов толчкового перемещения двигателя
224                 }
225             }
226             break;
227         case mdAuto: // Автоматический режим
228             // Кнопка "Старт" нажата?
229             if (button.modeStart != doubleButton.modeStart)
230             {
231         // Нажата кнопка "Старт"
232                 begin (1,0); // Задание программы движения "motion_nc" для КС 1
233                 run(1); // Запуск программы движения в КС 1
234             }
235             // Кнопка "Стоп" нажата?
236             if (button.modeStop !=doubleButton.modeStop)
237             {
238         // Нажата кнопка "Стоп"
239                 abort(1); // Прерывание программы движения для КС 1
240                 light.modeStart = 0;
241             }
242             break;
243         case mdNull:
244             // Хотя бы один двигатель не находится в слежении?
245             if (!isClosedLoop()){
246                 for (int i = 0; i < NUMBER_OF_MOTORS; i++){
247                     jogStop(i); // Включение двигателей в слежение
248                 }
249             }
250             mode = mdWait;
251         default:
252         }
253         Servo[0].IO[0].DataOut[0]=light.PultBtn[0];
```

```
254         Servo[0].IO[0].DataOut[2] = 8;
255     }else{ // Ключ в положении "Выключено"
256         mode = cmdNull;
257         Servo[0].IO[0].DataOut[2] = 0;
258         Servo[0].IO[0].DataOut[0] = 0;
259         light.PultBtn[0] = 0;
260         if (isClosedLoop()){ // Хотя бы один двигатель находится в слежении?
261             for (int i = 0; i < NUMBER_OF_MOTORS ; i++){
262                 kill(i); // Отключить двигатели
263             }
264         }
265     }
266   // Фиксация изменения нажатия кнопок
267     doubleButton.PultBtn[0] = button.PultBtn[0];
268   // Фиксация изменения нажатия кнопок
269     doubleButton.PultBtn[3] = button.PultBtn[3];
270 }
```

Программа включения/выключения станка

```
// Задание интервалов таймеров
#define MT_TIME_DRIVE_ON            10000
#define MT_TIME_DRIVE_PHASE_REF     22000
#define MT_TIME_DRIVE_OFF           10000
#define MT_TIME_DRIVE_STOP          5000
#define MT_TIME_DRIVE_ABORT         3500
#define MT_TIME_RESET               500
#define MT_TIME_FILTRED_CORR        500
#define TIMER_ERRORS                (1*2500)
#define TIMER_HOME_INCOMPLETE       (1*2500)
#define TIMER_START_PAUSED          (1*2500)
#define TIMER_START_HELD            (1*2500)

MTDesc mt; // Объявление структуры с данными станка

PLC (1, stanokOnOff) // Объявление программы ПЛК

void stanokOnOff() {

    readInputs(); // Чтение входных регистров

    // Управление сигналом готовности системы
    if (mt.ncNotReadyReq) {
        mtSignalReady(0); // Система не готова
    } else {
        mtSignalReady(1); // Система готова
    }

    mtControlRequest(); // Обработка команд пульта оператора

    if (CNC.request == mtcncNone)
        cncRequest(commandPop(CNC.commands));

  // Обработка запроса выключения УЧПУ и станка
    controlPowerCNC(CNC.request);

    // Автомат управления станком
    switch (mt.State) {
    case mtNotReady: {  // Ожидание входа от главного пускателя
        // Проверка наличия команды сброса в начальное состояние
```

```
41        if (CNC.request == mtcncReset) {mtReset();} // Сброс в начальное состояние
42        // Станок включен и система готова?
43        if (mtIsOn() && !mt.ncNotReadyReq) mt.State=mtStartOn;
44        break;
45      }
46
47      case mtStartOn: {  // Начало включения
48          // Проверка наличия команды сброса в начальное состояние
49        if (CNC.request == mtcncReset) {mtReset();} // Сброс в начальное состояние
50        // Проверка готовности системы
51        if (mt.ncNotReadyReq) {mtAbortRequest(); break;} // Аварийное выключение
52        mt.State = mtDriveOn;
53        // Фазировка выполнена?
54        if (!axesPhaseRefComplete() || !spinsPhaseRefComplete()) mt.State =
            mtPhaseRef;
55        break;
56      }
57
58      case mtPhaseRef: {  // Фазировка
59        if (CNC.request == mtcncReset) {mtReset();} // Сброс в начальное состояние
60        // Проверка готовности системы
61        if (mt.ncNotReadyReq) {mtAbortRequest(); break;} // Аварийное выключение
62        if (axesPhaseRef() || spinsPhaseRef()) { // Требуется фазировка?
63          mt.State = mtWaitPhaseRef;
64          // Запуск таймера фазировки
65          timerStart(mt.timerState, MT_TIME_DRIVE_PHASE_REF);
66        } else {
67          mt.State = mtDriveOn;
68        }
69        break;
70      }
71
72      case mtWaitPhaseRef: {  // Ожидание окончания фазировки
73        if (CNC.request == mtcncReset) {mtReset();} // Сброс в начальное состояние
74        if (mt.ncNotReadyReq) {mtAbortRequest(); break;} // Аварийное выключение
75        if (timerTimeout(mt.timerState)) {
76          // Ошибка: истекло время фазировки
77          errorSet(systemErrors.channel[0].phaseRefTimeout);
78          break;
79        }
80        // Фазировка выполнена?
81        if (axesPhaseRefComplete() && spinsPhaseRefComplete()) {
82          mt.State=mtDriveOn;
```

```
83        }
84        break;
85      }
86
87      case mtDriveOn: {  // Включение приводов
88        if (CNC.request == mtcncReset) {mtReset();} // Сброс в начальное состояние
89        if (mt.ncNotReadyReq) {mtAbortRequest(); break;} // Аварийное выключение
90        axesActivate(); // Включение в слежение всех осей
91        // Запуск таймера включения приводов
92        timerStart(mt.timerState, MT_TIME_DRIVE_ON);
93        mt.State=mtWaitDriveOn;
94        break;
95      }
96
97      case mtWaitDriveOn: {  // Ожидание включения приводов
98        if (CNC.request == mtcncReset) {mtReset();} // Сброс в начальное состояние
99        if (mt.ncNotReadyReq) {mtAbortRequest(); break;} // Аварийное выключение
100       if (timerTimeout(mt.timerState)) {
101         // Ошибка: истекло время включения приводов
102         errorSet(systemErrors.channel[0].driveOnTimeout);
103         break;
104       }
105       // Все оси находятся в слежении ?
106       if (axesActive()) {
107         mt.State=mtOthersMotorOn; // Включение вспомогательных двигателей
108       }
109       break;
110     }
111
112     case mtOthersMotorOn: {  // Включение вспомогательных моторов
113       if (CNC.request == mtcncReset) {mtReset();} // Сброс в начальное состояние
114       if (mt.ncNotReadyReq) {mtAbortRequest(); break;} // Аварийное выключение
115       mt.State=mtReady;
116       // Команда включения ручного режима
117       commandPush(CNC.commands, mtcncActivateManual, 6);
118       break;
119     }
120
121     case mtReady: {  // Станок включен
122       if (mt.ncNotReadyReq) {mtAbortRequest(); break;} // Аварийное выключение
123       // Штатное выключение станка
124       if (CNC.request == mtcncPowerOff) {
125         mt.State=mtStartOff;
```

```
126         break;
127       }
128       break;
129     }
130
131     case mtAbort: { // Аварийное торможение
132       if (CNC.request == mtcncReset) {mtReset();} // Сброс в начальное состояние
133       if (timerTimeout(mt.timerState)) {
134         // Ошибка: истекло время аварийного торможения
135         errorSet(systemErrors.channel[0].abortTimeout);
136         axesDeactivate(); // Выключение осей
137         spinsDeactivate(); // Выключение шпинделей
138         break;
139       }
140       // Все оси и шпиндели аварийно остановлены?
141       if (axesAborted() && spinsAborted()) {
142         mt.State=mtNotReady;
143         CNC.mode = cncOff;
144       }
145       break;
146     }
147
148     // Выключение станка
149     case mtStartOff: {  // Начало выключения
150       if (CNC.request == mtcncReset) {mtReset();} // Сброс в начальное состояние
151       mt.State=mtOthersMotorOff;
152       break;
153     }
154
155     case mtOthersMotorOff: {  // Выключение вспомогательных моторов
156       if (CNC.request == mtcncReset) {mtReset();} // Сброс в начальное состояние
157       mt.State=mtAxisStop;
158       break;
159     }
160
161     case mtAxisStop: {  // Останов осей
162       if (CNC.request == mtcncReset) {mtReset();} // Сброс в начальное состояние
163       if (mt.ncNotReadyReq) {mtAbortRequest(); break;} // Аварийное выключение
164       mt.State=mtAxisWaitStop;
165       // Запуск таймера останова
166       timerStart(mt.timerState, MT_TIME_DRIVE_STOP);
167       break;
168     }
```

```
169
170     case mtAxisWaitStop: {  // Ожидание останова осей
171       if (CNC.request == mtcncReset) {mtReset();} // Сброс в начальное состояние
172       if (mt.ncNotReadyReq) {mtAbortRequest(); break;} // Аварийное выключение
173       if (timerTimeout(mt.timerState)) {
174         // Ошибка: истекло время останова осей
175         errorSet(systemErrors.channel[0].stopTimeout);
176         break;
177       }
178       // Все оси и шпиндели остановлены?
179       if (axesStopped() && spinsAborted()) {
180         mt.State=mtDriveOff;
181         axesDeactivate(); // Выключение осей
182         spinsDeactivate(); // Выключение шпинделей
183         // Запуск таймера выключения приводов
184         timerStart(mt.timerState, MT_TIME_DRIVE_OFF);
185       }
186       break;
187     }
188
189     case mtDriveOff: {  // выключение осей
190       if (CNC.request == mtcncReset) {mtReset();} // Сброс в начальное состояние
191       if (timerTimeout(mt.timerState)) {
192         // Ошибка: истекло время выключения приводов
193         errorSet(systemErrors.channel[0].driveOffTimeout);
194         axesForceKill(); // Принудительное выключение осей
195         spinsForceKill(); // Принудительное выключение шпинделей
196         mt.State = mtNotReady;
197         break;
198       }
199       if (axesInactive() && spinsInactive()) {
200         mt.State=mtWaitOff;
201       }
202       break;
203     }
204
205     case  mtWaitOff: { // Ожидание выключения станка
206       if (CNC.request == mtcncReset) {mtReset();} // Сброс в начальное состояние
207       if (mtIsOff()) { // Станок выключен?
208         mt.State = mtNotReady;
209         CNC.mode = cncOff;
210       }
211       break;
```

```
212     }
213     }
214
215     mtUpdateCNCIndication(); // Обновление индикации пульта
216     writeOutputs(); // Запись в выходные регистры
217 }
218
219 int hasEmergencyStopRequest() // Нажата кнопка аварийного останова?
220 {
221     if (mt.IN.emgButtonUnlocked == 0)  return 1;
222     else return 0;
223 }
224
225 int mtIsOn() // Станок включён?
226 {
227     return mt.IN.mtOn;
228 }
229
230 int mtIsOff() // Станок выключен?
231 {
232     return !mt.IN.mtOn;
233 }
234
235 void mtReset() // Сброс в начальное состояние
236 {
237     errorScanRequest(clearNCReset); // Обновление флагов ошибок
238     CNC.request = mtcncNone;
239     commandFlush(CNC.commands);
240 }
241
242 void mtSignalReady(int ready) // Управление сигналом готовности системы
243 {
244     if (ready) {
245         if (mt.State != mtWaitOff) {
246             mt.OUT.controlReady = 1; // Система готова
247         } else {
248             mt.OUT.controlReady = 0; // Система не готова
249         }
250     } else {
251         mt.OUT.controlReady = 0; // Система не готова
252     }
253 }
254
```

```
255 void mtAbortRequest() // Обработка запроса аварийного торможения
256 {
257     mt.State = mtAbort;
258     axesAbortAll(); // Аварийное выключение осей
259     spinsAbortAll(); // Аварийное выключение шпинделей
260     timerStart(mt.timerState, MT_TIME_DRIVE_ABORT);  // запускаем таймер
261 }
```

Предметный указатель

О

П

Р

С