ПРОМСИТЕХ

PositionServo

```
;***************************** I/O List  ***********************************
;       Input A1    -   not used
;       Input B1    -   not used
;       Input C1    -   not used
;       Output 1    -   Pick Arm
;       Output 2    -   Gripper
;********************** Initialize and Set
UNITS = 1
ACCEL = 75
DECEL =75
MAXV = 10
;********************** Events ************
;Set Events handling here
;********************** Main Program  *****
PROGRAM_START:

ENABLE

EVENT  SPRAY_GUNS_ON ON    ;Enable the Eve

EVENT  SPRAY_GUNS_OFF ON   ;Enable the Eve

WAIT UNTIL IN_A4==0        ;Make sure Arm i

MOVEP 0                    ;Move to positi

OUT1 = Output_On           ;Turn on output

WAIT UNTIL IN_A4 == 1      ;Check input to

OUT2 = Output_On           ;Turn on output

WAIT TIME 1000             ;Delay 1 sec to

OUT1 = Output_Off          ;Turn off outpu

WAIT UNTIL IN_A4==0        ;Check input to

MOVED 100                  ;Move to Place

OUT1 = Output_On           ;Turn on output

WAIT UNTIL IN_A4 == 1      ;Check input to

OUT2 = Output_Off          ;Turn off outpu

WAIT TIME 1000             ;Delay 1 sec to

OUT1 = Output_Off          ;Retract Pick a

WAIT UNTIL IN_A4 == 0      ;Check input to

GOTO PROGRAM_START

END
```

S950

Содержание

Инструкция по технике безопасности

Вся информация по технике безопасности, которая предоставлена в данной инструкции по эксплуатации, изложена следующим образом:

Сигнальное сообщение! (Характеризует степень серьезности опасности)
Примечание (описывает опасность и информирует, как действовать дальше)

Пиктограмма		Сигнальные сообщения	
	Предупреждение об опасном электрическом напряжении	**ОПАСНО!**	Предупреждение о приближающейся опасности. Последствия при несоблюдении: Смерть или тяжелые повреждения.
	Предупреждение об общей опасности	**ВНИМАНИЕ!**	Предупреждение о возможных, очень опасных ситуациях. Последствия при несоблюдении: Смерть или тяжелые повреждения.
	Предупреждение о повреждении оборудования	**СТОП!**	Предупреждение о возможном повреждении оборудования и материала. Последствия при несоблюдении: Повреждения контроллера/привода или оборудования к нему.
	Информация		Общее примечание. При его соблюдении управление системой контроллера/привода значительно облегчается.

1. Начало работы

1.1 Введение

Характеристики

Модель 940 сервопозиционер: Модель 940 является программируемым цифровым приводным контроллером, который можно сконфигурировать как обособленный программируемый приводной контроллер или как привод с улучшенными рабочими характеристиками скорости и крутящего момента для централизованных систем управления.

MotionView: MotionView представляет собой универсальное связное конфигурируемое программное обеспечение (софт) используемое приводами 94 и 940. Софт оборудован автоматическим механизмом самоконфигурации, распознающим привод, к которому он подсоединен и компонующим соответствующий набор инструментов. Программная платформа «MotionView» разделена на три секции или окна, "Окно дерева параметров", "Окно обзора параметров" и "Окно сообщений". Дополнительную информацию см. в пункте 1.3

Язык программирования «SimpleMotion» (SML): SML программный софт, используемый «MotionView». Софт SML предоставляет очень гибкую среду разработки для создания решений прикладных задач привода. В возможности софта входит создание шагов сложных точно заданных перемещений, режим ввода/вывода данных (I/O), выполнение сложных логических решений, выполнение ветвления программы, использование синхронизированной обработки события, а также ряд других функций, обеспечиваемых PLC и высокопроизводительными контроллерами механизма работы.

Пользовательская программа (или программа индексатор): Это программа SML, созданная пользователем для описания программного режима работы привода 940. Пользовательскую программу можно хранить в вашем PC или в памяти 940. Прежде чем 940 сможет выполнить пользовательскую программу при помощи инструментов «MotionView Studio», ее необходимо компилировать (преобразовать) в бинарную форму.

«MotionView Studio»: «MotionView Studio» является частью программной платформы «MotionView». Это программный комплект, содержащий все инструменты ПО необходимые для программирования и отладки системы отслеживания координат. В эти инструменты входят полноэкранный текстовый редактор, компилятор программ, утилита состояния и монитора, онлайновый осциллограф и функция отладки, позволяющая пользователю осуществлять шаги в программе во время ее разработки.

ВНИМАНИЕ!

— Опасность внезапного запуска двигателя! Во время использования ПО «MotionView» или же работая с сервопозиционером 940 при помощи RS-232/485 или Ethernet, двигатель может быть запущен внезапно, последствием чего может стать повреждение оборудования и/или травмирование персонала. Убедитесь, что созданы условия для работы прибора в таком режиме и что установлены все защитные ограждения и кожухи для защиты персонала.

— Опасность поражения электрическим током! Напряжение сети составляет 115 VAC или 230 VAC относительно земли. Избегайте непосредственного контакта с печатной платой и элементами цепи во избежание серьезных повреждений или смертельного исхода. Отключите питание и подождите 60 секунд, прежде чем обслуживать привод. Конденсаторы держат заряд после отключения питания.

1.2 Начало работы с Сервопозиционером модели 940 (PositionServo)

Перед тем как сервопозиционер 940 сможет выполнять программу движения ее необходимо правильно установить и сконфигурировать. Начинающим пользователям рекомендуется прочесть соответствующие пункты в данной инструкции для лучшей конфигурации программируемых параметров и функций 940. Им

также рекомендуется обратиться к Руководству пользователя сервопозиционера для правильной установки аппаратного обеспечения. Привод сервопозиционера 940 обладает рядом функций и параметров программируемых посредством программного обеспечения «MotionView». Ниже приведен перечень программируемых функций и параметров, характерных для программного управления. Они перечислены в том же порядке, в каком они появляются в «Окне дерева параметров» ПО «MotionView». Пожалуйста, обращайтесь к Руководству пользователя сервопривода модели 940 за более детальной информацией.

Параметры

• **Автозагрузка – Активирована/Деактивирована** (Autoboot - Enable / Disable)

Если данная опция активирована, при включении привод начнет выполнение пользовательской программы сохраненной во флэш-памяти привода. Если во флэш-памяти нет верной программы, тогда ее необходимо запустить вручную через «MotionView» или главный интерфейс (Host Interface).

ОПАСНО!

Опасность внезапного запуска двигателя! Во время использования софта «MotionView» или же работая с приводом позиционного сервомеханизма 940 при помощи RS-232/485 или Ethernet, двигатель может быть запущен внезапно, последствием чего может стать повреждение оборудования и/или травмирование персонала. Убедитесь, что созданы условия для работы прибора в таком режиме и что установлены все защитные ограждения и кожухи для защиты персонала.

• Group ID

Функция Group ID позволяет группировать приводы 940 вместе через Ethernet. При использовании с командами SEND и SENDTO приводы одной группы могут использовать совместно и обновлять переменные. Идентификационным номерам группы ID (Group ID Numbers) может быть присвоено значение от 0 до 32767. Подробную информацию см. в пунктах относительно SEND и SENDTO.

Коммуникации

• Установка IP (IP Setup) – воспроизводит свойства и установочные параметры для коммуникационного порта Ethernet (IP адрес)(IP Address).

Цифровые входы/выходы(Digital I/O)

• Входы

- В 940 есть 12 цифровых входов. Входы организованы в три группы по четыре, [A1 - A4], [B1 - B4], и [C1 - C4]. Каждая группа совместно использует свой общий вход [Acom, Bcom, и Ccom].
- Посредством ПО «MotionView» входам можно индивидуально задать время дребезга. Время дребезга (debounce time) может быть задано от 0 до 1000ms. (1ms = 0.001 сек.)
- Входы можно контролировать посредством пользовательской программы, главного интерфейса и/или им можно присвоить функции особого назначения (Special Purpose Functions). Подробную информацию см. п. 1.6

• Выходы

- В 940 есть 5 цифровых выходов. Назначение одного выхода предопределено, он включается при активации привода и в режиме вращения (RUN mode). Выходы 1 - 4 можно также активировать через пользовательскую программу, главный интерфейс или им можно присвоить функцию особого назначения (Special Purpose Function). Подробную информацию см. в п. 1.6

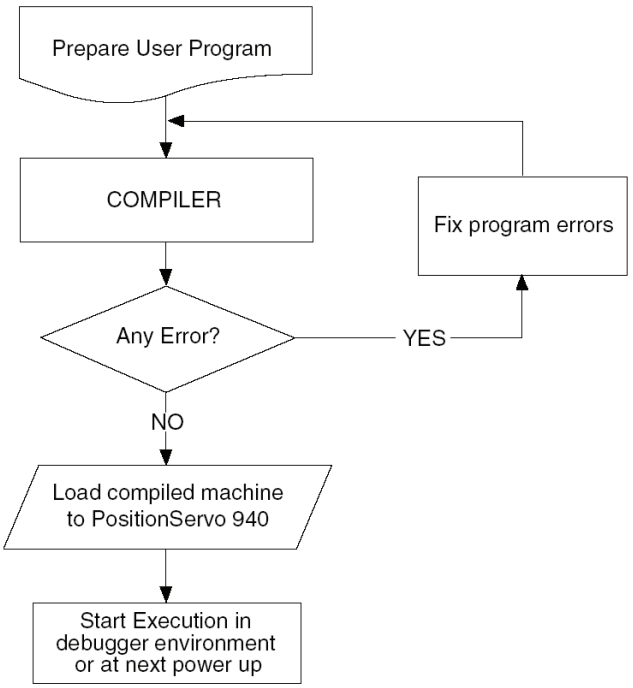
Программа индексатор

При выборе файла программы индексатора из узлового дерева, «Окно обзора параметров» отображает пользовательскую программу приводов. Эту область можно использовать для ввода, редактирования и отладки пользовательской программы. В меню и на инструментальной линейке отображаются дополнительные программные параметры. Подробную информацию см. в п. 1.3.

1.3 Обзор блок-схемы программирования

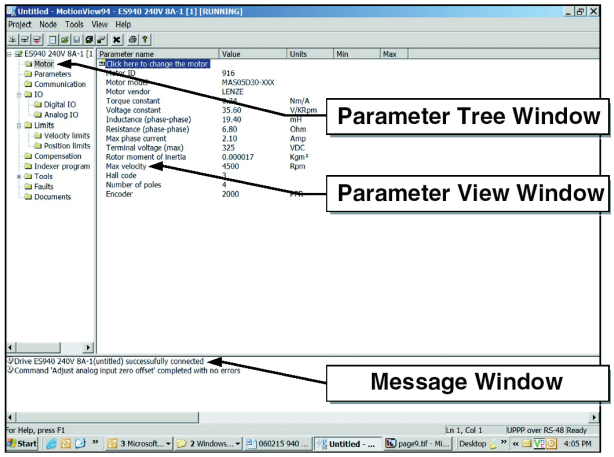
«MotionView» использует язык программирования «SimpleMotion» (SML), программная структура которого похожа на Бейсик. Посредством SML можно легко и быстро создать мощные прикладные программы движения. При помощи SML описывается логика вашей системы, движение, управление вводами/выводами (I/O) и взаимодействие пользователя с системой программными средствами, т.е. созданием программы. В структуру программы входит полный комплект арифметических операторов и операторов логического программирования, позволяющих пользователю задавать команды движения,

управлять вводами/выводами (I/O) и контролировать ход программы. Прежде чем привод 940 сможет выполнять пользовательскую программу, ее необходимо скомпилировать (преобразовать) в бинарный код и загрузить в привод. Чтобы скомпилировать программу, нажмите на линейке инструментов кнопку «Компилировать» (“Compile”). Программу можно компилировать и загрузить одновременно, выбрав на линейке инструментов кнопку «Компилировать и загрузить» (“Compile and Load”). После загрузки скомпилированная программа хранится и в памяти 940 и во внутренней флэш-памяти. Приведенный ниже рисунок подводит итог процессу составления программы.



Prepare User Program – подготовьте пользовательскую программу, COMPILER – компилятор,
Any Error? – Есть ли ошибки?, YES - да, Fix program errors – исправьте программные ошибки, NO – нет, Load compiled machine to PositionServo 940 – загрузите скомпилированное устройство в сервопозиционер 940
Start Execution in debugger environment or at next power up – начните выполнение в режиме отладчика или после включения питания

1.4 Программное обеспечение MotionView / MotionView Studio



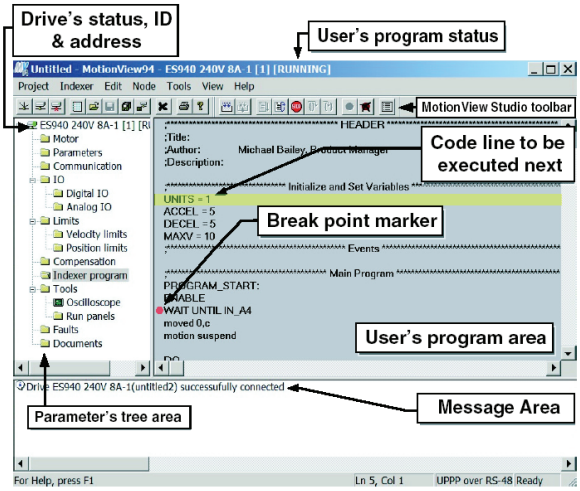
– окно дерева параметров

– окно обзора параметров

– окно сообщений

«MotionView» является универсальной программой, которая используется для коммуникации и конфигурации приводов 94 и 940. Платформа «The MotionView» разделена на три окна. Первое – окно дерева параметров (“Parameter Tree Window”). Это окно используется как Windows Explorer. Различные параметры привода представлены в этом окне папками и файлами. При выборе файла с нужным параметром вся соответствующая информация для этого параметра появляется во втором окне – окне обзора параметров. Дальше пользователь может активировать, деактивировать или редактировать характеристики или параметры. Третье окно - окно сообщений. Это окно расположено в нижней части экрана и отображает все состояние всей связи, все ошибки и каждую в отдельности.

MotionView Studio

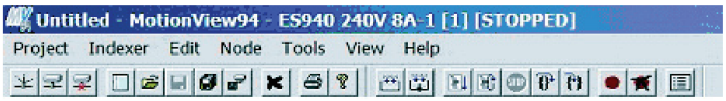


Drive's status, ID& address – состояние привода, ID и адрес
User's program status – состояние пользовательской программы
MotionView Studio toolbar – панель инструментов «MotionView Studio»
Code line to be executed next – строка, которую программа будет выполнять следующей
Break point marker – маркер точки прерывания программы
User's program area – область пользовательской программы
Message Area – область сообщений
Parameter's tree area – область дерева параметров

Разметка экрана Motionview Studio

«MotionView Studio» является частью универсальной программы «MotionView» и включает в себя комплект инструментов, используемых для ввода, компилирования, загрузки и отладки пользовательской программы. Для просмотра и разработки пользовательской программы выберите файл программы индексатора (“Indexer Program”) из окна дерева параметров. При выборе этого файла в окне обзора параметров будет отображена пользовательская программа и откроется доступ к линейке инструментов Studio, что позволит пользователю редактировать, компилировать и отлаживать пользовательскую программу. Отображаемая программа является текущей сохраненной в приводе программой. Текст этой программы всегда загружается из памяти 940 при запуске «Motion View». Это действие всегда выполняется независимо от состояния выполнения программы. Сервопозиционер обрабатывает сигналы обратной связи от инкрементного энкодера или резольвера. В режимах позиционирования или скорости может также использоваться второй энкодер.

Менюнабора инструментов Studio и опции панели инструментов



Менюнабора инструментов Studio
При разработке и редактировании программы открывается доступ к пунктам меню «Индексатор» (Indexer) и «Редактировать» (Edit). Эти ярлыки доступны только в области программирования (окно обзора параметров). Данные опции используют для загрузки, компиляции, сохранения и отладки программы. На приведенных примерах рассматривается, как пользоваться этими опциональными ярлыками.

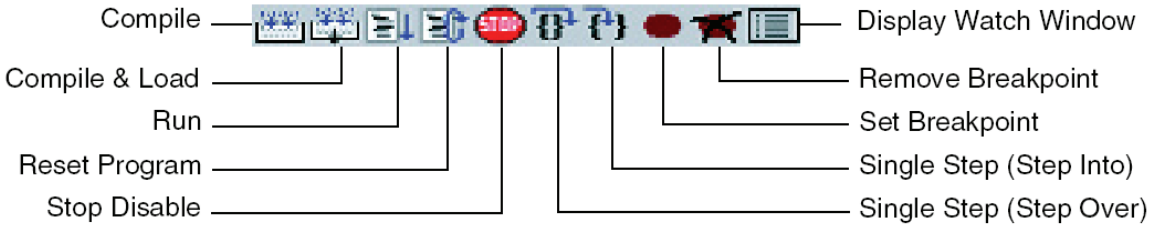
Обратите внимание, что для использования этих возможностей, необходимо выбрать программу индексатор из узлового дерева. Это расширит опции меню. Далее для активации ярлыков меню необходимо нажать мышью в любой точке окна обзора параметров.

● Загрузка **Load** пользовательской программы из PC в «MotionView»
- Выберите из раскрывающегося меню «Индексатор» (“**Indexer**”).
- Выберите из раскрывающегося меню «Импортировать программу из файла» (“**Import program from file**”) и выберите нужную программу.
Эта операция загружает программу из файла в окно редактора, но не загружает ее в память привода 940.
● Компиляция “**Compile**“ программы и загрузка ее в привод 940.
- Выберите из раскрывающегося меню «Индексатор» (“**Indexer**”).
- Выберите из раскрывающегося меню «Компилировать и переслать в привод» (“**Compile and send to drive**”).
Для проверки синтаксических ошибок без загрузки программы в привод выберите «Компилировать» (“**Compile**”) из меню «Индексатор» (“**Indexer**”). Если компилятор найдет синтаксическую ошибку, процесс компиляции будет остановлен и программа не загрузится в память привода. Отчет об ошибке появляется в нижней части экрана в окне сообщений.
● Сохранение”**Save**” пользовательской программы из «MotionView» в компьютер (PC).
- Выберите из раскрывающегося меню «Индексатор» (“**Indexer**”).
- Выберите из раскрывающегося меню «Экспортировать программу в файл» (“**Export program to file**”).
Программа сохранит данную программу в файл «данные пользователя» (“**User Data**”) в «MotionView».
● Запуск пользовательской программы в приводе.
- Выберите из раскрывающегося меню «Индексатор» (“**Indexer**”).
- Выберите из раскрывающегося меню «Запустить» (“**Run**”).
Если программа уже запущена, тогда вначале ее нужно перезапустить (**Restart**) или остановить (**Stop**).
● Использование **шага программы** при работе с пользовательской программой.
- Выберите из раскрывающегося меню «Индексатор» (“**Indexer**”).
- Выберите из раскрывающегося меню «Скачок/ Перешагнуть» (“**Step in / Step over**”).
940 будет выполнять программу пошагово. Состояние текущей программы будет выделено на экране. Если программа запущена, ее нужно будет остановить или перезапустить.
● Установка точек прерывания программы “**Breakpoint(s) in the program**”.
- Выберите в программе точку, на которой вы бы хотели остановить программу.
- Выберите из раскрывающегося меню «Индексатор» (“**Indexer**”).
- Выберите из раскрывающегося меню «Тумблер точки прерывания» (“**Toggle breakpoint**”).
Для отладки пользовательской программы удобней поставить точки прерывания на критических узлах программы. Эти точки прерывания обозначаются красными точками и останавливают выполнение программы приводом, но не блокируют привод и позиционные переменные. После остановки программы, пользователь может продолжить выполнение программы, выполнять программу пошагово или перезапустить ее.
● Останов **Stop** выполнения программы.
- Выберите из раскрывающегося меню «Индексатор» (“**Indexer**”).
- Выберите из раскрывающегося меню «Остановить» (“**Stop**”).
Программа будет остановлена, после выполнения текущей команды. Можно возобновить выполнение программы, выбрав «Запустить» (**Run**). Пожалуйста, обратите внимание, что кнопка СТОП (STOP) не останавливает и не блокирует движение, а только останавливает выполнение управляющей программы.

● **Перезапуск** выполнения программы
- Выберите из раскрывающегося меню «Индексатор» (“**Indexer**”).

- Выберите из раскрывающегося меню «Перезапустить» (“**Restart**”).
Программа перезагрузится, и привод будет заблокирован. Все позиционные переменные станут недействительными.

Комплект инструментов «Studio» Опции панели инструментов
При разработке пользовательской программы становится активной панель инструментов «MotionView Studio». На панели инструментов находятся ярлыки для большинства опций вкладки меню опций индексатора. Панель инструментов активна только когда вы в области программирования (окно обзора параметров). Данные опции используются ля загрузки компиляции, сохранения и отладки программы.



Compile	Компилировать	Display Watch Window	Смотровое окно на экране дисплея
Compile & Load	Компилировать и загрузить	Remove Breakpoint	Удалить точку прерывания
Run	Запустить	Set Breakpoint	Установить точку прерывания
Reset Program	Перезагрузить программу	Single Step (Step Into)	Один шаг (Скачок)
Stop Disable	Остановить Заблокировать	Single Step (Step Over)	Один шаг (Перешагнуть)

Пиктограммыпанели инструментов «Motion View Studio»

1.5 Основыпрограммирования
Пользовательская программа состоит из команд, которые при их выполнении не только запускают механизм движения, но также обрабатывают I/O приводов и принимают решения на основе параметров привода. До запуска движения необходимо сконфигурировать определенные параметры привода и ввода/вывода (I/O). Для их конфигурации выполните следующие операции.

Установка параметров (Parameter setup) - Выберите **Параметр (“Parameter”)** в окне дерева параметров и задайте следующие параметры.

Установка привода (Drive mode) в режим «Позиционирование» (Position).
- Выберите в окне обзора параметров **«Режим привода» (“Drive mode”)**.
- Выберите из раскрывающегося меню «**Позиционирование**» **(“Position”)**.

Установка режима адресации (Reference mode) на внутренний (Internal).
- Выберите в меню обзора параметров **«Адресация» (“Reference”)**.
- Выберите в раскрывающемся меню **«Внутренний» (“Internal”)**.

Установка фии переключения разрешения (Enable Switch Function) на “**запрет**” **(Inhibit).**
- Выберите в окне обзора параметров **«Фия переключения разрешения»** (**“Enable Switch Function”)**.
- Выберите в меню «**Запрет**» **“Inhibit”**.

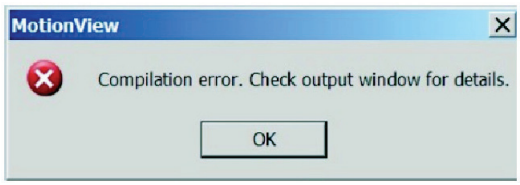
Конфигурация I/O
Вход A3 является входом специального назначения Запретить/Разрешить (Inhibit/Enable). (см. п.1.6) Перед выполнением программы вход A3 должен быть активирован, чтобы разблокировать привод и вывести его из режима запрета. Примечание: Если привод начинает выполнять пользовательскую программу и подходит к команде «Запретить» (“Enable”), а вход 3A не активирован, это приведет к ошибке “F_36”,(«Блокировка привода» “Drive Disable”).

Программа базового движения
Выберите в дереве параметров **«Программа индексатор» (“Indexer program”)**. В окне обзора параметров отобразится текущая пользовательская программа, сохраненная в приводе 940. Обратите внимание, если в памяти 940 нет действующей программы, программная область будет пустой.

Удалите существующую программу и замените ее следующей:

```
UNITS=1
ACCEL = 5
DECEL = 5
MAXV = 10
ENABLE
MOVED 10
MOVEDISTANCE  -10
END
```

После введения текста в область программы, выберите на панели инструментов пиктограмму «Компилировать и загрузить» (“**Compile and load**”). После завершения компиляции должно появится следующее сообщение:



Ошибка компиляции. Подробнее см. окно выхода.

Нажмите “OK”, чтобы закрыть диалоговое окно «Ошибка компиляции» (“Compilation error”). В окне сообщений, расположенном внизу экрана, появится причина ошибки компиляции. «MotionView» также выделит строку программы, в которой возникла ошибка.

```
UNITS=1
ACCEL = 5
DECEL = 5
MAXV = 10 ;
ENABLE
MOVED 10 ;
MOVEDISTANCE  -10
END
```

В приведенном примере причина ошибки в том, что “**MOVEDISTANCE**” - не действительная команда. Измените текст “**MOVEDISTANCE**” на “**MOVED**”.

```
UNITS=1
ACCEL = 5
DECEL = 5
ENABLE
MOVED 10
MOVED -10
END
```

После редактирования программы, выберите на панели инструментов пиктограмму «Компилировать и загрузить» (“Compile and load”). После завершения компиляции должно появится следующее окно сообщений.

Компиляция завершена без ошибок.

Теперь программа скомпилирована, загружена в память привода и готова к запуску. Нажмите “OK”, чтобы закрыть диалоговое окно.

Для запуска программы выберите на панели инструментов пиктограмму «Пуск» (“**Go**”). Привод начнет выполнять пользовательскую программу. Двигатель совершит 10 оборотов против

часовой стрелки и затем 10 оборотов по часовой стрелке. После выполнения всего кода программа остановится, и привод останется включенным.

Для повторного запуска программы выберите на панели инструментов пиктограмму «Перезапустить» (“**Restart**”). Это блокирует привод и перезагрузит программу на выполнение с начала. Обратите внимание, сама программа не запускается автоматически. Для ее запуска выберите пиктограмму «Запустить» (“Run”) на панели инструментов или из раскрывающегося меню «Индексатор» (“**Indexer**”).

Компоновка программы

При разработке программы важно ее структурировать. Рекомендуется разделить программу на следующие 7 секций:

Заголовок: Заголовок определяет название программы, ее автора и описание возможностей. Он может также включать дату создания и номер редакции.

Перечень I/O: Перечень I/O определяет назначение входов и выходов привода. Например вход A1 можно использовать как Пусковой выключатель.

Инициализация & Множественные переменные: Инициализация и множественные переменные определяют настройки привода и системные переменные. Например, здесь задаются величины ускорения, замедления и максимальной скорости.

События: Событие – это маленькая программа, работающая независимо от главной программы. Эта секция используется для определения события.

Главная программа: Главная программа представляет собой область, в которой определяется функционирование привода.

Подпрограммы: В этой области хранятся все и каждая из подпрограмм. Эти подпрограммы вызываются в главную программу командой «**GO SUB**».

Устранитель неисправностей: В этой области хранится код устранения неисправностей. Во время использования устранителя неисправностей этот код будет применен, как только привод совершит ошибку.

Ниже приведен пример программы «Pick and Place», разбитой на указанные выше сегменты.

```
 ************************************ЗАГОЛОВОК*****************************************
;Заглавие:       Pick and Place программа пример
;Автор:          Lenze / AC Technology
;Описание:       Это программа образец, показывающая простой алгоритм, который берет деталь,
                 передвигает в заданное место и опускает ее.
;*********************************** Перечень I/O ***************************************
; Вход A1 - не используется
; Вход A2 - не используется
; Вход A3 - вход START/STOP
; Вход A4 - не используется
; Вход B1 - не используется
; Вход B2 - не используется
; Вход B3 - не используется
; Вход B4 - не используется
; Вход C1 - не используется
; Вход C2 - не используется
; Вход C3 - не используется
; Вход C4 - не используется
; Выход 1 - Манипулятор
; Выход 2 - Захват
; Выход 3 - не используется
; Выход 4 - не используется
;************************Инициализация и множественные переменные************************
UNITS = 1
ACCEL = 75
DECEL =75
MAXV = 10
;V1 =
;V2 =


;********************************** События *************************************
;Задайте здесь обработку событий
;***************************** Главная программа ********************************
PROGRAM_START:
```

```
ENABLE
MOVED  10                ; Переход в положение Взять
OUT1 = 1                 ; Включить выход 1 для удлинения манипулятора
WAIT TIME 500            ; Задержка в полсекунды для удлинения манипулятора
OUT2 = 1                 ; Включить выход 2 для подключения захвата
WAIT TIME 500            ; Задержка в полсекунды чтобы взять деталь
OUT1 = 0                 ; Выключить выход 1, чтобы отвести манипулятор
MOVED -10                ; Движение в положение Места
OUT1 = 1                 ; Включить выход 1 для удлинения манипулятора
WAIT TIME 500            ; Задержка в полсекунды для удлинения манипулятора
OUT2 = 0                 ; Выключить выход 1, чтобы Расцепить захват
WAIT TIME 500            ; Задержка в полсекунды чтобы положить деталь на место
OUT1 = 0                 ; Отвести манипулятор
GOTO PROGRAM_START
END
;******************************** Подпрограммы ************************************
Здесь ввести код подпрограммы
;************************** Программа устранителя неисправностей ********************
; Здесь ввести код устранителя неисправностей
ON FAULT
ENDFAULT
```

Сохранение файла конфигурации в ПК

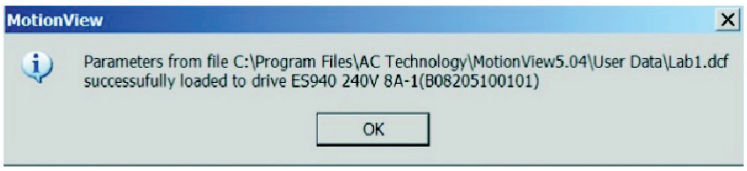
«Конфигурационный файл» содержит все настройки параметров для привода, а также пользовательскую программу. После завершения вами настройки параметров привода и написания пользовательской программы вы можете сохранить эти настройки в вашем компьютере. Чтобы сохранить настройки выберите из Узлового раскрывающегося меню «Сохранить конфигурацию как» (“**Save configuration As**”). Затем просто присвойте вашей программе имя (Основное движение (Basic Motion)) и нажмите Сохранить (**Save**). Конфигурационный файл имеет расширение “dcf” и будет сохранен по умолчанию в «Данные пользователя» (“**User Data**”) в «MotionView».

Загрузка файла конфигурации в привод

Возникают ситуации, когда желательно импортировать (или экспортировать) источник программы в другой привод. В других случаях программа пишется автономно. В обоих этих случаях программу или конфигурационный файл необходимо загрузить из ПК в привод. Для загрузки файла конфигурации в привод, выберите из Узлового раскрывающегося меню «Загрузить конфигурационный файл в привод» (“**Load configuration file to drive**”). Далее просто выберите программу, которую хотите загрузить, и нажмите Открыть (**Open**). Вначале «MotionView» скомпилирует выбранную программу. После компиляции должно появится следующее окно сообщений.

Компиляция завершена без ошибок.

Нажмите “OK”, чтобы закрыть диалоговое окно. Далее «MotionView» загрузит выбранный файл в привод и после завершения загрузки отобразит следующее окно сообщений.

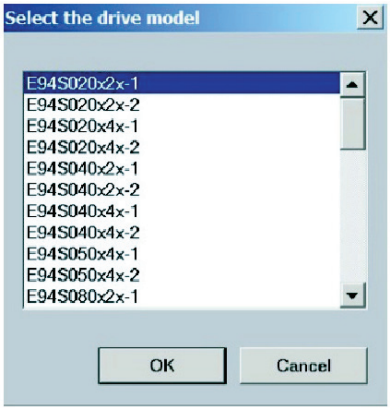


Загрузка параметров из C:\Program Files\AC Technology\MotionView5.04\User Data\Lab1.dcf в привод ES940 240V 8A-1(B08205100101) успешно завершена.

Нажмите “OK” чтобы закрыть диалоговое окно.

Создание нового конфигурационного файла

Возникают ситуации, когда необходимо разработать новое приложение, а у вас нет соединения с приводом. Эту задачу можно выполнить, загрузив виртуальный привод. Для создания нового конфигурационного файла выберите из Узлового раскрывающегося меню «Новый конфигурационный файл» (“**New configuration file**”). Должно появится следующее окно сообщений.



Выберите нужный привод и нажмите “OK”. На дерево параметров загрузится виртуальный привод. Отсюда вы сможете установить всю настройку параметров, а также создать пользовательскую программу. По завершении, для того, чтобы сохранить вашу работу, можно использовать вышеописанную операцию «**Сохранение конфигурационного файла в ПК**». Позже работу над программой можно продолжить автономно, выбрав из Узлового раскрывающегося меню «Открыть конфигурационный файл» (“**Open configuration file**”).

Режим движения (Адресация)

940 можно запрограммировать для работы в одном из трех режимов: Момент, Скорость, или Положение. Прежде чем осуществить какое-либо движение, приводу нужно получить команду. Источник, подающий команды для этого движения называется «Адресация» (“Reference”). Существует два способа управления движения для 940, или два типа адресации. Если сигнал команды приводу поступает с внешнего источника, например, PLC или Приводного Контроллера, это называется внешняя адресация. Если команда приводу поступает из пользовательской программы или через одну из системных переменных, это называется внутренняя адресация. (См. таблицу ниже)

Режим	Установка параметра «Адресация» (“Reference”)	
	Внешний	Внутренний
Крутящий момент	Аналоговый вход AIN1	Системная переменная “IREF”
Скорость	Аналоговый вход AIN1	Системная переменная “IREF”
Положение	MA/MB вводы серии импульсов + пользовательская программа/Интерфейс (Выход генератора траектории)	Пользовательская программа/Интерфейс (Генератор траектории)

Единицы

Все предложения о перемещениях в приводе работают с единицами пользователя. Предложение на первой линии тестовой программы, UNITS=1, устанавливает отношения между единицами пользователя и оборотами двигателя. Оно просто отвечает на вопрос: “Сколько единиц пользователя содержится в одном обороте вала двигателя?” Если данное предложение отсутствует в программе, двигатель будет оперировать показаниями датчика, используя их в качестве единиц пользователя.

Временная ось

Временная ось всегда исчисляется в секундах, т.е. все величины, имеющие отношение ко времени, задаются в Единицах пользователя/сек (USER UNITS/SEC)

Включение/Выключение/Запрет привода

Установка функции включения на Запуск.

940 привод можно конфигурировать таким образом, чтобы получать адресацию команд из внешнего источника и не использовать внутреннюю пользовательскую программу для механизма движения. В основном, действия те же, что и для 94 привода. Привод будет включен, когда параметр функции включения установлен на Запустить, и вход 3A включен. Также,переключение входа 3A в состояние «выключено» выключит привод.

- Выберите из окна дерева параметров «Параметр» (“**Parameter**”).
- Выберите из окна обзора параметров «функция включения» (“**Enable switch function**”).
- Выберите из всплывающего меню «Запустить» (“**Run**”).

Установка функции включения на Запрет.

В описанном выше примере решение о том, когда включать и выключать привод определяется внешним устройством, PLC или контроллером движения. Пользовательская программа 940 позволяет программисту принимать решение и ввести его в программу привода. Привод выполнит пользовательскую программу по умолчанию, будь-то включенным или выключенным, однако если команда перемещения выполнена, когда привод выключен, появится ошибка. Когда параметр «**Функции включения**»

(**“Enable switch function”**) установлен на **Запрет**, а вход 3А включен, привод по умолчанию выключится и будет включен при выполнении пользовательской программой команды ENABLE.
- Выберите из окна дерева параметров «**Параметр**» (“**Parameter**”).
- Выберите из окна обзора параметров «**Функция включения**» (“**Enable switch function**”).
- Выберите из всплывающего меню «**Запретить**» (“**Inhibit**”).

Ошибки
Когда привод выявляет ошибку, имеют место следующие события:
- Если программа выполняет код пользователя, выполнение кода останавливается. Если программа устранителя неисправностей обнаружила ошибку, запускается код по ее устранению. (Подробнее см. ниже Устранитель неисправностей). Если ошибки устранены – пользовательская программа завершает выполнение.
- Код ошибки будет записан в регистре DFAULTS и станет доступным пользовательской программе. См. перечень кодов ошибок в разделе 2.15
- Выделенный выход «Готов к работе» (“Ready”) переключится в состояние ВЫКЛ (OFF).
- Выход с заданной специальной функцией «неисправность» переключится в положение ВКЛ (ON).
- Выход с заданной специальной функцией «готов к работе/включен» (“ready/enabled”) переключится в положение ВЫКЛ (OFF).
- LED индикатор включения, расположенный на передней панели привода переключится в положение ВЫКЛ (OFF):
- На дисплее отобразится F_XX, где XX – это номер кода ошибки.
Неисправность можно устранить одним из следующих способов:

-Выберите на панели инструментов пиктограмму «Перезапустить» (“**Restart**”).
-Выполнение предложения RESUME в конце программы устранителя неисправностей (см. пример устранителя неисправностей).
- Отправьте команду «Перезагрузить» (“**Reset**”) через главный интерфейс.
- Выключить компьютер и включить его снова (полная перезагрузка).

Устранитель неисправностей
Устранитель неиправностей является частью кода, который выполняется в случае генерирования приводом ошибки. Это позволяет программе в значительной степени восстановиться, вместо того, чтобы вывести привод из строя. Выход из Кода Устранителя Неисправностей осуществляется предложением “REST” или “RESUME”.

Без Устранителя Неисправностей
Для симуляции ошибки перезапустите программу-образец «Pick and Place». Когда программа запущена, отсоедините от привода коннектор обратной связи от датчика положения. Это вызовет генерирование приводом Ошибки Обратной связи (Feedback Error (F_Fb)) и переведет привод в режим неисправности. Пока привод находится в режиме неисправности (Fault Mode), выходы останутся в текущем состоянии (каждый включенный выход останется включенным), остановится выполнение программы, но механизм будет продолжать движение. Например, манипулятор «Pick and Place» может находиться в нежелательном месте, когда программа переходит в режим неисправности.

С Устранителем Неисправностей
Добавьте приведенную ниже программу Устранителя Неисправностей в программу “Pick and Place” и перезапустите программу. Когда программа запущена, снова отсоедините от привода коннектор обратной связи от датчика положения. Это сгенерирует Ошибку Обратной связи (Feedback Error, (F_Fb)). С этой точки программа неисправности начнет действовать и проконтролирует, чтобы выходы ввели устройство в безопасное положение.

```
;**************************Программа Устранителя Неисправностей**************************
ON FAULT            ;состояние запуска программы устранителя неисправностей
                    ;Движение остановлено, привод выключен, события закончились
                    ;во время выполнения программы устранителя неисправностей произведен поиск.

OUT2 = 0            ;Выход 1 выключен для выключения захвата.
                    ;Это опустит деталь, которая находилась в захвате
OUT1 = 0            ;Отвести манипулятор и убедиться, что он поднят и вне зоны доступа
RESUME Fprocess     ;программа перезапускается с отметки Fprocess
ENDFAULT            ;устранитель неисправностей должен завершиться этим предложением
```

Примечание

Данные команды нельзя использовать в Коде Устранителя неисправностей:

- ENABLE
- WAIT UNTIL
- MOVE
- MOVED
- MOVEP
- MOVEDR
- MOVEPR
- MDV
- MOTION SUSPEND
- MOTION RESUME
- GOTO, GOSUB
- JUMP
- ENABLE
- GEAR ON/OFF
- VELOCITY ON/OFF

См. раздел 2.1 для более подробной информации и раздел Языковые таблицы касательно предложения “ON FAULT/ENDFAULT”.

1.6 Использование расширенных возможностей отладки

Выберите на панели инструментов пиктограмму «Перезапустить» (“Restart”) для повторного запуска программы с начала.

Выберите на панели инструментов пиктограмму «Скачок» (Step into)

или пиктограмму «Перешагнуть» (Step over)
для выполнения программных предложений построчно.

Выбрав на панели инструментов пиктограмму «Вставить/Удалить точки прерывания программы», пользователь может установить точки прерывания по всей программе. Привод будет выполнять программу построчно, пока дойдет до одной из точек прерывания программы. В этой точке программа остановится, позволив пользователюдать оценку программным переменным, проверить ветвление программы или просто проверить выполнение кода.

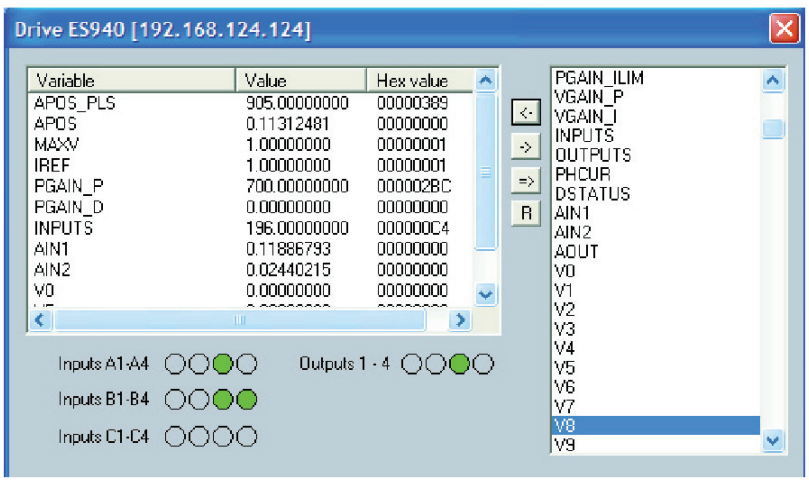
Для продолжения кодовой обработки информации можно работать с программой, используя описанную выше операцию, или можете выбрать на панели инструментов пиктограмму «Пуск».

Чтобы открыть Окно отладки переменных выберите на панели инсрументов пиктограмму «Обзор отладки» (“Debug View”). В окне отладки открывается система привода и пользовательская переменная, а также статус I/O.

Для добавления переменных используйте клавишу со стрелкой влево.

Для удаления переменных используйте клавишу со стрелкой вправо.

Используйте клавишу «Обновить» (“Refresh”) для обновления величины переменных. Обратите внимание, что величина переменных обновляется вручную при нажатии на кнопку «Обновить» или автоматически при остановке программы, по завершении шага или по достижении точки прерывания программы.



1.7 Входыи Выходы

Аналоговые входыи выходы

- 940 имеет два аналоговых входа. Эти аналоговые входы используются приводом как системные переменные и помечены “AIN1” и “AIN2”. Через них возможен прямой доступ к величине, (ее можно прочесть) через пользовательскую программу или через главный интерфейс. Эта величина может варьироваться от -10 до +10, что сопоставимо с ± 10 вольтами.
- 940 имеет один аналоговый выход. Этот аналоговый выход используется приводом как системная переменная и помечен “AOUT”. К нему возможен прямой письменный доступ через пользовательскую программу или через главный интерфейс. Его величина может варьироваться от -10 до +10 что сопоставимо с ± 10 вольтами.

Примечание

Если аналоговому выходу в «MotionView» назначена специальная функция, письменный запрос пользовательской программы к AOUT не изменит его величину.
Если аналоговому выходу установлен статус «Не присваивать», тогда он управляется переменной AOUT.

Цифровые входы

- 940 имеет двенадцать цифровых входов. Привод использует эти цифровые входы для принятия решений в пользовательской программе. Примерами могут послужить выключатели ограничения перемещения, бесконтактные датчики, нажимные кнопки и взаимодействие с другими устройствами.
- Через «MotionView» каждому входу можно присвоить индивидуальное время дребезга. Выберите “**IO**” из дерева параметров. Затем выберите папку «Цифровой вход» (“**Digital Input**”). В окне обзора параметров отобразится время дребезга. Его величину можно задть от 0 до 1000 мс (1мс = 0.001 сек).
- Двенадцать входов разделены на три группы: A, B и C. Каждая группа имеет четыре входа, и один общий вход, используемый группой совместно: Acom, Bcom и Ccom. Входы помечены индивидуально как **IN_A1 - IN_A4**, **IN_B1 - IN_B4** и **IN_C1 - IN_C4**.
- Помимо мониторинга выключения для каждого входа в отдельности все двенадцать входов можно представить одним двоичным числом. У каждого входа есть свой битовый номер от 0 до 11 для Входных сигналов (INPUTS) системных переменных. Каждый вход выделяет ресурс в 1 бит для переменной входного сигнала (INPUTS). См. таблицу ниже:

Вход	Номер бита системной переменной входного сигнала
A1	0
A2	1
A3	2
A4	3
B1	4
B2	5
B3	6
B4	7
C1	8
C2	9
C3	10
C4	11

Некоторые входы имеют дополнительное специальное функциональное назначение (выключатель ограничения перемещения, вход сигнала разрешения и вход привода). Конфигурация данных входов выполняется при помощи «MotionView». Обобщение функционального назначения входа приведено в таблице ниже и в следующих разделах. Программист получает доступ к статусу текущего состояния входов приводов через выделенные системные флаги или как к битам входных сигналов (INPUTS) системных переменных. Таблица, приведенная ниже, предоставляет сводные данные по входам:

Назначение	Специальное назначение
Вход A1	концевой выключатель (по часовой стрелке)
Вход A2	концевой выключатель (против часовой стрелки)
Вход A3	вход Запретить/Включить
Вход A4	
Общий для группы A	
Вход B	
Вход B2	
Вход B3	
Вход B4	
Общий для группы B	

Вход C1	
Вход C2	
Вход C3	Регистрация входного сигнала от датчика
Вход C4	
Общий для группы C	

Считывание цифровых входов

Ниже приводится программа образец «Pick and Place», измененная таким образом, чтобы использовать состояния входов “WAIT UNTIL” вместо состояний “WAIT TIME”. **IN_A4** используется в качестве бесконтактного датчика для фиксирования момента, когда манипулятор вытянут и когда он отведен назад. Когда манипулятор вытянут, **IN_A4** включится и будет равен “1”. Когда манипулятор отведен назад, **IN_A4** выключится и будет равен “0”.

```
;********************************** Главная программа ***********************************
ENABLE
PROGRAM_START:
WAIT UNTIL IN_A4==0            ;Убедитесь, что манипулятор отведен назад
MOVED  10                      ;Движение в положение взять
OUT1 = 1                       ;Включить выход 1, чтобы вытянуть манипулятор
WAIT UNTIL IN_A4 == 1          ;Манипулятор вытянут
OUT2 = 1                       ;Включить выход 2 для активации захвата
WAIT TIME 1000                 ;Задержка в 1 сек чтобы взять деталь
OUT1 = 0                       ;Выключить выход 1, чтобы отвести назад манипулятор
WAIT UNTIL IN_A4==0            ;Убедитесь, что манипулятор отведен назад
MOVED -10                      ;Перевести в положение места
OUT1 = 1                       ;Включите выход 1, чтобы вытянуть манипулятор
WAIT UNTIL IN_A4 == 1          ;Вытягивание манипулятора
OUT2 = 0                       ;Выключить выход 1, чтобы деактивировать захват
WAIT TIME 1000                 ;Задержка в 1 сек, чтобы положить деталь на заданное место
OUT1 = 0                       ;Отвести назад манипулятор
WAIT UNTIL IN_A4 == 0          ;Манипулятор отведен назад
GOTO
PROGRAM_START
END
```

После внесения вами описанных изменений, экспортируйте программу в файл и сохраните ее как “Pick and Place with I/O”, затем скомпилируйте, загрузите и протестируйте программу.

ASSIGN & INDEX – Использование входов для генерации предопределенных индексов

"**INDEX**" является переменной привода, которую можно сконфигурировать для того, чтобы как двоичное число представлять определенную совокупность входов. "**ASSIGN**" является командой, которая конфигурирует какие входы использовать и как мы хотим их сконфигурировать.

Ниже приведена программа «Pick and Place», модифицированная для использования функции "INDEX". В предыдущем примере программа просто брала деталь и переносила ее на место расположения. В демонстрационных целях добавим семь вариантов окончания программы, или мест расположения. Эти места расположения назовем Корзинами (Bins). Состояние трех входов B1, B2 и B3 определит, в какой корзине находится деталь.

Корзина 1 - Вход B1 включен
Корзина 2 - Вход B2 включен
Корзина 3 - Входы B1 и B2 включены
Корзина 4 - Вход B3 включен
Корзина 5 - Входы B1 и B3 включены
Корзина 6 - Входы B2 и B3 включены
Корзина 7 - Входы B1, B2 и B3 включены

Команда "ASSIGN" используется для назначения бита переменной "INDEX" отдельному входу. ASSIGN INPUT(НАЗНАЧИТЬ ВХОД) <название входа> AS BIT (КАК БИТ)<номер бита>

```
;************************ Инициализировать и задать переменные ************************
ASSIGN INPUT IN_B1 AS BIT 0   ;Назначить переменную INDEX равной 1 при включенном IN_B1
ASSIGN INPUT IN_B2 AS BIT 1   ;Назначить переменную INDEX равной 2 при включенном IN_B2
ASSIGN INPUT IN_B3 AS BIT 2   ;Назначить переменную INDEX равной 4 при включенном IN_B4
```

Место расположения корзины	Состояние входа	Величина INDEX
Корзина 1	Вход включен	1
Корзина 2	Вход B2 включен	2
Корзина 3	Входы B1 и B2 включены	3
Корзина 4	Вход B3 включен	4
Корзина 5	Входы B1 и B3 включены	5
Корзина 6	Входы B2 и B3 включены	6
Корзина 7	Входы B1, B2 и B3 включены	7

Ниже Главная программа была модифицирована, чтобы изменить конечное положение места на основе величины переменной "INDEX".

```
;******************************** Главная программа *********************************
ENABLE
PROGRAM_START:
WAIT UNTIL IN_A4==0       ;Убедитесь, что манипулятор отведен назад
MOVEP  0                  ;Передвинуть на (ABS) в положение «взять»
OUT1 = 1                  ;Включить выход 1, чтобы вытянуть манипулятор
WAIT UNTIL IN_A4 == 1     ; Манипулятор вытянут
OUT2 = 1                  ;Включить выход 2 для активации захвата
WAIT TIME 1000            ;Задержка в 1 сек, чтобы взять деталь
OUT1 = 0                  ;Выключить выход 1, чтобы отвести назад манипулятор
WAIT UNTIL IN_A4==0       ;Убедитесь, что манипулятор отведен назад
IF INDEX == 1             ;В данной области мы используем предложение If для
GOTO BIN_1                ;проверьте, в каком состоянии находятся входы B1, B2 и B3
ENDIF                     ;
IF INDEX == 2             ;   INDEX = 1 при включенном входе B1
GOTO BIN_2                ;   INDEX = 2 при включенном входе B2
ENDIF                     ;   INDEX = 3 при включенных входах B1 & B2.
                          ;   INDEX = 4 при включенном входе B3
                          ;   INDEX = 5 при включенных входах B1 & B3.
                          ;   INDEX = 6 при включенных входах B2 & B3.
IF INDEX == 7             ; INDEX = 7 при включенных входах B1, B2 & B3
GOTO BIN_7                ;Теперь мы можем направить программу к одному из семи
ENDIF                     ;мест расположения на основе трех входов.
BIN_1:                    ;Установка для Корзины 1
MOVEP 10                  ;Перейти к месту расположения Корзины 1
GOTO PLACE_PART           ;Скачок к программе места детали
BIN_2:                    ;Установка для Корзины 2
MOVEP 20                  ;Перейти к месту расположения Корзины 2
GOTO PLACE_PART           ;Скачок к программе места детали
BIN_7:                    ;Установка для Корзины 7
```

```
MOVEP 70                    ;Перейти к месту расположения Корзины 7
GOTO PLACE_PART             ;Скачок к программе места детали
PLACE_PART:
OUT1 = 1                    ;Включить выход 1, чтобы вытянуть манипулятор
WAIT UNTIL IN_A4 == 1       ;Манипулятор вытянут
OUT2 = 0                    ;Выключить выход 1 для деактивации захвата
WAIT TIME 1000              ;Задержка в 1 сек, чтобы положить деталь на место положения
OUT1 = 0                    ;Отвести назад манипулятор
WAIT UNTIL IN_A4 == 0       ;Манипулятор отведен назад
GOTO PROGRAM_START
END
```

Примечание

Примечание: Любому из 12 входов можно назначить битовое положение в пределах INDEX переменной. С INDEX переменной можно использовать только биты от 0 до 7. Биты 8-31 не используются и всегда задаются как 0. Не назначаемые в INDEX переменной биты задаются как 0.

BITS 8-31 (не используются)	A1	0	A2	A4	0	0	0	0

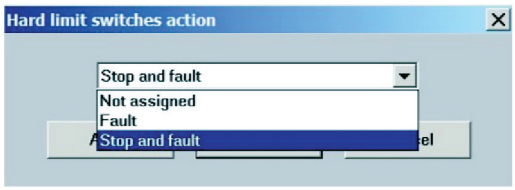
Функции входа концевого выключателя

Входы A1 и A2 можно сконфигурировать как входы специального назначения. Их конфигурируют из цифровой I/O папки в MotionView и задают им одну из трех следующих настроек.

- Настройка **«Не назначать»** (“**Not assigned**”) обозначает входы как входы общего назначения, используемые пользовательской программой.
- Настройка **«Неисправность»** (“**Fault**”) конфигурирует A1 и A2 как аппаратные концевые выключатели (Hard Limit Switches). Когда любой из входов включен, привод будет заблокирован, двигатель – принудительно остановлен, а привод сгенерирует неисправность.
- Настройка **«Остановка и неисправность»** (“**Stop and fault**”) сконфигурирует A1 и A2 как ограничительные концевые выключатели. Когда любой из входов включен, привод инициирует быструю остановку перед тем, как будет заблокирован и сгенерирует неисправность. Скорость торможения будет задана величиной, сохраненной в системной переменной “QDECEL”.

Примечание

Функция «Остановка и Неисправность» (“Stop and Fault”) доступна в режиме положения, когда параметр адресации задан как внутренний, т. е. когда команда движения выведена из пользовательской программы. Во всех остальных случаях «Остановка и Неисправность» будет функционировать также как и функция «Неисправность».

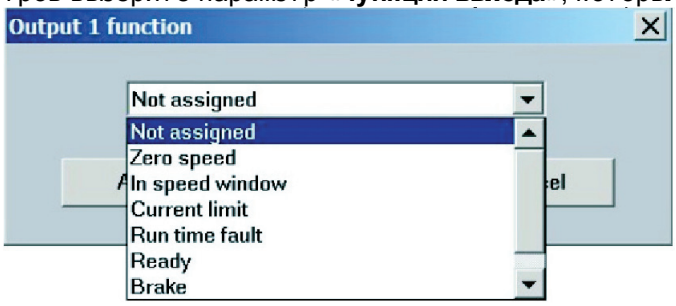


Stop and Fault – Остановка и Неисправность
Not assigned – Не назначено
Fault - Неисправность

Для установки этого параметра выберите из дерева параметров папку “IO”. Далее выберите папку “Цифровые IO”. Из окна обзора параметров выберите “Действие аппаратного концевого выключателя”.

Контроль цифровых выходов

- 940 имеет 5 цифровых выходов. “**RDY**” или READY (готовность) выход предназначен и активируется только при включении привода, а также в режиме **RUN** (работа). Остальные выходы отмечены каждый в отдельности как **OUT1 - OUT4**.
- Выходы можно сконфигурировать как выходы специального назначения. Если выход сконфигурирован как **выход специального назначения**, он будет активирован при активации присвоенного ему состояния. Например, если выходу присвоена функция «**Нулевая скорость**» (**“Zero speed”**), выход с таким назначением включится, когда двигатель не будет вращаться. Чтобы конфигурировать выход как выход специального назначения, выберите из дерева параметров папку “**IO**”. Далее выберите папку “**Цифровые IO**”. Из окна обзора параметров выберите параметр «**Функции выхода**», который вы хотите установить.



Not assigned – не присвоено, Zero speed – нулевая скорость, In speed window – в окне скорости, Current limit – токовый предел Run time fault – неисправность времени выполнения программы, Ready – готово, Brake – тормоз

- Выходы, не сконфигурированные как выходы специального назначения, можно активировать через пользовательскую программу или через главный интерфейс. Если выходу присвоен статус выхода специального назначения, ни пользовательская программа, ни главный интерфейс не смогут отменить его статус.
- Системная переменная “OUTPUTS” является переменной считывания и записи, которая позволяет пользовательской программе или главному интерфейсу осуществлять контроль и задавать статус всех четырех выходов одним двоичным числом. Каждый выход распределяет 1 бит в переменной OUTPUTS. Например, если вы зададите данную переменную равной 15 в пользовательской программе (OUTPUTS = 15), тогда включатся все 4 выхода.
- Приведенные ниже примеры подводят итог функциям выходов и соответствующим системным флагам. Для присвоения значения выходу, введите его флагу данные любой величины не равной 0 (действительной (TRUE)). Для очистки выхода, введите его флагу величину равную 0 (недействительную (FALSE)). Флаги можно также использовать в выражении. Если выражение оценивается как TRUE (действительное), выход будет переведен в положение ON (ВКЛ). Иначе он будет переведен в положение OFF (ВЫКЛ).

```
OUT1 = 1                        ;включите OUT1
OUT2 = 10                       ;любая отличная от 0 величина включает выход
OUT3 = 0                        ;выключите OUT3
OUT2 = APOS>3 && APOS<10        ;ВКЛ в положении внутри окна, иначе ВЫКЛ
```

1.8 События

Сканирование событий

Сканирование события является маленькой программой, работающей независимо от главной. SCANNED EVENTS очень полезно, в случае необходимости запустить действие, (оперировать I/O), пока двигатель в движении. В приведенном примере событие “SPRAY_GUNS_ON” будет задано, чтобы включить Выход 3 при положении сервомеханизма больше 25. (Примечание: событие запустится только в тот момент, когда положение сервомеханизма превысит 25).

```
;********************** Установка события **********************************
EVENT SPRAY_GUNS_ON  APOS>25
OUT3=1
ENDEVENT
;***************************************************************************
```

Код события необходимо ввести в раздел программы EVENT SETUP (установка события). Для SETUP (установки) События необходимо ввести команду “EVENT”. Далее имя события “SPRAY_GUNS_ON” и механизм запуска “APOS>25”. Следующим шагом будет ввод «предложений» или «кода». Это желаемая последовательность программных событий, которая будет иметь место после запуска события. В нашем случае мы включим выход 3, “OUT3=1”. Для завершения установки события необходимо ввести команду “ENDEVENT”.

События можно активировать (Включить) и деактивировать (Выключить) по ходу программы. Чтобы включить событие, вводится команда “**EVENT**”, и за ней имя события “**SPRAY_GUNS_ON**”. К этому с опозданием добавляется желаемое состояние события “**ON**” или “**OFF**”.

```
;*****************************************************************************************
EVENT SPRAY_GUNS_ON                          ON
;*****************************************************************************************
```

Чтобы узнать больше о сканировании событий, обратитесь к разделу 2.13.
Два события сканирования были добавлены к программе «Pick and Place» (приведенной ниже). Эти события будут использованы для запуска включения и выключения распылителя. Событие запустят после того, как механизм возьмет деталь и будет нести ее перед распылителями (POS 25). Как только деталь окажется в нужном положении, включится выход 3 для активации распылителей. Как только деталь минует распылители, (POS 75), выход 3 выключится, деактивировав распылители.

```
;********************************** События ******************************************
EVENT SPRAY_GUNS_ON  APOS>25
OUT3=1
ENDEVENT
EVENT SPRAY_GUNS_OFF  APOS>75
OUT3=0
ENDEVENT
;******************************** Главная программа ************************************
PROGRAM_START:
ENABLE
EVENT SPRAY_GUNS_ON ON
EVENT SPRAY_GUNS_OFF ON
WAIT UNTIL IN_A4==0         ;Убедитесь, что манипулятор отведен назад
MOVEP 0                     ;Перейти в положение «взять»
OUT1 = 1                    ;Включить выход 1, чтобы вытянуть манипулятор
WAIT UNTIL IN_A4 == 1       ;Вытягивание манипулятора
OUT2 = 1                    ;Включить выход 2 для активации захвата
WAIT TIME 1000              ;Задержка в 1 сек, чтобы взять деталь
OUT1 = 0                    ;Выключить выход 1, чтобы отвести назад манипулятор
WAIT UNTIL IN_A4==0         ;Убедитесь, что манипулятор отведен назад
MOVEP 100                   ;перейти в положение места
OUT1 = 1                    ;Включить выход 1, чтобы вытянуть манипулятор
WAIT UNTIL IN_A4 == 1       ;вытягивание манипулятора
OUT2 = 0                    ;Выключить выход 1 для отключения захвата
WAIT TIME 1000              ;Задержка в 1 сек, чтобы положить деталь на место
OUT1 = 0                    ;Отвести назад манипулятор
WAIT UNTIL IN_A4 == 0       ; Манипулятор отведен назад
GOTO PROGRAM_START
END
```

1.9 Переменнье и оператор определения

Переменные являются ресурсами в приводе. Некоторые из этих переменных можно использовать для считывания и записи, а иные – только для считывания. Определенные переменные используются для установки рабочих параметров привода, например, (ACCEL, DECEL, или MAXV). Другие переменные можно использовать для определения статуса привода (AIN, INPUTS, или APOS). Переменные можно также использовать как системные регистры. Эти системные регистры могут быть местными по

отношению к приводу (V1- V31), или сетевым переменным (N1 - N31). В приведенном ниже примере зададим положение триггера для события “**SPRAY_GUNS_ON**” равным “V1”, а положение триггера для события “**SPRAY_GUNS_OFF**” равным “V2”.
Команда определения **DEFINE** используется для того, чтобы присвоить имя состоянию переменной привода, (Выход ВКЛ = 1, Выход ВЫКЛ = 0). Можно также присвоить заданному числу имя, (MIN = 25, MAX = 75). В приведенном ниже примере мы назначаем имя «Выход включен» равным величине “1”, и «Выход выключен» - равным величине “0”. Определение и задание переменных должно производится в программном разделе «**Инициализировать и задать переменные**».

```
;************************ Инициализировать и задать переменные ****************************
UNITS = 1
ACCEL = 5
DECEL = 5
MAXV = 10
V1 = 25                              ;Задать Переменную V1 равной 25
V2 = 75                              ;Задать Переменную V2 равной 75
DEFINE      Выход включен  1         ;Определить Имя для включенного выхода
DEFINE      Выход выключен 0         ;Определить Имя для выключенного выхода
;************************************* СОБЫТИЯ  ***************************************
EVENT SPRAY_GUNS_ON  APOS > V1       ;Событие запустится когда положение пройдет 25
                                     ;в положительном направлении
OUT3= Output_On                      ;Включить распылители (выход 3 включен)
ENDEVENT                             ;окончание события
EVENT SPRAY_GUNS_OFF  APOS > V2      ;Событие запустится когда положение пройдет 75
                                     ;в отрицательном направлении.
OUT3= Output_Off                     ;Выключить распылители (выход 3 выключен)
ENDEVENT                             ;окончание события
;************************ Главная программа ***********************************
PROGRAM_START:
ENABLE
EVENT SPRAY_GUNS_ON ON       ;Активировать событие
EVENT SPRAY_GUNS_OFF ON      ;Активировать событие
WAIT UNTIL IN_A4==0          ;Перед запуском программы убедитесь, что манипулятор отведен назад
MOVEP 0                      ;Перейти в положение 0, чтобы взять деталь.
OUT1 = Output_On             ;Включить выход 1, чтобы вытянуть манипулятор
WAIT UNTIL IN_A4 == 1        ;Проверьте вход и убедитесь, что манипулятор вытянут
OUT2 = Output_On             ;Включить выход 2 для активации захвата
WAIT TIME 1000               ;Задержка в 1 сек, чтобы взять деталь
OUT1 = Output_Off            ;Выключить выход 1, чтобы отвести манипулятор назад
WAIT UNTIL IN_A4==0          ;Проверить вход и убедиться, что манипулятор отведен назад
MOVED 100                    ;Перейти в положение места
OUT1 = Output_On             ;Включить выход 1, чтобы вытянуть манипулятор
WAIT UNTIL IN_A4 == 1        ;Проверить вход и убедиться, что манипулятор вытянут
OUT2 = Output_Off            ;Выключить выход 1 для отключения захвата
WAIT TIME 1000               ;Задержка в 1 сек чтобы положить деталь на место
OUT1 = Output_Off            ;Отвести назад манипулятор
WAIT UNTIL IN_A4 == 0        ;Проверить вход и убедиться, что манипулятор отведен назад
GOTO PROGRAM_START
END
```

1.10 IF/ELSE операторы

IF/ELSE (если/в противном случае) оператор позволяет выполнять условно одно или больше предложений. Можно использовать IF или конструкцию IF/ELSE:

Пример с IF:

Этот пример дает приращение счетчику, Переменной “V1”, пока Переменная, “V1”, не превысит 10.

```
Again:
V1=V1+1
IF          V1>10
V1=0
ENDIF
GOTO    Again
END
```

Пример IF/ELSE:

Этот пример проверяет величину переменной V1, Если V1 больше 3, тогда V2 задается 1. Если V1 не больше 3, тогда V2 задается 0.

```
IF V1>3
V2=1
ELSE
V2=0
ENDIF
```

Независимо от того, используете ли вы оператора IF или IF/ELSE, конструкция должна заканчиваться ключевым словом ENDIF.

1.11 Механизм работы

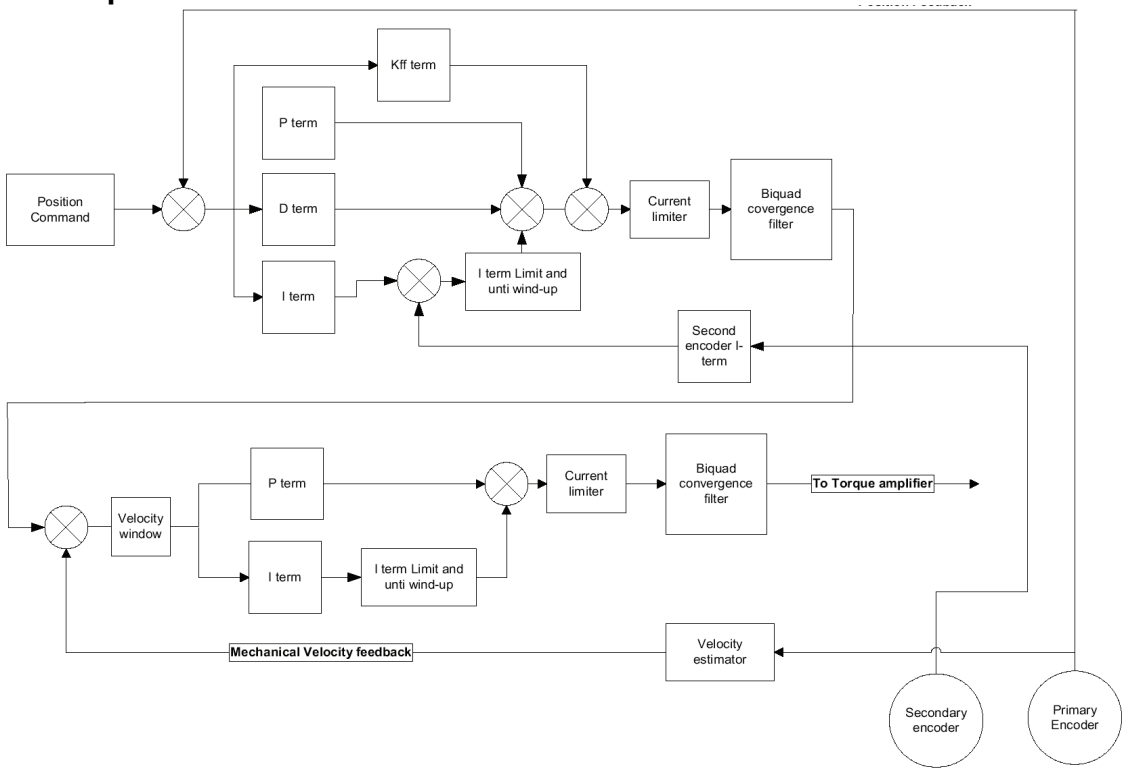


Диаграмма регулятора положения и скорости 940

Position Command – команда положения, P term – P терм, D term – D терм, I term – I терм, Kff term – Kff терм, Velocity window – окно скорости, Mechanical Velocity feedback – обратная связь по механической скорости, I term Limit and unti wind-up – от предела I терма и до конца выполнения, Position Feedback – позиционная обратная связь, Current limiter – ограничитель тока, Biquad covergence filter – биквадный фильтр конвергенции, Second encoder I-term – I-терм второго кодировщика, To Torque amplifier – К усилителю силового привода, Velocity estimator – блок оценки скорости, Primary Encoder – первичный кодировщик, Secondary encoder – вторичный кодировщик

«Команда положения» (**“Position Command”),** как показано выше на диаграмме регулятора, производится генератором траектории (**Trajectory Generator).** Генератор траектории обрабатывает команды движения, производимые пользовательской программой, чтобы рассчитать инкремент и декремент положения, называемые также «индексная» величина (“index” value), для каждого цикла сервомеханизма. Это рассчитываемое заданное (или теоретическое) положение затем подается на вход регулятора (**Regulator)**.

Главной задачей регулятора (**Regulator)** является задавать положение двигателей в соответствии с генератором траектории. Это осуществляется для контроля момента и скорости двигателя сравнением входа генератора траектории с положением обратной связи от энкодера. Конечно, всегда будет иметь место некоторая динамическая ошибка, которая носит название «Ошибка положения» (“Position Error”) и выражается следующим образом:

Ошибка положения = Заданное положение - Фактическое положение

При превышении погрешности положения пороговой величины «Position Error Excess», генерируется сообщение о неисправности (F_Fb). Допустимые Порог и Время ошибки положения (время до вывода сообщения о неисправности) можно задать. Эти параметры могут быть заданы только в «MotionView». (Из узлового дерева – Папка порогов/ Пороги положения (Limits folder/Position Limits))

Режимы управления

Существуют три режима работы для 940: Управление Моментом, Скоростью и Положением. Режимы управления по моменту и по скорости широко используются при адресации команд из внешнего устройства, (Ain). Режим управления положением используют при поступлении команды из пользовательской программы привода, или из внешнего устройства, энкодера или импульса перемещения и направления. Настройка режима привода осуществляется из папки **«Параметр»** (“**Parameter**”) в «MotionView». Для управления движением из пользовательской программы, привод должен быть в режиме управления положением. Даже если установка привода проведена в режиме положения, из пользовательской программы можно включать/выключать режим управления по скорости. Для активации этого режима выполняется команда VELOCITY ON, в то время как для деактивации выполняется VELOCITY OFF. Этот режим используется в особых случаях индексированного движения. Режим скорости – это режим, в котором заданная системная переменная VEL постоянно опережает заданное положение. В режиме электронной редукции (Gear mode) положение задается входами MA/MB, с масштабированием (передаточное число задается коэффициентом Gear Ratio). Приведенная ниже диаграмма показывает схемы адресации для различных режимов работы.

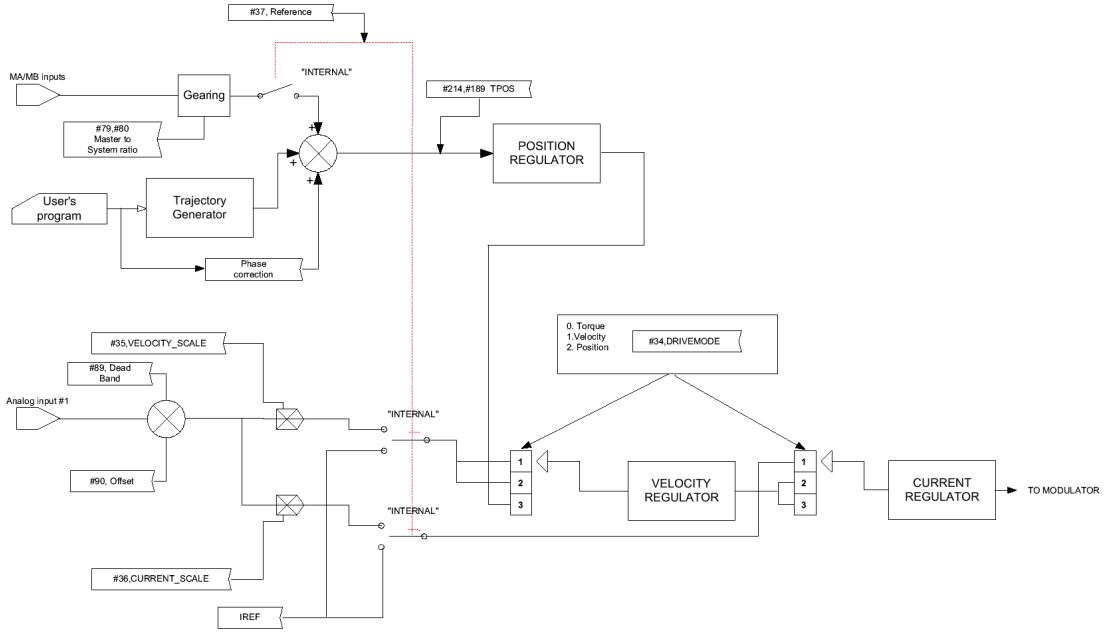


Диаграмма схем адресации

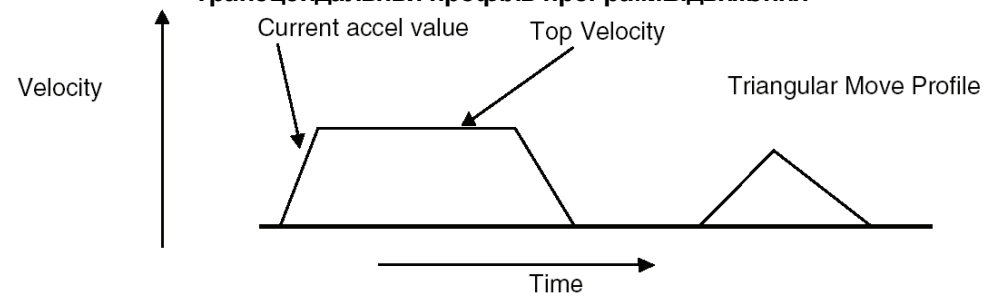
Reference - адресация, MA/MB inputs – входы MA/MB, Master to System ratio – коэффициент передачи Master — система, Gearing – передача, INTERNAL – внутренний, POSITION REGULATOR – регулятор положения, User's program – пользовательская программа, Trajectory Generator – генератор траектории, Phase correction – фазовая коррекция, Torque – вращающий момент, Velocity – скорость, Position - положение, DRIVEMODE - режим привода, VELOCITY_SCALE - шкала скорости, DeadBand - мертвая зона, Analog input #1 – аналоговый вход #1, Offset – смещение, CURRENT_SCALE - токовая шкала, VELOCITY REGULATOR – регулятор скорости, CURRENT REGULATOR – регулятор тока, TO MODULATOR – к модулятору

Режим движения от точки к точке

940 поддерживает два типа перемещения, абсолютное и перемещение приращениями. Для осуществления абсолютного перемещения используется предложение MOVEP (Перейти в положение). При выполнении абсолютного перемещения двигатель получает инструкции перемещаться в известное положение. Перемещение в это известное положение всегда определяется положением двигателя, в пределах собственной сети или нулевым. Например, предложение (MOVEP 0), вызовет перемещение двигателя к нулевой, позиции, (или позиции в пределах собственной сети), независимо от положения двигателя в начале перемещения. Предложение MOVED (Расстояние перемещения) осуществляет перемещения приращениями (или относительное движение) от своего фактического положения. Например, (MOVED 10)

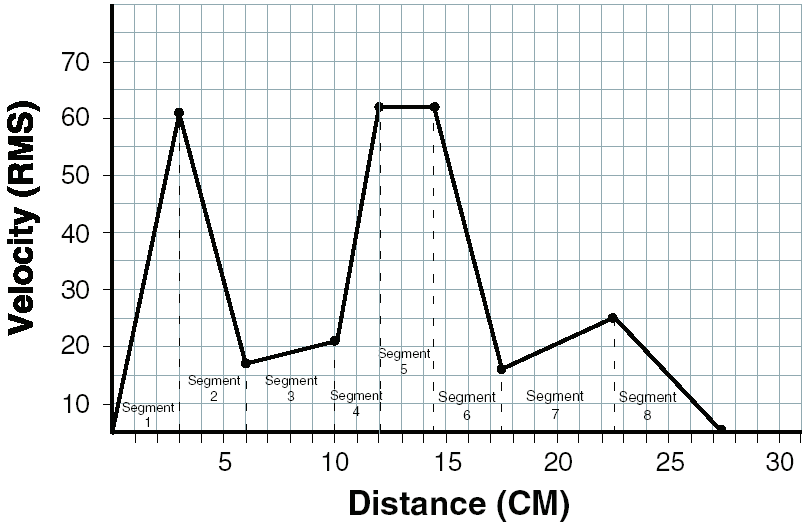
вызовет индексирование двигателем 10 пользовательских единиц впереди своего фактического положения.
Команды MOVEP и MOVED генерируют так называемый трапецеидальный профиль программы движения. При трапецеидальном движении двигатель ускоряется с заданным ускорением (ACCEL) до заданной предельной скорости (MAXV), во время торможения перед конечным положением использует настройки торможения (DECEL). Если расстояние, которое необходимо пройти, довольно мало, используется треугольный профиль программы движения. Движение по треугольнику – это ускорение, и торможение перед непосредственным достижением максимальной скорости, с тем, чтобы достичь желаемого конечного положения.

Трапецеидальный профиль программыдвижения



Velocity – скорость, Current accel value – текущее ускорение, Top Velocity – максимальная скорость
Triangular Move Profile – треугольный профиль программы движения, Time – время

Сегментное движение
Команды MOVED и MOVEP просты и удобны, но если требуемый профиль программы движения сложнее простого трапецеидального движения, тогда используется сегментное движение или MDV. Приведенный ниже профиль разделен на 8 сегментов или 8 MDV шагов. MDV шаг, (Движение Расстояние Скорость (Move Distance Velocity)) имеет два аргумента. Первым аргументом является расстояние, пройденное в том сегменте. Это расстояние определяется от фактического положения двигателя и есть в единицах пользователя. Вторым аргументом является желаемая целевая скорость на конец сегментного движения. То есть скорость двигателя на момент достижения им «положения».



Distance (CM) – расстояние (см), Velocity (RMS) – скорость (RMS)

Номер сегмента	Расстояние, пройденное за один сегмент	Скорость в конце сегмента
1	3	56
2	3	12
3	4	16
4	2	57
5	2,5	57
6	3	11
7	5	20
8	5	0
-	-	-

Ниже приведена пользовательская программа в описанному выше примеру. Пожалуйста, обратите внимание, что конечная скорость последнего сегментного шага должна равняться “0” (MDV 5 , 0). Если не придерживаться этого, последует неисправность F_24 (Ошибка очередности движения (Motion Queue Underflow)).

```
;Сегментное движение
LOOP:
WAIT UNTIL IN_A4==0    ;Подождать, пока вход A4 выключится перед тем, как начать движение
MDV 3 , 56             ;Перейти на 3 единицы с ускорением до 56 пользовательских единиц в сек.
MDV 3 , 12             ;Перейти на 3 единицы с замедлением до 12 пользовательских единиц в сек.
MDV 4 , 16             ;Перейти на 4 единицы с ускорением до 16 пользовательских единиц в сек.
MDV 2 , 57             ;Перейти на 2 единицы с ускорением до 57 пользовательских единиц в сек.
MDV 2.5 , 57           ;Перейти на 2,5 единицы, достигая 57 пользовательских единиц в сек.
MDV 3 , 11             ;Перейти на 3 единицы с замедлением до 11 пользовательских единиц в сек.
MDV 5 , 20             ;Перейти на 5 единиц с ускорением до 20 пользовательских единиц в сек.
MDV 5 , 0              ;Перейти на 5 единиц с замедлением до 0 пользовательских единиц в сек.
WAIT UNTIL IN_A4==1    ;Подождать, пока вход A4 включится перед циклом
GOTO LOOP
END
```

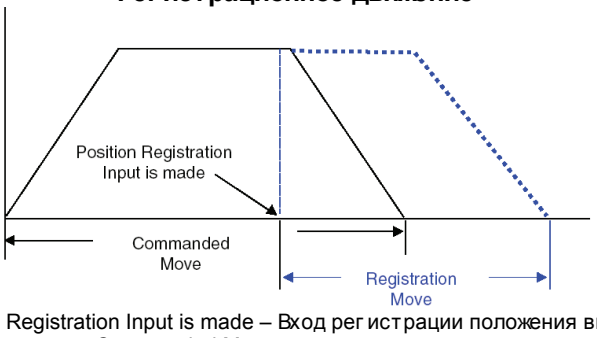
Примечание

- При выполнении MDV движения сегментные шаги сохраняются в очередности движения (Motion Queue). Если программа зацикливается, очередность заполняется и возникает неисправность F_23 (Ошибка переполнения очередности движения (Motion Queue Overflow)).
- Так как MDV движение использует очередность движения «скачок в» (Step into [ ]) или «перешагнуть на» (Step over [ ]) параметры отладки работать не будут.

Регистрация

Для регистрационного движения можно использовать как абсолютное перемещение так и перемещение приращением. С этим движением ассоциируются команды MOVEPR и MOVEDR. Они обладают двумя аргументами. Первый аргумент устанавливает заданное расстояние перемещения или положение. Второй аргумент точно определяет перемещение, совершаемое после того как становится виден регистрационный вход. Если регистрационное движение является абсолютным (MOVEPR 10,30), тогда второй аргумент “30” просто определит положение, к которому нужно двигаться, как только включится регистрационный вход. Если регистрационное движение является перемещением приращения, (MOVEDR 10,30), тогда вторым аргументом будет расстояние, которое необходимо пройти от точки, с которой виден регистрационный вход.

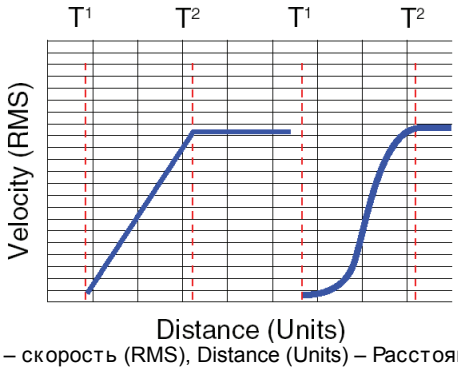
Регистрационное движение



Position Registration Input is made – Вход регистрации положения включен
Commanded Move – заданное движение

S-кривая ускорения

Очень часто для профиля движения важно быть максимально гладким. Это необходимо для того, чтобы снизить износ устройства, или потому, что гладкий профиль наиболее точно позиционирует механизм в конце движения. Для воспроизведения гладких профилей движения, 940 реализует S-кривую ускорения. При прямолинейном профиле ускорения скорость нарастает до заданной по линейному закону. С S-кривой ускорения двигатель вначале ускоряется медленно, в средней прямолинейной зоне в два раза быстрее и плавно прекращает ускорение, достигнув заданной скорости. При постоянном ускорении, изменения скорости резкие в начале ускорения, а также, при достижении заданной скорости. С S-кривой ускорение постепенно нарастает до максимального значения, затем постепенно снижается до нуля. Недостатком при S-кривой ускорения является то, при одном и том же перемещении, максимальное ускорение здесь будет в два раза выше ускорения по прямой линии, что часто требует в два раза большего максимального момента вращения. При этом заданное положение достигается за одно и то же время, независимо от типа используемого способа ускорения.



Velocity (RMS) – скорость (RMS), Distance (Units) – Расстояние (единицы)

Для использования S-кривой ускорения командами MOVED, MOVEP или MDV требуется только дополнительная “,S” в конце предложения.

Примеры

```
MOVED  10 , S
MOVEP   10 , S
MDV 10,20,S
MDV 10,0,S
```

Очередность движения

Привод 940 выполняет пользовательскую программу пошагово. Когда команда (MOVED или MOVEP) выполнена, профиль программы движения сохраняется в истории движения (Motion Queue). Программа по умолчанию остановится на команде до тех пор, пока не выполнится текущая команда. Как только шаг завершен, программа будет выполнять следующую команду. Это действенно будет приостанавливать программу до завершения движения. За стандартным шагом (MOVED или MOVEP) следует один аргумент. Этот аргумент посылает двигателю расстояние или положение, к которому нужно идти. Добавлением второго аргумента “C”, (MOVEP 0,C) или (MOVED 100,C), привод получает разрешение CONTINUE (продолжить) выполнение пользовательской программы на шаг. В этой точке профили программы множественного движения можно сохранить в очередности. Очередность движения может вмещать до 32 профилей. Наподобие команды EVENT, аргумент продолжения “C” очень удобен, когда необходимо запустить действие (обработать I/O), в то время как двигатель находится в движении. Ниже программа образец «Pick and Place» модифицирована для использования аргумента продолжения “C”.

```
;*************************** Главная программа ******************************
PROGRAM_START:
ENABLE
WAIT UNTIL IN_A4==0     ;Перед запуском программы убедитесь, что манипулятор отведен назад
MOVEP 0                 ;Перейти в положение 0, чтобы взять деталь
OUT1 = 1                ;Включить выход 1, чтобы вытянуть манипулятор
WAIT UNTIL IN_A4 == 1   ;Проверить вход и убедиться, что манипулятор вытянут
OUT2 = 1                ;Включить выход 2 для активации захвата
WAIT TIME 1000          ;Задержка в 1 сек., чтобы взять деталь
OUT1 = 0                ;Выключить выход 1, чтобы отвести назад манипулятор
WAIT UNTIL IN_A4==0     ;Проверить вход и убедиться, что манипулятор отведен назад
MOVED 100,C             ;Перейти в «положение места» и продолжить выполнение кода
WAIT UNTIL APOS >25     ;Подождать, пока положение будет больше 25
OUT3 = 1                ;Включить выход 3 для пульверизации детали
WAIT UNTIL APOS >=75    ;Подождать, пока положение не будет больше или равно 75
OUT3 = 0                ;Выключить выход 3, для деактивации пульверизаторов
WAIT UNTIL APOS >=95    ;Подождать до завершения шага перед тем как вытянуть манипулятор
OUT1 = 1                ;Включить выход 1, чтобы вытянуть манипулятор
WAIT UNTIL IN_A4 == 1   ;Проверить вход и убедиться, что манипулятор вытянут
OUT2 =0                 ;Выключить выход 1, чтобы раскрыть захват
WAIT TIME 1000          ;Задержка в 1 сек. чтобы положить деталь на место
OUT1 = 0                ;Отвести назад манипулятор
WAIT UNTIL IN_A4 == 0   ;Проверить вход и убедиться, что манипулятор отведен назад
GOTO PROGRAM_START
END
```

При добавлении аргумента “C” стандартным командам MOVED и MOVEP, генерируемый профиль программы движения рассматривается как MDV движение. При использовании MDV движения выполнение программы не приостанавливается. Генерируемые профили программы движения сохраняются непосредственно в очередности движения и затем один за одним выполняются. Если в предложениях MOVED и MOVEP отсутствует “C” модификатор, в таком случае профили программы движения, генерируемые этими предложениями, поступают в очередь движения и программа приостанавливается, пока не выполнится каждый профиль.

1.12 Подпрограммы и циклы

Подпрограммы
Часто возникает необходимость в повторении последовательности шагов в нескольких местах программы. В таких случаях могут пригодиться подпрограммы. Синтаксис подпрограммы прост. Подпрограммы должны располагаться после главной программы (после END) и должны начинаться дополнительным именем: меткой (где дополнительное имя – это имя подпрограммы), а заканчиваться должны предложением RETURN(возврат).
Обратите внимание, что в подпрограмме могут встречаться более одного предложения RETURN. Подпрограммы вызываются при использовании предложения GOSUB.

Циклы
Язык SML поддерживает блок-команду WHILE/ENDWHILE, которую можно использовать для создания циклов повторения. Обратите внимание на то, что для создания циклов можно также использовать предложения IF-GOTO.
Следующий пример иллюстрирует вызов подпрограммы, равно как и внедрение цикла с помощью команд WHILE / ENDWHILE.

```
;********************** Инициализировать и задать переменные ******************
UNITS = 1
ACCEL = 15
DECEL = 15
MAXV = 100
APOS = 0
DEFINE LOOPCOUNT  V1
DEFINE LOOPS 10
DEFINE DIST V2
DEFINE REPETITIONS V3
REPETITIONS = 0
;************************** Главная программа *******************************
PROGRAM_START:
ENABLE
MAINLOOP:
LOOPCOUNT=LOOPS ;Установить счет циклов, чтобы выполнить 10 циклов
DIST=10                          ;Задать расстояние = 10
WHILE LOOPCOUNT   ;Выполнять циклы при счете циклов больше 0
DIST=DIST/2                  ;увеличить расстояние в ½
GOSUB   MDS                ;Обратиться к подпрограмме
WAIT TIME  100             ;Задержка выполняется после возврата из подпрограммы
LOOPCOUNT=LOOPCOUNT-1  ;уменьшить значение счетчика циклов
ENDWHILE
REPETITIONS=REPETITIONS+1  ;внешний цикл
IF REPETITIONS < 5
GOTO MAINLOOP
ENDIF
END
;***************************** Подпрограммы *********************************
MDS:
V4=dist/3
MDV V4,10
MDV V4,10
MDV V4,0
RETURN
```

2. Программирование

2.1 Введение

Одним из наиболее важных аспектов программирования является разработка структуры программы. Прежде чем начать писать программу, вам необходимо разработать план для этой программы. Какие задачи необходимо выполнить? В каком порядке их необходимо выполнять? Что нужно сделать, чтобы сделать программу легкой для восприятия, и ее обслуживали другие? Имеют ли место операции, которые повторяются?

Большинство программ представляют собой не просто линейный перечень инструкций, в котором каждая инструкция выполняется в одном и том же точном порядке каждый раз при прогоне программы. Программам необходимо выполнять разные вещи в ответ на внешние события и ввод оператора. SML включает в себя инструкции к программной управляющей структуре и функции сканирования события, которые могут быть использованы для контроля за ходом выполнения в прикладной программе.

Инструкции к управляющей структуре – это такие инструкции, под влиянием которых программа изменяет путь выполнения. Сканирование события является инструкцией, выполняемой одновременно с основной частью прикладной программы.

Структура программы

Заглавие – Ввести описание программы и данные названия

```
;****************************** ЗАГЛАВИЕ *********************************
;Название:     программа пример «Pick and Place»
;Автор:        Lenze / AC Technology
;Описание:     Это программа образец, показывающая простой алгоритм, который берет деталь,
               передвигает в заданное место и опускает ее.
```

Перечень I/O – Определить, какие I/O будут использоваться

```
;********************************* Перечень I/O  ******************************
; Вход A1 - не используется
; Вход A2 - не используется
; Вход A3 - Вход включен
; Вход A4 - не используется
; Вход B1 - не используется
; Вход B2 - не используется
; Вход B3 - не используется
; Вход B4 - не используется
; Вход C1 - не используется
; Вход C2 - не используется
; Вход C3 - не используется
; Вход C4 - не используется
;
; Выход 1 - манипулятор
; Выход 2 - захват
; Выход 3 - не используется
; Выход 4 - не используется
```

Инициализировать и задать переменнье – Определить и назначить значения переменных

```
;************************ Инициализировать и задать переменные *****************
UNITS = 1
ACCEL = 75
DECEL =75
MAXV = 10
;V1 =
;V2 =
DEFINE Выход включен 1
DEFINE Выход выключен 0
```

События – Определить название события, Запуск и программу

```
;***************************** События ************************************
EVENT SPRAY_GUNS_ON   APOS > V1   ;Запуск события произойдет при прохождении
                                  ;отметки 25 в положительном направлении.
OUT3= Output_On                   ;Включить распылители (выход 3 включен)
ENDEVENT                          ; Окончание события
EVENT SPRAY_GUNS_OFF  APOS > V2   ;Запуск события произойдет при прохождении положением
                                  ;отметки 75 в отрицательном направлении.
OUT3= Output_Off                  ;Выключить распылители (выход 3 выключен)
ENDEVENT                          ; Окончание события
```

Главная программа – Определить движение и управление устройства

```
;***************************** Главная программа *****************************
PROGRAM_START:
ENABLE
EVENT SPRAY_GUNS_ON  ON    ;Активировать событие
EVENT SPRAY_GUNS_OFF ON    ;Активировать событие
WAIT UNTIL IN_A4==0        ;Перед запуском программы убедиться, что манипулятор отведен назад
MOVEP 0                    ;Перейти в положение 0, чтобы взять деталь
OUT1 = Output_On           ;Включить выход 1, чтобы вытянуть манипулятор
WAIT UNTIL IN_A4 == 1      ;Проверить выход и убедиться, что манипулятор вытянут
OUT2 = Output_On           ;Включить выход 2 для активации захвата
WAIT TIME 1000             ;Задержка в 1 сек., чтобы взять деталь
OUT1 = Output_Off          ;Выключить выход 1, чтобы отвести назад манипулятор
WAIT UNTIL IN_A4==0        ;Проверить вход и убедиться, что манипулятор отведен назад
MOVED 100                  ;Перейти в «положение места»
OUT1 = Output_On           ;Включить выход 1, чтобы вытянуть манипулятор
WAIT UNTIL IN_A4 == 1      ;Проверить выход и убедиться, что манипулятор вытянут
OUT2 = Output_Off          ;Выключить выход 1 для деактивации захвата
WAIT TIME 1000             ;Задержка в 1 сек., чтобы положить деталь на место
OUT1 = Output_Off          ;Отвести назад манипулятор
WAIT UNTIL IN_A4 == 0      ;Проверить вход и убедиться, что манипулятор вытянут
GOTO PROGRAM_START
END
```

Подпрограмма – Здесь должны храниться все без исключения коды подпрограмм

```
;************************* Подпрограммы *********************************
; Введите здесь код подпрограммы
```

Устранитель неисправностей – действие программа при обнаружении неисправности

```
;*********************** Программа устранителя неисправностей ******************
; Введите здесь код устранителя неисправностей
ON FAULT
ENDFAULT
```

Раздел заглавия программы содержит описательную информацию, название программы, номер редакции, описание хода работы и имя программиста.

Раздел перечня I/O программы содержит перечисление всех входов/выходов привода.

Раздел программы **«Инициализировать и задать переменные»** определяет названия для переменных и констант пользователя, используемых в программе.

Раздел События содержит все сканирования событий. Помните, что выполнить предложение **EVENT <eventname> ON** в главной программе означает активировать события. Пожалуйста, обратите внимание, что не все предложения SML выполнимы изнутри тела EVENT (события). Для более подробной информации см. “EVENT” и “ENDEVENT”, в разделе 3 данного руководства (Языковые таблицы (LANGUAGE REFERENCE)). Команда GOTO является одним из предложений, которые нельзя выполнить из тела события. Однако команда JUMP может быть использовано для скачка к коду в теле главной программы. Эта техника позволяет ходу программы изменяться с опорой на выполнение события. Для более подробной информации см. “JUMP” в разделе 3 данного руководства, (Языковые таблицы (LANGUAGE REFERENCE)).

Тело **главной программы** включает в себя основную часть программы, которая может включать все команды движения и математические команды, метки, команды входам/выходам и вызовы подпрограмм. Главное тело должно заканчиваться предложением END.

Подпрограммы – это программы, которые вызываются из главного тела программы. Когда вызывается подпрограмма, (GOSUB), выполнение программы переносится из главной программы в вызванную подпрограмму. Далее подпрограмма будет работать до появления команды RETURN. Выполнив команду RETURN, выполнение программы вернется назад в главную программу к строке кода, следующей за командой GOSUB.

Исправитель неисправностей представляет собой раздел кода, выполняемый при обнаружении приводом неисправности. Этот раздел кода начинается командой “ON FAULT” и заканчивается “ENDFAULT”. При появлении неисправности прерывается нормальный ход программы, движение останавливается, блокируется привод. Прекращается сканирование события, и выполняются команды в устранителе неисправностей до выхода программы из него. Выход из устранителя неисправностей может осуществляться тремя способами:

- Команда “RESUME” закончит выполнение подпрограммы устранителя неисправностей и возобновит выполнение главной программы. Месторасположение, вызванное в команде “RESUME” определит, откуда начнется программа.
- Команда “RESET” закончит выполнение подпрограммы устранителя неисправностей и перезагрузит главную программу до ее первой команды.
- Команда “ENDFAULT” завершит пользовательскую программу.

Имейте в виду, что пока выполняется устранитель неисправностей, события не обрабатываются и поэтому невозможно определить другие ошибки. Поэтому код устранителя неисправностей должен выполняться как можно быстрее. Если для обработки неисправности необходимо написать большой код,

его необходимо поместить в главную программу и использовать для него команду “RESUME”. Обратите внимание, что не все команды SML можно использовать в устранителе неисправностей. Для более подробной информации см. “ON FAULT/ENDFAULT” в разделе 3 данного руководства (Языковые таблицы (LANGUAGE REFERENCE)).
Комментарии допускаются в любом разделе программы, перед ними ставится точка с запятой. Они могут находится на той же строке, что и инструкция или на отдельной. Любой текст после точки с запятой на строке будет игнорироваться компилирующей программой.

2.2 Переменные

Все переменные сервопривода имеют порядковые номера. При помощи этих номеров можно получить доступ к любой переменной из пользовательской программы или из главного интерфейса. Помимо номеров у некоторых переменных есть предопределенные имена, и с помощью этих имен к ним можно получить доступ из пользовательской программы. Для доступа к переменным при помощи порядкового номера используется такой синтаксис:

```
@102 = 20 ; установить переменную #102 в значение 20
@23=@100 ; копировать значение переменной #100 в переменную #23
```

В приводе 940 существует два типа переменных – **Переменные пользователя** и **системные переменные**.
Переменные пользователя представляют собой фиксированный комплект переменных, которые может использовать программист для сохранения данных и осуществления арифметических манипуляций. Все переменные относятся к типу простых (single type), (см. ниже). Возможности переменных простого типа (тип менее изменяемых) освобождают программиста от задачи помнить о необходимости применять правила преобразования между типами, и тем самым значительно упрощать программирование.
Переменные пользователя

V0 – V3 Определенные пользователем переменные. Переменные могут содержать любую цифровую величину, в том числе, и логические (Булев 0 - FALSE и не 0 - TRUE) величины. Их можно использовать в любых действительных арифметических и логических выражениях.

N0 – N31 Определенные пользователем сетевые переменные. Переменные могут содержать любую цифровую величину, в том числе, и логические (Булев 0 - FALSE и не 0 - TRUE) величины. Их можно использовать в любых действительных арифметических и логических выражениях. Переменные могут использоваться сообща по сети Ethernet с использованием команд SEND и SENDTO.

Так как SML представляет собой тип низкоуровневого языка, то не существует особого типа для переменных Булевого типа (переменных, которые могут принимать значения 0 или 1). Регулярные переменные используются для того, чтобы помогать Булевым. Присвоение переменной состояния “FALSE” осуществляется при задании ее равной “0”. Присвоение переменной состояния “TRUE” осуществляется присвоением ей любого значения, отличного от “0”.
Кроме переменных пользователя, поддерживаются также и системные переменные.
Системные переменные являются назначенными переменными, которые обладают определенным значением. Например, переменная **APOS** содержит фактическое положение вала двигателя. Для более подробной информации см. Раздел 2.9.
Область действия
SML переменные доступны по всей протяженности системы. Каждую переменную можно в любое время прочесть и задать из любой пользовательской программы, подпрограммы или командой на базисном языке. Нет никаких мер предосторожности для защиты переменной от изменений. Это так называемая глобальная область действия.
Энергозависимость
Все переменные энергозависимы, т.е. не сохраняют свое значение при выключении питания. После включения питания всем переменным присвоено значение 0. Загрузка или перезагрузка программы не изменяет значения переменных.
Флаги, Разрешение и Точность
Как упоминалось выше, любую переменную можно использовать как флаг в логическом выражении и как условие в условном выражении. Флаги часто используются для индикации осуществления некоторого события, изменения логического состояния входа или выполнения программы до определенной точки. Переменные с отличным от нуля значением рассматриваются как “TRUE” («истина»), а переменные со значением равным “0” рассматриваются как “FALSE” («ложь»). Переменные сохраняются в системе как 4 байта (двойное слово) для целой части и 4 байта (двойное слово) для дробной части. Таким образом, каждая переменная в данной системе сохраняется, как 64 бита в формате 32.32 с фиксированной точкой. Максимальное число может быть представлено в данном формате как +/- 2,147,483,648. Разрешение переменной в этом формате равняется 2.3E-10.

2.3 Арифметические операции

Поддерживаются 4 арифметические функции. Константы, равно как и пользовательские и системные переменные могут быть частью арифметических выражений.

Примеры

```
V1=V1+V2              ;Суммировать две пользовательские переменные
V1=V1-1               ;Вычесть из переменной константу
V2=V1+APOS            ;Суммировать пользовательскую и системную переменные
APOS=20               ;Задать системную переменную
V5=V1*(V2+V3*5/2+1)   ;Сложное выражение
```

Оператор	Символ
Сложение	+
Вычитание	-
Умножение	*
Деление	/

Полученный избыток после операций " * " и " / " вызовет арифметическую ошибку остатка # 19. Полученный избыток/недостаток операций "+" и "-" не вызовет арифметической ошибки.

2.4 Логические выражения и операторы

Побитовые, Булевы операторы и операторы сравнения считаются Логическими операторами. Попросту говоря, это операторы, имеющие дело с логическими величинами объектов. Существуют два возможных значения для логических операндов: TRUE («истина») и FALSE («ложь»). Любое значение, содержащееся в пользовательской переменной, системной переменной или флаге для данных типов операторов рассматривается как TRUE или FALSE. Если значение переменной приравнивается "0", оно считается FALSE. Все остальные значения (отличные от 0), включая отрицательные числа, считаются TRUE.

2.5 Побитовые операторы

Поддерживаются следующие побитовые операторы

Оператор	Символ
AND	&
OR	|
XOR	^
NOT	!

С этими операторами можно использовать как пользовательские, так и системные переменные.

Примеры:

```
V1=V2 & 0xF           ;стереть все биты кроме 4х нижних
IF  (INPUTS & 0x3)    ;проверить входы 0 и 1
V1=V1 | 0xff          ;задать 8 нижних битов
V1=INPUTS ^ 0xF       ;инвертировать входы 0-3
V1=! IN_A1            ;инвертировать 0
```

2.6 Булевы операторы

Эти операторы используют в логических выражениях.

Оператор	Символ
AND	&&
OR	||
NOT	!

Примеры:

```
IF APOS >2 && APOS <6 || APOS >10 && APOS <20
. {statements if true}
ENDIF
```

Приведенный выше пример проверяет, если APOS (фактическое положение) в одном из двух окон; от 2 до 6 единиц или от 10 до 20 единиц.

Другими словами:

```
Если (APOS больше чем 2 AND (и) меньше чем 6)
OR (или)
Если (APOS больше чем 10 AND (и) меньше чем 20)
THEN (тогда) оценивается, как TRUE («истина»). Или же является FALSE" («ложь»)
```

2.7 Операторы сравнения

Поддерживаются следующие операторы:

Операторы	Символы
Больше	>
Меньше	<
Больше или равно	>=
Равно или меньше	=<

Не равно	<>
Равно	= =

Примеры:

```
IF APOS <= 10
IF APOS > 20
IF APOS == 5
IF V1 < 2  &&  V2 <> 4
```

2.8 Системные переменные и флаги

Системными переменными являются те переменные, которые имеют предопределенное значение. Они дают пользователю доступ к определенным параметрам привода. Большинству этих переменных может быть задано значение из «MotionView». В большинстве случаев значение этих переменных можно прочесть и задать в вашей программе. Переменные можно только прочесть, только записать или считать и записать. Только считывающиеся переменные можно только прочесть и нельзя задать. Например, INPUT = 5, является запрещенным действием, потому что вам нельзя задавать значение входа.

Системные флаги являются системными переменными, которые могут иметь только значение 0 или 1. Например, IN_A1 является системным флагом, отображающим состояние цифрового входа 1. Так как входы могут быть только ON (включенными) или OFF (выключенными), значит и значение IN_A1 может равняться только 0 или 1.

2.9 Структура запоминающего устройства системных переменных

Все системные переменные находятся в RAM памяти привода и поэтому являются энергозависимыми. Однако значения некоторых из этих системных переменных хранятся также в EPM. Когда системная переменная изменяется из «MotionView», ее значение изменяется и в RAM, и в EPM. Когда системная переменная изменяется из пользовательской программы, ее значение изменяется только в RAM.

Главный интерфейс обладает функцией изменения значения и в EPM и в памяти, так что пользователь может выбирать, изменить ему значение переменной и в RAM, и в EPM или только в RAM.

2.10 Резюме системных переменных и флагов

Полный перечень системных переменных приведен в Приложении “A”. Каждый аспект 940 можно контролировать манипуляцией значений, хранящихся в Системных Переменных. Все системные переменные начинаются с “VAR_” , к которому прибавляется имя переменной. В качестве альтернативы системным переменным может быть назначен адрес @NUMBER, где NUMBER это индекс переменной. Наиболее часто используемые переменные также имеют альтернативные имена, приведенные ниже. Переменные могут относиться к типам «Только считываемые» Read-Only (R) или «Считывание/Запись» Read/Write (R/W). Системные флаги относятся к только считываемым Read Only (R).

Флагам не присваивают порядковых номеров. Они являются продуктом побитовой маски, примененной приводом к определенной системной переменной и доступной пользователю только в пользовательской программе.

Индекс	Переменная	Тип доступа	Описание переменной	Единицы
186	UNITS	R/W	Шкала единиц пользователя(1)	Единиц пользователя/ Rev.(об.)
215	APOS	R/W	Фактическое положение двигателя	Единицы пользователя
214	TPOS	R	Теоретическое/заданное положение	Единицы пользователя
217	TV	R	Заданная скорость	Единицы пользователя/сек
213	RPOS	R	Положение регистрации. Действительно при установленном системном флаге F_REGISTRATION	Единицы пользователя
218	TA	R	Заданное ускорение	
184	INPOSLIM	R/W	Максимальное отклонение положения для флага INPOSITION, чтобы оставаться заданным	Единицы пользователя
180	MAXV	R/W	Максимальная скорость для команд движения	Единицы пользователя/сек
181	ACCEL	R/W	Ускорение для команд движения	Единицы пользователя /сек^2
182	DECEL	R/W	Торможение для команд движения	Единицы пользователя /сек^2
Индекс	**Переменная**	**Тип доступа**	**Описание переменной**	**Единицы**
183	QDECEL		Быстрое торможение для предложения STOP MOTION QUICK	Единицы пользователя /сек^2
185	VEL	R/W	Задание скорости в режиме скорости	Единицы пользователя/сек
46	PGAIN_P	R/W	Позиционный коэффициент обратной связи (P-gain)	-
47	PGAIN_I	R/W	Позиционный коэффициент обратной связи (I-gain)	-
48	PGAIN_D	R/W	Позиционный коэффициент обратной связи (D-gain)	-
	PGAIN_VFF	R/W	Позиционный коэффициент обратной связи VFF (прямой связи по скорости)	-
49	PGAIN_ILIM	R/W	Предел позиционного I-коэффициента обратной связи	-
44	VGAIN_P	R/W	P-коэффициент обратной связи для скорости	-

45	VGAIN_I	R/W	I-коэффициент обратной связи для скорости	-
65	INPUTS	R	Состояния цифровых входов. Первые 12 битов соответствуют 12 входам привода.	-
66	OUTPUTS	R/W	Цифровые выходы. Первые 5 битов представляют выходы от #0 до #4	-
	INDEX	R/W	Используются нижние 8 битов. Подробную информацию см. в предложении ASSIGN	-
188	PHCUR	R	Фазный ток двигателя	A(мпер)
54	DSTATUS	R	Регистр флагов состояния	-
	DFAULTS	R	Регистр кода неисправности	-
71	AIN	R	Аналоговый вход. Измеряется в вольтах. Диапазон от -10 до +10	V(ольт)
88	AOUT	R/W	Величина аналогового выхода в вольтах. Действительный диапазон от -10 до +10 (V)[2]	V(ольт)

[1] Если переменной “UNITS” присваивается значение “0”, (ноль), тогда “USER UNITS” задается к QUAD ENCODER COUNTS. Эта настройка осуществляется по умолчанию в начале программы перед выполнением UNITS=<value>.
[2] Любое значение вне диапазона +/- 10 назначенное к AOUT будет автоматически подогнано под этот диапазон

Любое значение вне диапазона +/- 10, назначенное к AOUT будет автоматически подогнано под этот диапазон.
Пример:

```
AOUT=100 , AOUT будет назначено значение 10.
V0=236
VOUT=V0, VOUT будет назначено 10 а V0 останется неизменной.
```

Системные флаги

IN_A1-4, IN_B1-4, IN_C1-4	R	Цифровые входы. TRUE, если вход активен, в противном случае FALSE
OUT1, OUT2, OUT3, OUT4, OUT5	W	Цифровые выходы OUTPUT1 – OUTPUT5
F_ICONTROL OFF	R	Интерфейс управления состоянием (ON/OFF) #27 в регистре DSTATUS
F_IN_POSITION	R	TRUE, когда фактическое положение (APOS) в рамках, заданных INPOSLIM переменной и движение завершено
F_ENABLED	R	Ставится, когда привод заблокирован
F_EVENTS OFF	R	Нерабочее состояние событий (ON/OFF) #30 в регистре DSTATUS
F_MCOMPLETE	R	Ставится, когда движение завершено и в очереди движений отсутствуют команды движения
F_MQUEUE_FULL	R	Очередь движений полная
F_MQUEUE_EMPTY	R	Очередь движений пуста
F_FAULT	R	Ставится при обнаружении ошибки
F_ARITHMETIC_FLT	R	Арифметическая ошибка
F_REGISTRATION	R	Ставится, при обнаружении знака совмещения. Содержание переменной RPOS действительно, если данный флаг активен. Флаг сбрасывается любой регистрационной командой MOVEPR,MOVEDR или командой REGISTRATION ON
F_MSUSPENDED	R	Ставится, если движение приостановлено предложением MOTION SUSPEND

Ниже приведена флаговая логика:

```
If
      TPOS-INPOSLIM < APOS < TPOS+INPOSLIM && F_MCOMPLETE && F_MQUEUE_EMPTY
      F_IN_POSITION = TRUE
Else
      F_IN_POSITION = FALSE
End If
```

Для режимов скорости и электронного редуцирования флаги F_MCOMPLETE и F_MQUEUE_EMPTY игнорируются и присваивается TRUE («истина»).

2.11 Управляющие структуры

Управляющие структуры позволяют контролировать ход выполнения программы. Наибольшая сила и полезность любого языка программирования заключается в его способности вносить изменения в порядок предложений при помощи структур и циклов.

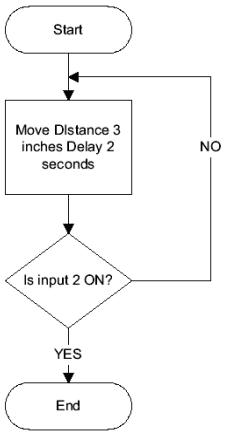
СтруктурыDO/UNTIL

Это предложение используется для выполнения программы поблочно, а затем продолжения выполнения данного блока до тех пор, пока условие станет истинным (удовлетворительным). Разница между предложениями DO/UNTIL и WHILE заключается в том, что команда DO/UNTIL проверяет условие после выполнения блока, таким образом, условные предложения всегда выполняются как минимум один раз. Синтаксис для предложения DO/UNTIL таков:

```
DO
      …команда
UNTIL <условие>
```

Данная блок-схема и кодовый сегмент иллюстрируют использование команд DO/UNTIL.

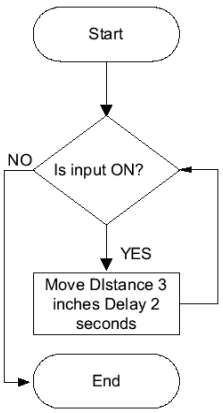
```
… statements

DO
      MOVED 3
      WAIT TIME 2000
UNTIL   IN_A3

… statements
```



Структура WHILE

Эта команда используется, если вы желаете выполнить блок программы при действительном условии. Синтаксис команды WHILE таков:

```
WHILE <условие>

      … команда

ENDWHILE

… команда

WHILE  IN_A3
      MOVED 3
      WAIT TIME 2000
ENDWHILE

… команда
```



Подпрограммы

Подпрограмма представляет собой группу SML команд, расположенных в конце главного тела программы. Она начинается меткой, которую команда GOSUB использует для вызова подпрограммы, и заканчивается коамндой RETURN. Подпрограмма выполняется с использованием предложения GOSUB в главном теле программы. Подпрограммы нельзя вызвать из обработчика событий или ошибок.

При выполнении команды GOSUB выполнение переносится в первую строку подпрограммы. Затем подпрограмма выполняется, пока не дойдет до команды RETURN. Во время выполнения команды RETURN выполнение программы возвращается в программную строку главной программы, следующую за командой GOSUB. В теле подпрограммы могут встречаться больше одной команды RETURN.

Подпрограммы могут быть представленными в форме вложений до 16 единиц. Только последнее тело программы может содержать команду GOSUB. См. Часть 3 Языковые таблицы для более подробной информации о командах GOSUB и RETURN. Данные блок-схема и кодовый сегмент иллюстрируют использование подпрограмм.

```
… команды
```

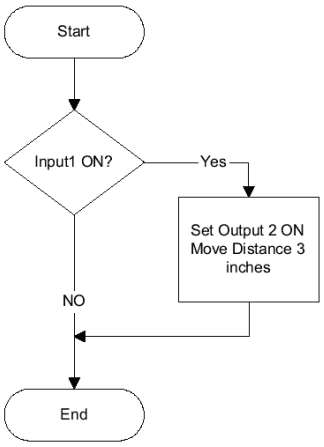
```
GOSUB CalcMotionParam
MOVED V1
OUT2=1
… команды
END
;подпрограммы обычно расположены после команды END главной программы
;CalcMotionParam:
V1 = (V3*2)/V4
RETURN
```

Структура IF

Команда “IF” используется для выполнения команды или блока команд единожды, если условие действительно. Упрощенный синтаксис для IF таков:

```
IF condition    (если выполняется условие)
…statement(s)    (команда)
ENDIF
```

Данные блок-схема и кодовый сегмент иллюстрируют использование команды IF.



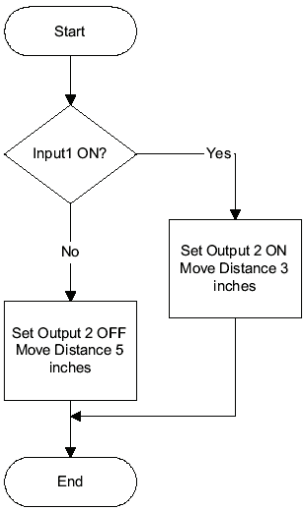

```
… команды

IF IN_A2
      OUT2=1
      MOVED 3
ENDIF

… команды
```

Структура IF/ELSE

Команда IF/ELSE используется для выполнения команды или блока команд единожды, если условие действительно, и для выполнения другой команды или блока команд, если условие недействительно. Упрощенный синтаксис для команды IF/ELSE таков:

```
IF <condition>
 …statement(s)
ELSE
…statement(s)
ENDIF
```

Данные блок-схема и кодовый сегмент иллюстрируют использование команды IF/ELSE.




```
… команды
IF IN_A2
      OUT2=1
      MOVED 3
ELSE
      OUT2=0
      MOVED 5
ENDIF
… команды
```

Структура WAIT

Команда WAIT используется для того, чтобы приостановить выполнение программы до тех пор, пока или в то время когда условие верное. Упрощенный синтаксис для этого предложения таков:

```
WAIT UNTIL  <condition>
WAIT WHILE <condition>
WAIT TIME <time>
WAIT MOTION COMPLETE
```

GOTO/Метка

Предложение GOTO можно использовать для переброса выполнения программы на новую точку, обозначенную меткой. Это предложение часто используется как операция предложения IF. Целевая метка может находиться выше или ниже предложения GOTO в прикладной программе.
Метки могут представлять собой любую начинающуюся буквой буквенно-цифровую строку длиной до 64 символов, за которой ставится двоеточие":".

```
GOTO TestInputs
…statements
TestInputs:
…statements
IF (IN_A1) GOTO TestInputs
```

Резюме команд программной структуры
Данная таблица содержит краткую характеристику команд, имеющих отношение к ветвлению программы.

Команда	Описание
GOTO	Вызывает подпрограмму
DO/UNTIL	Выполняет единократно до тех пор, пока условие не станет верным
IF и IF/ELSE	Выполняют при верном условии
RETURN	Возвращает из подпрограммы
WAIT	Ожидает в течение заданного времени или пока условие не станет верным
WHILE	Выполняет, когда условие верное

2.12 Командысканирования событий

Сканирование событий – это маленькая программа, работающая независимо от главной программы. SCANNED EVENTS очень полезно в тех случаях, когда необходимо запустить операцию (манипулировать I/O), когда двигатель в движении. При установке Событий, первым шагом является определение операции, которая запустит событие, равно как и последовательность команд, которые требуется выполнить сразу же после запуска события. Сканирование событий осуществляется каждые 256µS. Однако чтобы сканировать событие, его необходимо сначала активировать. Активировать и деактивировать события можно из пользовательской программы, из другого события или из этого же (см. объяснения ниже). Как только событие определено и активировано, оно будет сканироваться постоянно, пока не встретит триггерное условие, эта частота сканирования независима от хронометража главных программ. Как только встретится триггерное условие, предложения Event будут выполнены одновременно с пользовательской программой.
Сканирование событий используется для записи событий и выполнения операций независимо от главного тела программы. Например, если вы хотите включить выход 3 при положении больше 4 дюймов, или вам необходимо включить выход 4, несмотря на то, что входы 2 и 3 включены, вы можете использовать эти предложения сканирования событий.

```
EVENT PositionIndicator APOS > 4
      OUT3=1
END EVENT

EVENT  Inputs3and4 IN_A4 & IN_B1
      OUT4=1
ENDEVENT
… команды
```

Сканирование событий может также использоваться с таймером для выполнения операций на периодической основе.
Программные команды, содержащиеся в части действий сканирования событий могут являться командами любой допустимой программы кроме: Вызовов подпрограмм (GOSUB), DO/WHILE, WHILE, WAIT, GOTO а также команд движения: MOVED,MOVEP,MDV,STOP, MOTION SUSPEND/RESUME.

EVENT <name> INPUT <inputname>
Это предложение сканирования события используется для выполнения одного блока программы каждый раз, когда вход специального назначения <inputname> (название входа) изменяет свое состояние с low->hi. Если его необходимо запустить при изменении с hi->low, тогда перед <inputname> необходимо поставить символ восклицательный знак (!) (!IN_A4).

EVENT <name> TIME <timeout>
Эта команда сканирования события используется для выполнения блока программы с частотой повторения определенной аргументом <timeout> (тайм-аут).
EVENT <name> expression

Эта команда сканирования события используется для выполнения блока программы, когда выражение оценивается как верное.

EVENT <name> ON/OFF

Эта команда используется для активации/деактивации сканирования события. Предложение можно использовать внутри программного блока события.

Резюме команд сканирования событий

Данная таблица содержит краткую характеристику команд, имеющих отношение к сканированию событий. Для более подробной информации см. Часть 3 "Языковые таблицы".

Название	Описание
EVENT <name> ON/OFF	активировать / деактивировать событие
EVENT <name> INPUT <inputname>	Сканирование событий на вход <#>
EVENT <name> TIME <value>	Периодическое событие со <значением> частоты повторения.
EVENT <name> expression	Сканирование событий при выражение = верное

2.13 Движение

Обзор перемещений

Команда на изменение положения, которая генерирует движение, поступает из интерполятора (profile generator), или сокращенно профайлера (profiler). Интерполятор используется командами MOVE, MOVED, MOVEP, MOVEPR, MOVEDR и MDV. Команды MOVE (перемещения) генерируют движение в положительном или отрицательном направлениях во время или до тех пор, пока не встретятся определенные условия. Например, можно специфицировать движение, в то время как выделенный вход остается включенным (или выключенным). MOVEP генерирует перемещение в определенное абсолютное положение. MOVED генерирует инкрементное перемещение на расстояние, то есть перемещение на определенное расстояние от фактического положения. MOVEPR и MOVEDR являются регистрационными перемещениями. MDV команды используются для генерации сложных профилей. Профили, генерируемые этими командами, попадают в стек движения, глубина которого измеряется 32 уровнями. По умолчанию, когда одно из этих предложений (кроме MDV) выполняется, выполнение главной пользовательской программы приостанавливается до осуществления сгенерированного движения. Запросы движения, генерируемые предложением MDV или MOVE предложениями с "C" модификатором не приостанавливают программу. Они просто попадают в стек движения и выполняются профайлером в том порядке, в котором были загружены. Стек движения может вмещать до 32 перемещений. Язык SML позволяет программисту загружать перемещения в стек и продолжать работать с программой. В обязанности программиста входит проверка стека движения перед загрузкой новых перемещений, чтобы убедиться, что в нем достаточно места. Это выполняется проверкой соответствующего флага в регистре состояния системы.

Инкрементное (MOVED) и абсолютное (MOVEP) движение

Предложения MOVED и MOVEP используются для создания инкрементных и абсолютных перемещений соответственно. Результатом этих команд является движение, которое по умолчанию представляет собой перемещение с трапецевидной скоростью или скоростью по S-кривой, если с предложением используется S-модификатор,

Например:

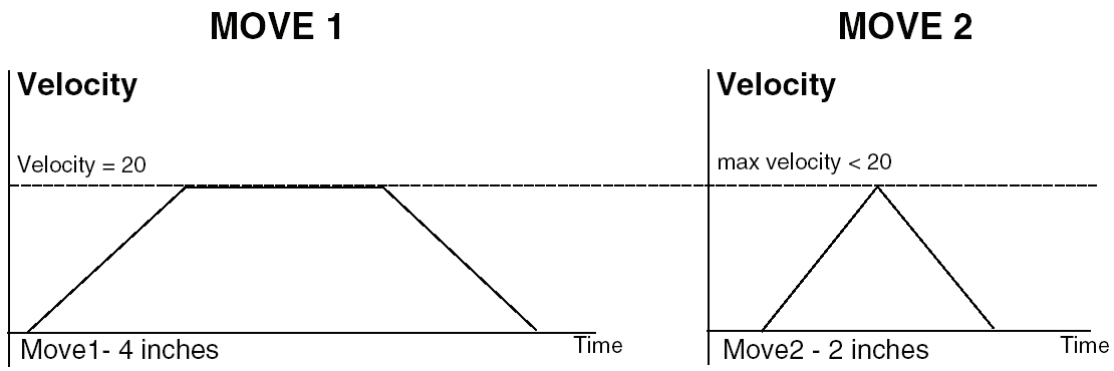
```
MOVEP 10      ;будет иметь результатом трапецевидное перемещение
```

Но

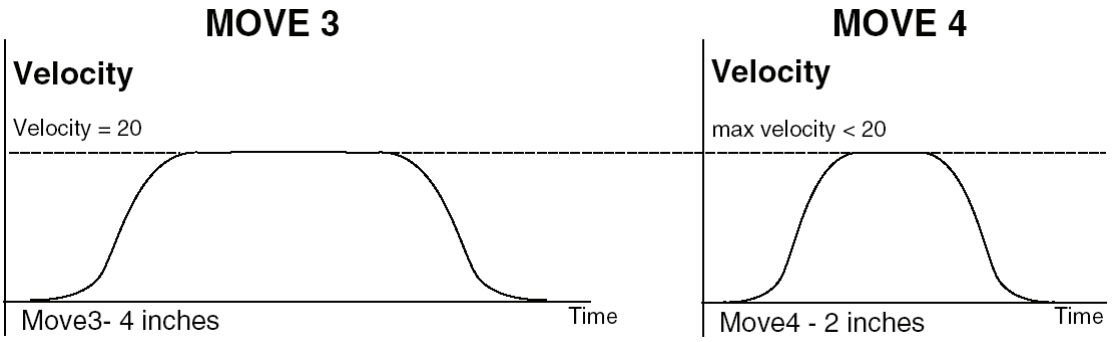
```
MOVEP 10,S    ;будет иметь результатом перемещение по S-кривой
```

В приведенном выше примере, (MOVEP 10), длина перемещения определена аргументом, стоящим после команды MOVEP (10). Этот аргумент может быть числом, переменной или любым действительным арифметическим выражением. Максимальная скорость перемещения определена заданием системной переменной MAXV. Ускорение и торможение определены заданием системных переменных ACCEL и DECEL соответственно.

Если значения для скорости, ускорения и торможения для определенного расстояния таковы, что времени на ускорение до определенной скорости недостаточно, профиль движения будет иметь результатом треугольный или двойной S-профиль. (см. рисунок ниже):



Трапецевиднье перемещения




```
ACCEL = 200
DECEL = 200
MAXV = 20
MOVED 4      ;Move1
MOVED 2      ;Move2
MOVED 4,S    ;Move3
MOVED 2,S    ;Move4
```

У всех четырех приведенных выше перемещений одинаковые значения ускорения, торможения и максимальной скорости. Значение расстояния для перемещений 1 и 3 больше, нежели для перемещений 2 и 4. В перемещениях 1 и 3 длина расстояния достаточна для того, чтобы двигатель развил ускорение до профилированной максимальной скорости, и достиг этой скорости до торможения и остановки. В перемещениях 2 и 4 расстояние так мало, что в то время как двигатель работает с ускорением по направлению к профилированной максимальной скорости, ему уже необходимо тормозить и останавливаться прежде, чем он достигнет максимальной скорости.

Инкрементное (MOVED) перемещение
Инкрементное перемещение определяется как перемещение на некоторое расстояние из фактического положения. Переместиться на четыре оборота от фактического положения – вот пример инкрементного движения.
MOVED является командой, которая используется для создания инкрементных перемещений. Упрощенный синтаксис таков:
MOVED <+/-расстояние>
+/- знак передаст валу двигателя направление вращения.

Абсолютное (MOVEP) перемещение
Абсолютное движение определяется как движение из текущего положения к некоторому заданному положению. Заданное положение определяется как положение относительно заданной нулевой точки. Нулевая точка для системы устанавливается во время цикла возврата к исходному положению, выполняемому обычно сразу же при включении питания.
Во время цикла возврата к исходному положению, двигатель будет совершать инкрементные перемещения, вместе с тем следуя физическому вводу, индексу, или обоим.

Регистрационные (MOVEDR MOVEPR) перемещения

MOVEPR и MOVEDR используются для перемещения в положение или на расстояние соответственно как MOVEP и MOVED. Разница в том, что во время выполнения предложений, они ищут регистрационный сигнал или регистрационный вход. Если во время движения обнаружен регистрационный сигнал, генерируется новое целевое положение. При перемещении MOVEDR привод прирастит расстояние обнаруженное в регистрационной команде. Это приращение за начало отсчета возьмет положение, с которого был виден регистрационный вход. При перемещении MOVEPR новое положение будет абсолютно тем же положением, которое было указано в регистрационной команде.
Пример:

```
MOVEDR  5, 1    ;команда совершает перемещение на расстояние в 5 пользовательских
                ;единиц или регистрационное положение + 1 пользовательская единица,
                ;если регистрационный вход активируется во время движения.
```

Существуют два исключения из этого режима работы.

Исключение первое:

Перемещение не будет модифицировано на “Положение регистрации + смещение”, если регистрация будет обнаружена в момент торможения системы на пути к завершению движения.

Исключение второе:

Как только регистрационный вход становится видим, должно быть достаточно места для того, чтобы двигатель замедлил ход до остановки, используя профилированную величину Decel. Если новое регистрационное перемещение больше расстояния, необходимого для остановки, тогда двигатель перерегулирует положение новой регистрации.

Сегментные перемещения

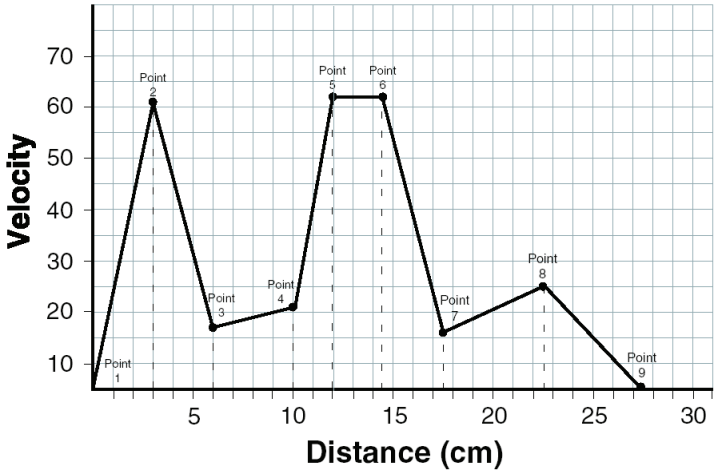
Помимо простых перемещений, генерируемых предложениями MOVED и MOVEP можно генерировать сложные профили с использованием сегментных перемещений. Сегментное перемещение представляет собой часть сложного перемещения. Сложное перемещение состоит из двух и более сегментов, начинающихся и заканчивающихся с нулевой скоростью.

MDV сегменты

Сегменты создаются с использованием ряда MDV команд. Упрощенный синтаксис для MDV (Перемещение на расстояние со скоростью) команды таков:

MDV <расстояние>,<скорость>

<Расстояние> - это длина перемещения сегмента. <Скорость> - это конечная скорость для перемещения сегмента. Начальная скорость равна нулю или конечной скорости предыдущего сегмента. Последний сегмент в завершенном перемещении должен иметь нулевую скорость. Если конечная скорость конечного сегмента отличается от 0, появится ошибка стека движения в работе. Представленный ниже профиль можно разбить на 8 MDV перемещений. Первый сегмент определит расстояние между точкой 1 и точкой 2, а также скорость в точке 2. Итак, при расстоянии между точкой 1 и 2 равном 3 единицам, а также скоростью в точке 2 равно 56 RPM, команда будет следующей: MDV 3 , 56. Второй сегмент предоставит расстояние между точками 2 и 3, а также скоростью в точке 3, и так далее. Любой профиль можно запрограммировать, используя MDV перемещения.



Velocity – скорость, Distance – расстояние

Данная таблица представляет приведенный выше график.

Номер сегмента	Перемещение за один сегмент	Скорость в конце сегмента
1	3	56
2	3	12
3	4	16
4	2	57
5	2,5	57
6	3	11
7	5	20
8	5	0
-	-	-

```
;перемещения сегментов
MDV  3 , 56
MDV  3 , 12
MDV  4 , 16
MDV  2 , 57
MDV  2.5 , 57
MDV  3 , 11
MDV  5, 20
MDV  5 , 0
END
```

Для расчета ускорения, вытекающего из движения сегмента, можно использовать нижеследующее уравнение.

Accel = (V2f - V20) / 2*D

Vf – Конечная скорость

V0 – Начальная скорость

D – Расстояние

Ускорение по S-кривой
Вместо того, чтобы использовать линейное ускорение, движение, созданное с использованием сегментных перемещений (MDV предложений), может использовать ускорение по S-кривой. Синтаксис для MDV перемещения с ускорением по S-кривой таков:
MDV <расстояние>,<скорость>,S
Сегментные перемещения с использованием ускорения по S-кривой займут столько же времени, сколько и с линейным ускорением. Ускорение по S-кривой полезно, так как является более гладким в начале и в конце сегмента, однако, максимальная величина ускорения сегмента будет вдвое выше величины в сегменте с линейным ускорением.

Движение SUSPEND/RESUME.
Временами необходимо осуществлять контроль движения, предварительно нагружая стек движения профилями движения. Затем на основе пользовательской программы, начать выполнять данные профили движения в некоторый предопределенный момент. Команда “MOTION SUSPEND” приостановит движение до выполнения предложения “MOTION RESUME”. Пока движение приостановлено, команда движения, выполняемая пользовательской программой, будет загружена в стек движения. Как только команда “MOTION RESUME” будет выполнена, профили движения с предварительным нагружением будут выполнены в том порядке, в каком их загрузили.
Пример:
```
MOTION SUSPEND
MDV 10,2     ;помещено в стек
MDV 20,2     ;помещено в стек
MDV 2,0      ;помещено в стек
MOVED 3,C    ;нужно использовать ",C "модификатор. Иначе программа зависнет.
```

При использовании команд MOVED,MOVEP и MOVE, необходимо принять меры предосторожности. Если какая-либо из MOVE команд написана без “C” модификатора, программа зависнет, или будет заблокирована. Команда “MOTION SUSPEND” эффективно останавливает все выполнения движения. Если программа выполняет предложения “MDV” и “MOVED” , данные профили перемещения загружаются в стек движения. Если в последнем “MOVED” отсутствует “C” модификатор, тогда пользовательская программа подождет завершения этого профиля перемещения, прежде чем продолжить. Но из-за того, что движение было приостановлено, перемещение не завершится и программа зависнет в этом месте на неограниченный срок.

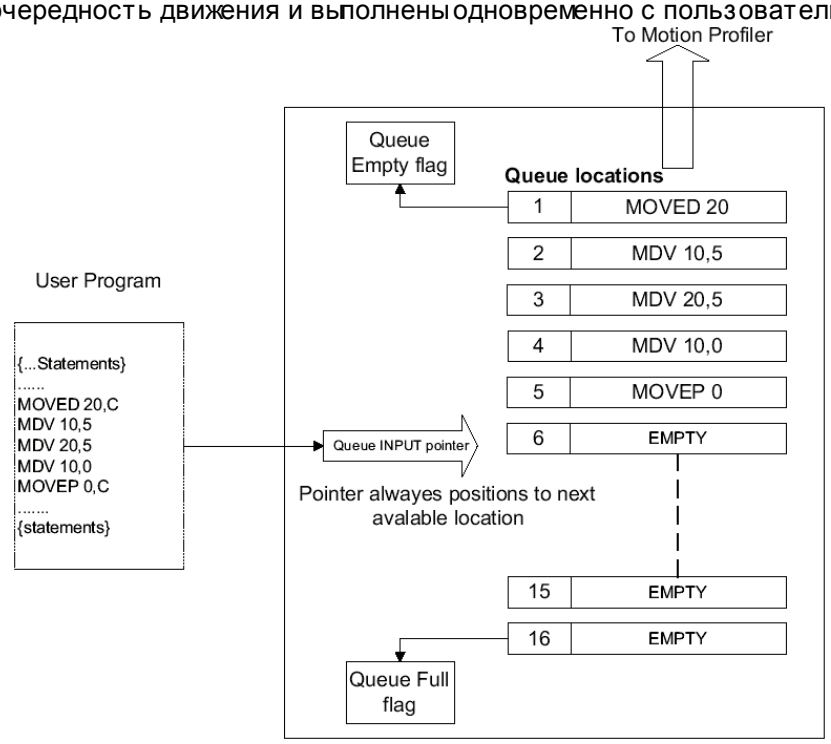
Условные перемещения (MOVE WHILE/UNTIL)
Команды “MOVE UNTIL <выражение>” и “MOVE WHILE <выражение>” начнут свои профили движения на основе профильных настроек ускорения и максимальной скорости. Предложение “MOVE UNTIL <выражение> будет продолжать перемещение до тех пор, пока <выражение> не станет верным. “MOVE WHILE <выражение>” также будет продолжать перемещение до тех пор, пока его <выражение> остается верным. Выражением может быть любое действительное арифметическое или логическое выражение или их комбинация.
Примеры:
```
MOVE WHILE APOS<20          ;Перемещаться, пока положение меньше  20,
                            ;затем остановиться с текущим коэффициентом торможения.
MOVE UNTIL APOS>V1          ;Перемещаться в положительном направлении, пока
                            ;положение не превысит величины переменной
V1MOVE BACK UNTIL APOS<V1   ;Перемещаться в отрицательном направлении, пока
                            ;положение не станет меньше величины переменной
V!MOVE WHILE  IN_A1         ;Перемещаться в положительном направлении, пока
                            ;активирован вход A1.
MOVE WHILE !IN_A1           ;Перемещаться в положительном направлении, пока
                            ;вход A1 не активирован.
                            ;Восклицательный знак (!) перед IN_A1 обращает
                            ;(или инвертирует) величину IN_A1.
```
При помощи этого последнего примера очень удобно находить датчик или выключатель.

Очередность движения и выполнение команд в движении
Когда программа выполняет предложение MOVE, MOVED или MOVEP, она по умолчанию ждет, пока движение завершится, прежде чем перейти к следующему предложению. Это эффективно остановит программу на время, пока не выполнится требуемое движение. Обратите внимание, что, несмотря на это, “EVENTS” не приостанавливаются, а продолжают выполняться параллельно с пользовательской программой. Подобно команде EVENT , аргумент продолжения “C” очень полезен, когда необходимо запустить действие (обработать I/O), пока двигатель пребывает в движении. Ниже приведен пример аргумента продолжения “C”.

```
;Эта программа осуществляет мониторинг I/O параллельно с движением:
START:
      MOVED 100,C       ;начать перемещение максимум со 100 оборотов
WHILE F_MCOMPLETE=0     ;во время перемещения
IF IN_A2 == 1           ;если обнаружен датчик
      OUT1=1            ;включить выход на
      WAIT TIME 500     ;500 мс
      OUT1=0            ;выключить выход
      WAIT TIME 500     ;подождать 500 мс
      ENDIF
ENDWHILE
      MOVED -100        ;вернуться
      WAIT TIME 1000    ;подождать
      GOTO START  ;и начать всю программу
      END
```

Эта программа начинает движение со 100 оборотов. Пока двигатель вращается, вход A2 под наблюдением. Если вход A2 активируется во время движения, выход 1 включается на 500мс и выключается. Программа будет продолжать цикл WHILE, ведя наблюдение за входом A2, пока перемещение не завершится. Если вход 2 останется включенным, или активированным, во время перемещения, тогда выход 1 будет включаться и отключаться каждые 500 мс до завершения перемещения. Если вход A2 активируется только, когда привод пройдет датчик, подключенный ко входу, тогда выход 1 останется включенным только в течение 500 мс. Присоединив аргумент продолжения “C” к MOVE программа сможет осуществлять наблюдение за входом, одновременно выполняя профиль движения. Без этого модификатора программа будет приостановлена до тех пор, пока не выполнится все движение, тем самым, исключая возможность следить за входом во время перемещения. После того, как двигатель прошел все расстояние, далее он возвращается в начальное положение и процесс повторяется. Эту программу можно использовать для простого механизма покраски, который включает краскораспылитель, как только край детали (или ее метка) пересекает датчик(и).

Представленная ниже диаграмма иллюстрирует структуру и режим работы очередности движения. Все перемещения перед выполнением загружаются в Очередность движения. Если перемещение является стандартным “MOVEP 10” или “MOVED 10”, оно будет загружено в очередь и выполнение программы приостановится до тех пор, пока перемещение не завершится. Если перемещение обладает аргументом продолжения “MOVEP 10,C” или “MOVED 10,C”, или является “MDV” перемещением, тогда перемещения будут загружены в очередность движения и выполнены одновременно с пользовательской программой.



To Motion Profiler – к профайлеру движения, Queue Empty flag – флажок пустой очереди, Queue locations – ячейки очереди,
User Program – пользовательская программа, Statements – предложения, Queue INPUT pointer – указатель входа в очередь
Pointer alwayes positions to next avalable location – указатель всегда указывает на ближайшую свободную ячейку
EMPTY – пустая, Queue Full flag – флажок полной очереди

Очередность движения может вместить 32 профиля движения. Регистр состояния системы показывает состояние очередности движения. Если установлен флажок, очередь полная. Если существует возможность переполнения, программист должен проверять этот флажок перед выполнением любых перемещений, особенно в программах, где предложения перемещения выполняются скачкообразно. Попытки выполнить команды движения, когда очередь движения заполнена, приведут к ошибке #23. MDV команды не имеют “C” опции и потому они никогда не приостанавливают программу. Если последняя в очереди MDV команда не задает нулевую скорость движения, появится ошибка опустошения стека #24. “MOTION SUSPEND” и “MOTION RESUME” можно использовать для управления пользовательской программой и очередностью движения. Если профилями движения, загруженными в очередь, управлять неправильно, очередность движения переполнится, и это приведет к ошибке работы привода.

2.14 Регистр состояния системы(DSTATUS регистр)

Регистр состояния системы, (DSTATUS), является только считываемым регистром. Его биты показывают различные состояния подсистем 940. Некоторые из флажков доступны, как переменные системных флагов, их краткая характеристика приведена в таблице ниже:

Бит в регистре	Описание
0	Задается при включении привода
1	Задается при какой-либо ошибке в DSP подсистеме
2	Задается при действительной программе привода
3	Задается при какой-либо ошибке в байткоде или системе, или DSP
4	Задается при действительном исходном коде привода
5	Задается, если движение завершено, а целевое положение – в указанных пределах
6	Задается, когда устройство для наблюдения (scope) запущено и данные собраны
7	Задается при полном стеке движения
8	Задается при пустом стеке движения
9	Задается при останове байткода
10	Задается, когда байткод работает
11	Задается, если байткод настроен для работы в пошаговом режиме
12	Задается, если байткод достиг конца программы
13	Задается, если достигнут текущий предел
14	Задается при какой-либо ошибке байткода
15	Задается, если не выбран действительный двигатель
16	Задается при арифметической ошибке байткода
17	Задается при пользовательской ошибке в байткоде
18	Задается, если завершена инициализация DSP
19	Задается, если была запущена регистрация
20	Задается, если регистрационную переменную обновили из DSP после последнего запуска
21	Задается при ошибке модуля движения
23	Задается, если движение приостановлено
24	Задается, если программа запросила приостановить движение
25	Задается, если система ожидает завершения движения
26	Задается, если команда движения завершена и очередность движения пуста
27	Задается, если байткод запросил перезагрузку
28	Задается, если выключен установленный интерфейс-контроль. Этот флажок ставится/убирается предложением ICONTROL ON/OFF
29	Задается при достижении положительного концевого выключателя
30	Задается при достижении отрицательного концевого выключателя
31	События выключены. Если этот флажок установлен, все события выключены. После выполнения EVENTS ON все события предварительно активированные предложениями EVENT EventName ON снова включаются.

2.15 Кодыошибок (DFAULTS регистр)

Ошибки привода записываются в специальную переменную под названием регистр “DFAULTS” или «Регистр ошибок» (“Fault Register”). В регистре системного статуса также ставятся определенные флажки.

Всякий раз, когда в приводе появляется ошибка, запись о ней посылается в память специальной системной переменной под названием Регистр ошибок (DFAULTS). Вдобавок к этому, будут установлены специальные флажки регистра статуса системы, чтобы помочь обозначить, к какому классу ошибок принадлежит текущая ошибка. Ниже приведена таблица, содержащая возможные коды ошибок. Примечание: Коды с 1 по 16 отведены под ошибки подсистемы DSP. Остальные коды генерируются различными системами 940.

Код ошибки	Соотнесенные флажки в регистре состояния	Описание
1	1,3	Превышение напряжения
2	1,3	Недействительный код датчиков Холла
3	1,3	Превышение тока
4	1,3	Перегрев
5	1,3	Зарезервированный
6	1,3	Превышение скорости. (Превышение скорости, заданной в файле двигателя.)
7	1,3	Превышение ошибки отработки положения
8	1,3	Попытка включения, при неверно введенных данных мотора или не выбран двигатель
9	1,3	Сработан датчик перегрева мотора
10	1,3	Ошибка подпроцессора
11-13	-	Зарезервированный
14	1,3	Недонапряжение
15	1,3	Защита ограничения по току
16	-	Зарезервированный
17	3	Неисправимая ошибка
18	16	Деление на ноль
19	16	Переполнение
20	3	Переполнение стека подпрограммы. Глубина стека подпрограммы > 16 уровней
21	3	Потеря значимости стека подпрограммы. Выполнение RETURN без вызова подпрограммы.
22	3	Переполнение стека оценки переменной. Слишком сложное выражение для обработки компилятором.
23	21	Переполнение очередности движения. Превышение глубины на 32 уровня.
24	21	Недополнение очередности движения. Последняя в очереди MDV команда имела отличную от нуля конечную скорость.
25	3	Неизвестный код операции. Ошибка интерпретатора байткода.
26	3	Неизвестный байткод. Ошибка интерпретатора байткода.
27	21	Привод заблокирован. Попытка выполнить движение, пока привод заблокирован.
28	16,21	Слишком высокое ускорение. Параметры команды движения рассчитывают величину ускорения, превышающую возможности системы.
29	16,21	Слишком низкое ускорение. Параметры предложения движения рассчитывают величину ускорения ниже возможностей системы.
30	16,21	Слишком большая скорость. Параметры предложения движения рассчитывают скорость, превышающую возможности системы.
31	16,21	Слишком низкая скорость. Параметры предложения движения рассчитывают скорость ниже возможностей системы.
32	3,21	Сработал положительный концевой выключатель
33	3,21	Сработал отрицательный концевой выключатель
34	3,21	Попытка совершить положительное движение со сработанным положительным концевым выключателем
35	3,21	Попытка совершить отрицательное движение со сработанным отрицательным концевым выключателем
36	3	Аппаратура заблокирована (вход разблокировки не активен при попытке включить привод из программы или интерфейса)
37	3	Пониженное напряжение
38	3	Потеря EPM
39	3,21	Достигнут положительный порог, установленный программным обеспечением.
40	3,21	Достигнут отрицательный порог, установленный программным обеспечением.
41	3	Попытка использовать переменную с неизвестным ID из пользовательской программы.

2.16 Ограничения и запреты

Запретыпо использованиюкоммуникационных интерфейсов
Параллельное соединение с портами RS232 и RS485 разрешено с целью ретрансляции (перенесения данных) между интерфейсами.
Использование RS232 или RS485 параллельно с Ethernet может привести к непредсказуемому поведению, так как привод попытается выполнять команды из двух интерфейсов одновременно.

Ограничение параметров движения
Вследствие конечной точности расчетов, существуют некоторые ограничения вычислений для ускорения/торможения и максимальной скорости перемещения. Именно благодаря этим ограничениям вы можете получить арифметические ошибки во время выполнения программ. Минимальные/Максимальные величины выражаются в единицах отсчета или единицы отсчета/замеры, где замеры – интервал отсчета позиционного цикла и равен 256uS.

Параметр	MIN	MAX	Единицы
Accel(ускорение) / Decel(замедление)	65/(2^32)	512	counts/sample^2
MaxV (максимальная скорость)	0	2048	counts/sample
Max расстояние перемещения	0	+/- 2^31	counts

Ограничения глубиныстеков и очередностей
Очередность движения 32
Стек подпрограмм 32
Количество событий 32

3. Языковые таблицы

Формат
Каждая команда, системная переменная или операнд документируются с использованием приведенного ниже формата. Если ни в одном из полей нет информации, метка все равно присутствует.

KEYWORD	Длинное имя	Тип
Цель		
Синтаксис	KEYWORD=величина Переменная=KEYWORD Аргументы	
Пометки		
Смотрите также:		
Пример		
Ключевое слово:	KEYWORD – это название команды, системной переменной или системного флажка в том виде, в каком они появляются в программе.	
Long Name: **(длинное имя)**	Длинное имя является интерпретацией ключевого слова. Например: MOVEP – это ключевое слово, а Перемещение в Положение (Move to Position) будет его длинным именем. Длинное имя предоставляется только в качестве помощи читателю и не может использоваться в программе.	
Тип	Это поле определяет, к какому типу предложения или системной переменной относится ключевое слово.	
Назначение	Назначение ключевого слова.	
Синтаксис	В этом поле показано, как правильно использовать ключевое слово. Опционные аргументы будут заключены в квадратные скобки [ ]. Аргументы будут написаны курсивом.	
Аргументы	Данные, обеспечиваемые предложением, и которые изменяют поведение предложения. Например, MOVED=100. MOVED является предложением, а 100 - аргументом.	
Пометки	Поле пометок содержит дополнительную информацию об использовании предложения или системной переменной.	
Смотри также	Данное поле содержит перечень предложений или системных переменных, имеющих отношение к назначению ключевого слова.	
Пример	Поле примера содержит кодовый сегмент, иллюстрирующий использование ключевого слова.	
Ссылка		

ASSIGN	Назначить вход как порядковый бит	Команда
Цель	Назначение ключевого слова мотивирует выделенный вход действовать, как определенный бит индекса (INDEX) системной переменной. После таких назначений изменения в состоянии входа вызовут изменения в определенном бите, который присвоен входу.	
Синтаксис	ASSIGN INPUT <имя входа> AS BIT <bit #>	
Название входа	Имя входа(IN_A1..IN_A2 etc.) Bit# INDEX переменный номер бита от 0 до 7	
Пометки		
См. также		
Пример:		

```
  ASSIGN INPUT IN_B4 AS  BIT 0          ;состояние порядкового бита 0
                                        ;соответствует состоянию входа B4
```

DEFINE	Определить имя	Псевдокоманда
Назначение	DEFINE используется для определения символических имен переменных и	

константа для удобства программирования. Это псевдо предложение, то есть оно не выполняется. DEFINE можно использовать также для замены цепочки символов.

Синтаксис DEFINE <имя> <цепочка>
имя любая цепочка символов
цепочка любая цепочка символов

Пометки DEFINE должен находиться перед любым выполняемым предложением

Пример:

```
DEFINE  Пять 5
DEFINE Три 3
DEFINE  Результат V1
DEFINE  SUMM Пять + Три

ProgramStart:
Result = Five + Three                    ;тоже, что и  V1 = 5+3
Result = SUMM                            ;тот же результат, что и выше
End
```

DISABLE	**Выключает сервомеханизм**	**Команда**

Назначение DISABLE выключает питание двигателя и блокирует сервомеханизм.

Синтаксис DISABLE

Пометки Как только выполняется команда DISABLE, выключается питание привода и двигатель свободно вращается. APOS продолжит отображение текущего положения двигателя. Даже если обновить TPOS до величины APOS, при выполнении ENABLE рекомендуется вернуть двигатель в исходную позицию.

См. также ENABLE

Пример:

```
DISABLE
```

DO UNTIL	**Делать/Пока не**	**Команда**

Назначение DO <команду(ы)> UNTIL <условие> выполняет команду(ы) между DO и UNTIL повторно пока условие не станет TRUE (истинным).

Синтаксис DO <предложение(я)> UNTIL <условие>
<команда(ы)> любая действительная команда(ы)
<условие> Условие, которое необходимо протестировать.
Условие может быть сравнением, истинным или ложным входом, системным флажком или переменной, используемой как флажок (если 0 – ложь, другая – истина). Сравнения сравнивают величины двух операндов и определяют, истинно или ложно условие. Сравнение может быть больше (>), меньше (<), меньше или равно (<=), или больше или равно (>=). Операнды сравнения могут быть пользовательской переменной, системными переменными, величинами аналогового входа или константами.

```
IN_A1            ;вход оценен как «истинный», когда активен
V1               ;пользовательская переменная. «Истина», когда
                 не 0, «ложь» при 0
INPOSITION       ;системный флажок
V1 > V2          ;сравнение пользовательской переменной
V1 > APOS        ;сравнение  пользовательской  и  системной
                 переменных
APOS < 8.4       ;сравнить системную переменную и константу
```

Пометки В отличие от предложения WHILE, тело цикла всегда будет выполнятся по крайней мере один раз, потому что предложение DO/UNTIL проверяет <условие> AFTER (после) того, как тело цикла выполнено.

См. также WHILE, IF

Пример:

```
DO MOVED V1      ;Продолжить введение циклов при помощи предложений Do Move
UNTIL IN_B4      ;Пока не включится вход
WHILE IN_A2      ;IN_A2 активирован (TRUE)
```

ENABLE	**Включает сервомеханизм**	**Команда**

Назначение Включает питание двигателя и активирует сервопривод

Синтаксис ENABLE

Пометки
См. также DISABLE
Пример:
```
ENABLE                    ;разрешение на запуск сервопривода
```

END	**Окончание программы**	**Команда**

Назначение Используется для завершения (окончания) пользовательской программы и ее событий
Синтаксис END
Пометки END можно использовать по всей программе
См. также DISABLE
Пример:
```
ENABLE                    ;разрешение на запуск сервопривода
```

EVENT	**Запуск обработчика событий**	**Команда**

Назначение Ключевое слово EVENT создает обработчика сканирования событий. Предложение также задает один из четырех возможных типов событий.
Синтаксис
1. EVENT <имя> INPUT <имявхода>
Или
2. EVENT <имя> INPUT !<название входа>
Или
3. EVENT <имя> TIME <период >
Или
4. EVENT <имя> <выражение>

имя	любая действительная буквенно-цифровая строка
имявхода	любой действительный вход “IN_A1 - IN_C4”
период	любое целое число. Выражается в мс
выражение	любое арифметическое или логическое выражение

Последующие команды нельзя использовать в обработчике событий:
MOVE, MOVED, MOVEP, MOVEDR, MOVEPR,MDV
MOTION SUSPEND
MOTION RESUME
STOP MOTION
DO UNTIL
GOTO
GOSUB
HALT
VELOCITY ON/OFF
WAIT
WHILE
При запрещенных GOTO или GOSUB, для изменения хода программы из обработчика событий можно использовать специальную команду JUMP.
См. описание JUMP в разделе Языковые таблицы.

Пометки
Для синтаксиса1 и 2:
Событие будет иметь место, когда вход с <именем/числом> перейдет от L (ложь) к H (истина), для синтаксиса 1 и от H к L для синтаксиса 2.
Для синтаксиса 3:
Событие будет иметь место, когда выделенный <период>, период времени закончится. Это событие можно использовать как периодическое для проверки некоторых условий.
Для синтаксиса 4
Событие будет иметь место, когда выражение, <выражение>, будет оценено как истинное. Выражение может быть любым арифметическим или логическим выражением или их комбинацией. Данное событие можно использовать при реализации программных концевых выключателей или при изменении хода программы на основе некоторых условий. В выражении можно использовать любую переменную (пользовательскую и системную) или константы.
См. также ENDEVENT, EVENT ON, EVENT OFF

Пример:

```
V0=0
V1=0
EVENT в событии IN_A1
V0 = V0+1               ;считать
ENDEVENT
EVENT period TIME1000   ;1000 мс = 1сек
V3=V0-V1                ;новый подсчет – старый подсчет = кол-во импульсов в секунду
V0=V1                   ;сохранить как старый подсчет
;-------------------------------------------------------------------------------
EVENT  InEvent  ON
EVENT period    ON
{программные предложения}
END
```

ENDEVENT	**Завершение обработчика событий**	**Команда**
Назначение	Показывает завершение обработчика событий	
Синтаксис	ENDEVENT	
Пометки		
См. также	EVENT, EVENT ON, EVENT OFF	

Пример:	`EVENT InputRise IN_B4`
	`V0=V+1`
	`ENDEVENT`	

EVENT ON/OFF	**Включает или выключает события**	**Команда**
Назначение	включает или выключает события, созданные предложением обработчика EVENT	
Синтаксис	EVENT <имя> ON EVENT <имя> OFF <имя> имя обработчика событий	
Пометки		
См. также	EVENT	
Пример:	`EVENT InputRise ON` `EVENT InputRise OFF`	

EVENT ON/OFF	**Глобально активирует/деактивирует события**	**Команда**
Назначение	Активирует/Деактивирует выполнение событий, предварительно активированных командой EVENT<Имясобытия>ON. Это глобальный контроль ON/OF. Влияет на флаг#30 в DSTATUS регистре - F_EVENTSOFF. После выполнения EVENTS ON включенные/выключенные состояния отдельных событий пересохраняются.	
Синтаксис	EVENTS ON Пересохраняет выполнение предварительно активированных событий. EVENTS OFF Блокирует выполнения всех событий	
Пометки	События глобально активируются после перезагрузки и контролируются предложениями индивидуальных событий Имясобытия ON.	
См. также	EVENT ON/OFF	
Пример:		

```
*******************************************************************************
EVENT SKIPOUT IN_B4    ;проверить край входного сигнала входа B4
JUMP TOGGLE            ;переадресовать выполнение кода на TOGGLE(переключить)
ENDEVENT               ;завершить событие
EVENT OVERSHOOT IN_B3 ;проверить край входного сигнала входа B3
JUMP SHUTDOWN          ;переадресовать выполнение кода на  SHUTDOWN(завершение)
ENDEVENT               ;завершить событие
*******************************************************************************
EVENT SKIPOUT ON
EVENT OVERSHOOOT ON
*******************************************************************************
.....Пользовательский код......
EVENTS OFF             ;выключает все события
……..…Пользовательский код……..
EVENTS ON              ;включает любое предварительно активированное событие
```

FAULT	Ошибка, генерируемая пользователем	Команда

Назначение Позволяет пользовательской программе задать заказную системную ошибку. Это полезно, когда специальной программе требуется стандартная обработка ошибок в специальных условиях, как предоставленные интерфейсом данные вне диапазона и т.д. Номера заказных ошибок должны быть в диапазоне от 128 до 240 (десятичные)

Синтаксис FAULT Номер ошибки Задает системную ошибку.
Номер ошибки – константа в диапазоне 128-240
Переменные в этой команде не разрешаются.

Пометки Заказная ошибка будет обработана как любая обычная ошибка. В учет неисправностей будет внесена запись.

См. также ON FAULT

Пример:

```
FAULT 200  ;Задает #200
V0=200
FAULT V0   ;Не действителен. Переменные здесь не разрешены
```

GOTO	Идти к	Команда

Назначение Передать выполнение программы команде, следующей за меткой.

Синтаксис GOTO <метка>

Пометки

См. также GOSUB, JUMP

Пример:

```
GOTO Label2
{команды...}
Label2:   {команды...}
```

GOSUB	Перейти к подпрограмме	Команда

Назначение GOSUB передает управление <имяподпрограммы> подпрограмме.

Синтаксис GOSUB <subname>
<subname> действительное имя подпрограммы

Пометки После возвращения из подпрограммы программа начинает выполнение со следующей после предложения GOSUB строки.

См. также GOTO, JUMP, RETURN

Пример:

```
DO
GOSUB  CALCMOVE
MOVED V1
WHILE  1
END
SUB CALCMOVE
V1=(V2+V3)/2
RETURN
```

HALT	Остановить выполнение программы	Команда

Назначение Используется для остановки выполнения главной программы. Предложение HALT не останавливает события. Выполнение снова запустится предложением RESET или выполнением JUMP к коду из EVENT обработчика.

Синтаксис HALT

Пометки Это предложение удобно для написания событийно-управляемых программ.

См. также RESET

Пример:

```
{Команды…}
HALT
```

JUMP	Скачок к метке из обработчика событий	Команда

Назначение Это предложение специального назначения для использования только в программе обработчика событий. Когда событие запущено, а предложение обработано, выполнение пользовательской программы передается аргументу <метка>, вызванному в предложении “JUMP”. Это предложение особенно полезно, когда существует необходимость в изменении хода программы на основе некоторых событий. Выполнение программы передается команде, следующей за меткой.

Синтаксис JUMP <метка> <метка> является меткой любой действительной программы

Пометки Может использоваться только в EVENT обработчике.

См. также EVENT

Пример:

```
{Команды…}
EVENT ExternalFault INPUT IN_A3   ;активировать событие при активации IN_A3
JUMP ExecuteStop                  ;направить ход программы к <ExecuteStop>
ENDEVENT
{команды…}
StartMotion:
EVENT ExternalFault ON
 ENABLE
MOVED 20
MOVED -100
{команды}
END
ExecuteStop:
STOP MOTION                       ;Здесь движение останавливается
DISABLE                           ;привод выключен
GOTO StartMotion
```

ICONTROL ON/OFF	**Активирует интерфейс-контроль**	**Команда**

Назначение Активирует/Деактивирует интерфейс-контроль. Влияет на флажок #27 в DSTATUS регистре F_ICONTROLOFF. Деактивируются все команды движения интерфейса и команды, меняющие любые выходы. За подробностями обращайтесь к руководству команд главного интерфейса. Эта команда полезна, когда программа обрабатывает критические состояния (как например концевые выключатели) и не может быть нарушена интерфейсом (обычно асинхронное тело по отношению к состоянию/событию программы)

Синтаксис

ICONTROL ON	Активирует интерфейс-контроль
ICONTROL OFF	Деактивирует интерфейс-контроль

Пометки После перезагрузки интерфейс-контроль активируется по умолчанию.

См. также

Пример:

```
EVENT LimitSwitch IN_A1      ;событие концевого выключателя
Jump LimitSwitchHandler      ;скачок к обработке концевого выключателя
ENDEVENT
V0=0
EVENT LimitSwitch ON
Again:
HALT                         ;система контролируется интерфейсом
LimitSwitchHandler:
EVENTS OFF                   ;выключить все события
ICONTROL OFF                 ;деактивировать интерфейс-контроль
STOP MOTION QUICK
DISABLE                      ;опционная деактивация
V0=1                         ;обозначить условие ошибки для интерфейса
ICONTROL ON
EVENTS ON
GOTO AGAIN
```

IF	If/Then/Else	Команда

Назначение IF тестирует условие, а затем выполняет специальное действие(я) между IF и ENDIF, если условие удовлетворено. Если условие ложно, действие не предпринимается, и выполняются команды, следующие за ENDIF. Опционально, используя ELSE, можно задать выполнение второго действия(й) при ложном условии.

Синтаксис

```
IF <условие>
{предложения 1}
ELSE
{предложения  2}
ENDIF
```

<Условие> подлежит тестированию. Это условие может быть сравнением, истинным или ложным входом, системным флагом или переменной, используемой в качестве флага (если 0 - ложь, другое - истина).

Сравниваются величины двух операндов и определяется, истинное условие или ложное. Сравнение может быть больше (>), меньше (<), меньше или равно (<=), или больше или равно (>=). Операнды сравнения могут быть пользовательской переменной, системными переменными, значениями аналогового входа, или константами.

```
IN_A1           ;вход оценивается истинным, если он активен
V1              ;пользовательская переменная.
                ;Истина при не 0, ложь при 0
INPOSITION      ;системный флаг
V1 > V2         ;сравнение пользовательской переменной
V1 > APOS       ;сравнение пользовательской и
                ;системной переменных
APOS < 8.4      ;сравнить системную переменную и константу
{Statements 1}  ;команды будут выполнены, если условие истинно
{Statements 2}  ;команды будут выполнены, если условие ложно
```

Пометки Только {Предложения 1} или {Предложения 2} будут выполнены. Невозможно выполнить и те, и другие.

См. также WHILE, DO

Пример:

```
IF APOS  > 4
V0=2
;-------------------------------------------------------
ELSE
V0=0
ENDIF
;-------------------------------------------------------
If V1 <> V2 && V3>V4
V2=9
ENDIF
```

MOVE	Перемещение	Команда

Назначение MOVE UNTIL выполняет движение, пока условие не станет истинным. MOVE WHILE выполняет движение в тот период времени, когда условие остается истинным. Предложение приостанавливает выполнение программы до завершения движения, за исключением предложения, используемого с C модификатором.

Синтаксис MOVE [BACK] UNTIL <condition> [,C]
MOVE [BACK] WHILE <condition> [,C]

BACK	Изменяет направление перемещения.
C (опция)	C[ontinue] – модификатор позволяет программе продолжаться, пока выполняется движение. Если второй профиль движения выполнен, в то время как первый еще в движении, второй профиль будет загружен в стек движения. Глубина стека движения составляет 32 элемента. Программист должен проверять системный флажок “F_MQUEUE_FULL”, чтобы убедиться, что в очереди есть свободное место. Если очередь заполняется или переполняется, привод сгенерирует ошибку.
<условие>	Условие, которое необходимо протестировать. Условие может выражаться сравнением, истинным или ложным входом, системным флагом или переменной, используемой в качестве флага (при 0 - ложь, другое - истина).

Пометки

См. также MOVEP, MOVED, MOVEPR, MOVEDR, MDV, MOTION SUSPEND, MOTION RESUME

Пример:

```
Команды…}
MOVE UNTIL V0<3
MOVE BACK UNTIL V0>4
MOVE WHILE V0<3
MOVE BACK WHILE V0>4
MOVE WHILE V0<3,C
```

MOVED	Расстояние перемещения	Команда

Назначение MOVED выполняет инкрементное движение (расстояние), указанное в пользовательских единицах. Требуемое расстояние может варьироваться от -231 до 231. Это предложение приостановит выполнение программы до завершения движения, за исключением команды с “C” модификатором. При использовании “S” модификатора ускорение по S-кривой осуществляется во время перемещения.

Синтаксис

C[ontinue]	MOVED <расстояние>[,S] [,C] “C” аргумент является опционным модификатором, который позволяет программе продолжать выполнение программы, во время выполнения профиля движения. Если привод выполняет предыдущий профиль движения, новый профиль движения будет загружен в стек движения. Глубина стека движения составляет 32 элемента. Программист должен проверять системный флажок “F_MQUEUE_FULL”, чтобы убедиться, что в очереди есть свободное место. Если очередь заполняется или переполняется, привод сгенерирует ошибку.
S[-кривая]	Опционнальный модификатор определяет ускорение по S-кривой.

См. также MOVE, MOVEP, MOVEPR, MOVEDR, MDV, MOTION SUSPEND, MOTION RESUME

Пример:

```
{Предложения…}
MOVED 3          ;перейти на 3 пользовательских единицы вперед
MOVED BACK 3     ;перейти на 3 пользовательских единицы назад
{Предложения…}
```

MOVEP	Перемещение в положение	Команда

Назначение MOVEP осуществляет движение в заданное абсолютное положение в пользовательских единицах. Требуемый диапазон для абсолютного перемещения – от -231 до 231. Данное предложение приостановит выполнение программы до завершения движения, за исключением предложения с использованием “C” модификатора. При использовании “S” модификатора, ускорение по S-кривой осуществляется во время перемещения.

Синтаксис MOVEP <абсолютное положение>[,S] [,C]

C[ontinue] “C” аргумент является опционным модификатором, который позволяет программе продолжать выполнение программы, во время выполнения профиля движения. Если привод выполняет предыдущий профиль движения, новый профиль движения будет загружен в стек движения. Глубина стека движения составляет 32 элемента. Программист должен проверять системный флажок “F_MQUEUE_FULL”, чтобы убедиться, что в очереди есть свободное место. Если очередь заполняется или переполняется, привод сгенерирует ошибку.

S[-кривая] опционный модификатор определяет ускорение по S-кривой.

См. также MOVE, MOVEP, MOVEPR, MOVEDR, MDV, MOTION SUSPEND, MOTION RESUME

Пример:

```
{команды…}
MOVEP 3          ;перейти на 3 пользовательские единицы абсолютного положения
{команды…}
```

MOVEDR	**Перемещение на зарегистрированное расстояние**	**Команда**

Назначение MOVEDR выполняет инкрементное движение, указанное в пользовательских единицах. Если во время движения регистрационный вход активируется (goes high), запишется фактическое положение, а значение смещения (второй аргумент в предложении MOVEPR) будет прибавлено к этому положению для образования нового целевого положения. Далее окончание перемещения будет видоизменено согласно этому новому целевому положению. Данное предложение приостанавливает выполнение программы до завершения перемещения, за исключением случая, когда предложение используется с “C” модификатором.

Синтаксис MOVEDR <расстояние>,<смещение> [,C]

C[ontinue] “C” аргумент является опционным модификатором, который позволяет программе продолжить выполнение пользовательской программы во время обработки профиля движения. Если во время обработки приводом перемещения, поступает запрос на новый профиль движения, он будет загружен в стек движения. Глубина стека движения составляет 32 элемента. Программист должен проверять системный флажок “F_MQUEUE_FULL”, чтобы убедиться, что в очереди есть свободное место. Если очередь заполняется или переполняется, привод сгенерирует ошибку.

См. также MOVE, MOVEP, MOVEPR, MOVED, MDV, MOTION SUSPEND, MOTION RESUME

Пример:

```
{Команды…}
MOVEDR 3, 2
{Команды…}
```

Данный пример перемещает двигатель на 3 пользовательских единицы и проверяет регистрационный вход. Если регистрация не будет обнаружена, перемещение завершится. При обнаружении регистрации записывается положение регистрации, и величина смещения 2 прибавляется к записанному регистрационному положению для расчета нового конечного положения.

MOVEPR	**Перемещение на зарегистрированное расстояние**	**Команда**

Назначение MOVEPR осуществляет перемещение абсолютного положения, определенного в пользовательских единицах. Если во время перемещения регистрационный вход активируется (goes high), тогда конечное положение перемещения сменяется новым целевым положением. Новое положение генерируется из второго аргумента в предложении MOVEPR, (смещение). Данное предложение приостанавливает выполнение программы до завершения перемещения, за исключением предложения с использованием “C” модификатора.

Синтаксис MOVEPR <расстояние>,<смещение> [,C]

C[ontinue] “C” аргумент является опционным модификатором, который позволяет программе продолжить выполнение пользовательской программы во время обработки профиля движения. Если во время обработки приводом перемещения, поступает запрос на новый профиль движения, он будет загружен в стек движения. Глубина стека движения составляет 32 элемента. Программист должен проверять системный флажок “F_MQUEUE_FULL”, чтобы убедиться, что в очереди есть свободное место. Если очередь заполняется или переполняется, привод сгенерирует ошибку.

См. также MOVE, MOVEP, MOVEPR, MOVED, MDV, MOTION SUSPEND, MOTION RESUME

Пример: Данный пример перемещает двигатель на 3 пользовательских единицы и проверяет регистрационный вход. Если регистрация не будет обнаружена, перемещение завершится. При обнаружении регистрации окончание перемещения сменяется новым абсолютным целевым положением с разницей в 2 пользовательские единицы.

```
{команды…}
MOVEPR 3, 2
{команды…}
```

MDV **Сегментное Перемещение** **Команда**

Назначение MDV определяет сегмент инкрементного движения, задавая в пользовательских единицах расстояние и конечную скорость (для каждого сегмента). Ускорение (или торможение) рассчитываются автоматически на основе этих двух параметров. Эта техника позволяет создавать сложные перемещения, состоящие из множества сегментов. Каждое MDV перемещение начинается и заканчивается с нулевой скоростью. На основе этого MDV перемещение должно состоять как минимум из двух сегментов. MDV предложение не приостанавливает выполнение главной программы. Каждый сегмент сразу же загружается в очередность движения. Если скорость последнего сегмента в очередности движения отлична от 0, привод сгенерирует ошибку #24 “Motion Queue Empty”(Очередь движения пуста). При использовании “S” модификатора в данном предложении ускорение/торможение будет осуществляться по S-кривой в отличие от линейного.

Синтаксис MDV <[-]расстояние сегмента>,<конечная скорость сегмента> [,S]

S[-кривая] опционный модификатор определяет S-кривую ускорения.

См. также MOVE, MOVEP, MOVEPR, MOVED, MDV, MOTION SUSPEND, MOTION RESUME

Пример:

```
{команды…}
MDV  5, 10      ;Пройти 5 пользовательских единиц и ускориться до скорости 10
MDV  10,10      ;Пройти 10 пользовательских единиц и достигнуть скорости 10
MDV 10,5        ;Пройти 10 пользовательских единиц и достигнуть скорости 5
MDV  5,;0       ;Пройти 5 пользовательских единиц и достигнуть скорости 0.
                ;Конечная скорость последнего MDV должна равняться 0.
{команды…}
```

MOTION SUSPEND **Приостановка** **Команда**

Назначение Данное предложение используется для временной приостановки движения без мигания сброса очереди движения. Если данное предложение выполняется во время обработки профиля движения, движение не будет приостановлено до завершения перемещения. При выполнении ряда сегментных MDV перемещений движение не будет приостановлено до тех пор, пока не будут обработаны все MDV сегменты. Если очередность движения пуста, любое предложение последующего движения будет загружено в очередь и останется там до выполнения предложения “Motion Resume” (возобновить движение). Любые предложения движения без “C” модификатора (кроме предложений MDV) заблокируют пользовательскую программу.

Чтобы проиллюстрировать эту блокировку программы, обратимся к следующей программе:

```
;Программа блокировки после выполнения MOVE
{команды}
MOTION SUSPEND  ;Движение удерживается, или приостановлено
MOVE 20         ;Профиль движения загружен в очередность
                ; движения и программа приостановлена до
                ; завершения перемещения.
…{предложения}  ;Данные предложения не будут выполнены потому
                ;что привод ожидает завершения перемещения,
                ;которое никогда не будет обработано,
                ;так как движение приостановлено.
MOTION RESUME   ;Как и приведенные выше предложения, эта
                ;команда не выполнится, и программа LOCKED-UP.
```

Эту ситуацию можно разрешить только кнопкой перезагрузки или выполнением команды MOTION RESUME из главного интерфейса.

Синтаксис MOTION SUSPEND

Пометки Выполнение любых MOVEx команд без“C” модификатора заблокирует пользовательскую программу. Разблокировать ее вы сможете только кнопкой перезагрузки или выполнением команды “Motion Resume” из главного интерфейса.

См. также MOVE, MOVEP, MOVEDR, MOVED, MOVEPR ,MDV, MOTION RESUME

Пример:

```
…{команды}
MOTION SUSPEND    ;После текущего движения движение будет приостановлено
                  ;команда завершена.
…{команды}
```

MOTION RESUME	**Возобновление**	**Команда**

Назначение Предложение возобновляет движение, предварительно приостановленное командой MOTION SUSPEND. Если движение не было приостановлено, никакого воздействия на операцию не будет.

Синтаксис MOTION RESUME

См. также MOVE, MOVEP, MOVEDR, MOVED, MOVEPR ,MDV, MOTION RESUME

Пример:

```
…{команды}
MOTION RESUME    ;Движение возобновлено от фактической команды в очереди
…{команды}       движения (если такая имеется)
```

ON FAULT/ ENDFAULT	**Возобновление**	**Команда**

Назначение Этой командой начинается раздел обработчика ошибок в пользовательской программе. Обработчик ошибок является частью кода, выполняемого при появлении в приводе ошибки. Программа обработчика ошибок должна начинаться с “ON FAULT” и заканчиваться “ENDFAULT”. Если программа обработчика ошибок не определена, тогда пользовательская программа будет останавливаться каждый раз, когда привод обнаружит ошибку. Впоследствии, когда обработчик ошибок будет определен, и будет обнаружена ошибка, привод будет заблокирован и выполнится программа обработчика ошибок. Команды RESUME и RESET можно использовать, чтобы перенаправить выполнение программы из обработчика ошибок назад в главную программу. Если эти их не использовать, тогда программа остановится, как только будет выполнено ENDFAULT.
В обработчике ошибок нельзя использовать следующие предложения:
MOVE, MOVED, MOVEP, MOVEDR, MOVEPR, MDV, MOTION SUSPEND, MOTION RESUME, GOTO, GOSUB, JUMP, ENABLE, GEAR ON/OFF, и VELOCITY ON/OFF

Синтаксис
```
ON FAULT
{…команды}
ENDFAULT
```

См. также RESUME, RESET

Пример:
```
…{Команды}          ;пользовательская программа
FaultRecovery:      ;процедура восстановления после ошибки
…{Команды}
END
ON FAULT            ;При появлении ошибки программа направляется сюда
…{Команды}          ;любой код для устранения ошибки
RESUME              ;Выполнение RESUME завершает обработчик ошибок и направляет
FaultRecovery       ;выполнение назад в пользовательскую программу. Если
                    ;игнорировать RESUME, программа остановится здесь.
ENDFAULT            ;Программа ошибки должна завершиться предложением ENDFAULT
```

REGISTRATION ON — Регистрация включена — Команда

Назначение Это предложение активизирует регистрационный вход (вход IN_C3). Когда регистрационный вход активирован, устанавливается флаговая переменная “F_REGISTRATION”, и фактическое положение фиксируется и сохраняется в системную переменную “RPOS”. Обе эти переменные доступны пользовательской программе в целях принятия решений. Предложение “REGISTRATION ON” также перезагрузит флаг “F_REGISTRATION”.

Синтаксис REGISTRATION ON Флаг “F_REGISTRATION” перезагружен, и регистрационный вход активирован.

См. также MOVEDR, MOVEPR

Пример:
```
; Осуществляет перемещение до тех пор, пока вход не активируется и не перейдет
обратно в положение датчика.
…{команды}
REGISTRATION ON
MOVE UNTIL IN_C3    ;Активирует регистрационный вход
MOVEP  RPOS
…{команды}          ;Перемещается, пока активирован вход, (ответ датчика)
                    ;Абсолютное перемещение к положению датчика
```

RESUME — Возобновление — Команда

Назначение Это предложение перенаправляет выполнение кода из программы обработчика событий назад в пользовательскую программу. Специальная направляемая строка пользовательской программы вызывается в <метке> аргумента и в предложении “RESUME”. Это предложение разрешено только в программе обработчика событий.

Синтаксис RESUME <label> <label> ссылка на адрес метки в пользовательской программе

См. также ON FAULT

Пример:

```
…{команды}
FaultRecovery:
…{команды}
END
ON FAULT          ;При появлении ошибки программа направляется сюда
…{команды}        ;любой код для устранения ошибки
RESUME            ;Выполнение RESUME завершает обработчик ошибок и направляет
FaultRecovery     ;выполнение за метку "FaultRecovery"(восстанов. после ошибки)
                  ;в пользовательской программе. Если игнорировать RESUME,
ENDFAULT          ;программа остановится здесь.
                  ;Программа ошибки должна завершиться предложением ENDFAULT
```

RETURN **Возврат из подпрограммы** **Команда**

Назначение Это предложение вернет выполнение кода из подпрограммы в точку программы, из которой была вызвана подпрограмма. Если это предложение выполняется без предварительного обращения к подпрограмме, (GOSUB), результатом станет ошибка #21 “Subroutine stack underflow” (Потеря значимости стека подпрограммы).

Синтаксис RETURN

См. также GOTO, GOSUB

Пример:

```
…{команды}
GOSUB MySub       ;Программа совершает скачок в подпрограмму "MySub"
MOVED 10          ;Выполнить перемещение сразу после выполнения подпрограммы
                  ;предложение RETURN.
…{команды}
END               ;завершение главной программы
MySub:            ;подпрограмма, вызванная из пользовательской программы
…{команды}        ;Код для выполнения в подпрограмме
RETURN            ;Возвращает выполнение в строку кода программы под командой
                  ; "GOSUB", (предложение MOVED 10).
```

SEND/SEND TO **Отправляет значение сетевой переменной(ьх)** **Команда**

Назначение Это предложение используется для совместного использования значения сетевых переменных между приводами по сети Ethernet. Сетевыми являются переменные N0 через N31. Переменные, которые необходимо послать, или синхронизировать вызываются предложением “SEND”. Например, “SEND [N5]” возьмет фактическое значение переменной N5 и загрузит его в переменную N5 каждого привода сети. Предложение SENDTO только обновляет сетевые переменные приводов одной группы ID, перечисленной в команде.

Синтаксис			
	SEND [Na,Nb, Nx-Ny],	a,b,x,y	Любое число от 0 до 31
	SENDTO GroupID [Na,Nb, Nx-Ny]	ГруппаID	ГруппаID приводов, чьи переменные будут затронуты (синхронизированы)

См. также Сетевые переменные

Пример:

```
…{команды}
N1=12              ;Задать N1 равной 12
SEND [N1]          ;Задать переменной N1 значение 12 в каждом приводе сети.
SEND [N5-N10]      ;Задается переменная N5 через N10 во всех приводах сети.
N20=25             ;Задать N20 равной 25
SENDTO 2 [N20]     ;Задать переменной N20 значение 25 только
                   ;в приводах с  GroupID = 2
.…{команды}
END                ;Завершение главной программы
```

STOP MOTION [Quick]	**Останавливает движение**	**Команда**

Назначение Это предложение используется для остановки всего движения. При выполнении предложения “STOP MOTION” удаляются все профили движения, сохраненные в очередности движения и движение немедленно будет остановлено параметром торможения, заданном в переменной “DCEL”. При использовании модификатора “QUICK” значение торможения будет взято из переменной “QDECEL”. Основное использование этой команды заключается в управлении аварийными остановками или при обнаружении ограничительного датчика. Обратите внимание, что фактическое положение не потеряется после выполнения данного предложения.

Синтаксис		
	STOP MOTION	Прекращается использование коэффициента торможения DECEL
	STOP MOTION QUICK	Прекращается использование коэффициента торможения QDECEL

См. также MOTION SUSPEND

Пример:

```
…{команды}
DECEL = 100
QDECEL = 10000.
…{команды}
STOP MOTION QUICK
```

VELOCITY ON/OFF	**Режим скорости**	**Команда**

Назначение	Команда VELOCITY ON активирует режим скорости в приводе. Команда VELOCITY OFF деактивирует режим скорости и возвращает привод в режим по умолчанию. (Режимом по умолчанию является позиционирование). Значение скорости для данного режима задается настройкой системной переменной “VEL”. Все переменные, имеющие отношение к положению, действительны в этом режиме.
Синтаксис	VELOCITY ON VELOCITY OFF
Пометки	Команда “VELOCITY ON” одна из команд, имеющих отношение к движению. Ее необходимо реализовывать, когда привод активирован. Если выполнять “VELOCITY ON”, пока привод заблокирован, появится ошибка # 27 - ”Drive disabled” (Привод заблокирован). Выполнение любых профилей, имеющих отношение к движению, пока привод находится в режиме скорости, будет загружено в очередность движения. При выполнении “VELOCITY OFF” привод по умолчанию возвращается назад в режим положения и немедленно начинает выполнять профили движения, сохраненные в очереди движения. Пожалуйста, обратите внимание, что переменную “VEL” можно задать на лету, благодаря динамическому управлению скоростью.
См. также	

Пример:

```
VEL=0                ;Задать нулевую скорость
VELOCITY ON          ;Включить режим скорости
VEL = 10             ;Задать скорость
…{Предложения}
VELOCITY OFF         ;Выключить режим скорости
```

WAIT	**Ожидание**	**Команда**
Назначение	Это предложение приостанавливает выполнение программы до тех пор, пока не встретится(ятся) некоторое условие(я). Условия включают в себя выражения «истину» и «ложь», завершение заданного времени, MOTION COMPLETE (завершение движения).	
Синтаксис	WAIT UNTIL <выражение>	ожидание, пока выражение не станет «истиной»
	WAIT WHILE <выражение>	ожидание, пока выражение «истинно»
	WAIT TIME <задержка времени>	ожидание, пока не завершится <задержка времени> в мс
	WAIT MOTION COMPLETE	ожидание завершения последнего движения в очередности движения
Пометки		
См. также	DSTATUS Системная переменная, пользовательские переменные и раздел флагов	

Пример:

```
WAIT UNTIL (APOS>2 && APOS       ;ждать пока Apos станет > 2 и <3 APOS>1)
WAIT WHILE (APOS <2 && APOS>1)      ;ждать пока Apos is <2 и >1
WAIT TIME 1000                   ;ждать 1 Sec (1 Sec=1000mS)
MDV 20, 20                       ;начать MDV движение
MDV 20,0                         ;начать MDV движение
WAIT MOTION COMPLETE             ;ждать до завершения движения
```

WHILE / ENDWHILE	**В то время как**	**Предложение**
Назначение	WHILE <выражение> неоднократно выполняет предложение(я) между ключевыми словами WHILE и ENDWHILE, пока выражение оценивается как TRUE («истинное»).	
Синтаксис	WHILE <выражение> {команда(ы)}…ENDWHILE	
Пометки	Блок предложений WHILE должен завершиться ключевым словом ENDWHILE.	
См. также	DO/UNTIL	

Пример:

```
WHILE  APOS<3        ;Выполнять предложения пока Apos не <3
{предложение(я)}..
ENDWHILE
```

Appendix A. Complete list of variables.

В приведенной ниже таблице дан перечень доступных переменных 940. Доступ к этим переменным можно получить из пользовательской программы или любого поддерживаемого интерфейс-протокола, наподобие RPC через Ethernet, PPP через RS232 или MODBUS-RTU через порт RS485.
Доступ к любой переменной открывается при помощи ее имени из пользовательской программы или при помощи значения индекса с использованием синтаксиса: @<VARINDEX> , где <VARINDEX> переменный индекс из таблицы.
Доступ к любой переменной интерфейса открывается при помощи величины ее индекса. Колонка “Format” содержит собственный формат переменной:
W: 32 бита целое
F: переменное (real)
При задании значения переменной через внешнее устройство, значение может быть адресовано как переменное или целое. Значение будет автоматически подогнано для соответствия данной форме.
Колонка “EPM” показывает, есть ли у переменной долговременная область памяти в памяти EPM. Пользовательская программа использует RAM (временную) копию переменных, сохраненных в EPM. Изменения переменных в пользовательской программе не влияют на значения в EMP. Однако функции интерфейса могут изменять и временную и долговременную копии переменной. Если главный интерфейс запросит изменение (долговременного) значения EPM, это изменение осуществится и в RAM памяти пользовательской программы и в EPM. Когда пользовательская программа считывает переменную, она считывает ее RAM (временную) копию. Функции интерфейса обладают выбором считывания из RAM (временной) или из EPM (долговременной) копии переменной. При включении питания все RAM копии переменных инициализируются со значениями EPM.
Колонка “Access”(Доступ) показывает, доступна ли переменная только для чтения (R), только для записи (W), или для считывания/записи (R/W). Запись на переменную, предназначенную только для считывания, или считывание с переменной, предназначенной только для записи, не будет работать.
Колонка “Units”(Единицы) показывает единицы измерения переменной. Единицы, которые предназначены только для данного руководства и используются для движения следующие:
UU – пользовательские единицы
EC – единицы счета датчика положения
S – секунды
PPS – импульсов за замер. Время замера - 255us - коэффициент цикла сервомеханизма
PPSS – импульсов за замер за замер. Время замера - 255us - коэффициент цикла сервомеханизма

Код	Имя	Формат	EPM	Доступ	Описание	Единицы
1	VAR_IDSTRING		N	R	Строка идентификаторов привода	
2	VAR_NAME		Y	R/W	Идентификатор привода	
10	VAR_M_ID		Y	R	ID двигателя	
11	VAR_M_MODEL		Y	R	Тип двигателя	
12	VAR_M_VENDOR		Y	R	Производитель двигателя	
13	VAR_M_ESET		Y	R	Зарезервированный	
14	VAR_M_HALLCODE		Y	R	Индекс кода Холла	
15	VAR_M_HOFFSET		Y	R	Зарезервированный	
16	VAR_M_ZOFFSET		Y	R	Зарезервированный	
17	VAR_M_ICTRL		Y	R	Зарезервированный	
18	VAR_M_JM		Y	R	Двигатель Jm	
19	VAR_M_KE		Y	R	Двигатель Ke	
20	VAR_M_KT		Y	R	Двигатель Kt	
21	VAR_M_LS		Y	R	Двигатель Ls	
22	VAR_M_RS		Y	R	Двигатель Rs	
23	VAR_M_MAXCURRENT		Y	R	Макс. ток двигателя (RMS)	
24	VAR_M_MAXVELOCITY		Y	R	Макс. скорость двигателя	
25	VAR_M_NPOLES		Y	R	Число электродов двигателя	
26	VAR_M_ENCODER		Y	R	Разрешение кодировщика	
27	VAR_M_TERMVOLTAGE		Y	R	Номинальное напряжение двигателя	
28	VAR_M_FEEDBACK		Y	R	Тип обратной связи	
29	VAR_ENABLE_SWITCH_TYPE	W	Y	R/W	Функция входа разрешения 0-запретить 1-запуск	Бит
30	VAR_CURRENTLIMIT	F	Y	R/W	Предел по току	[A]мпер

Код	Имя	Формат	EPM	Доступ	Описание	Единицы
31	VAR_PEAKCURRENTLIMIT16	F	Y	R/W	Предел по току на частоте 16кГц	[A]мпер
32	VAR_PEAKCURRENTLIMIT	F	Y	R/W	Предел по току на частоте 8кГц	[A]мпер
33	VAR_PWMFREQUENCY	W	Y	R/W	Выбор частоты коммутации	

34	VAR_DRIVEMODE	W	Y	R/W	Выбор режима работы привода 0-крутящий момент 1-скорость 2-положение	
35	VAR_CURRENT_SCALE	F	Y	R/W	Шкала аналогового входа #1 в A/V	A/V
36	VAR_VELOCITY_SCALE	F	Y	R/W	Шкала аналогового входа #1 в RPM/V	об/V
37	VAR_REFERENCE	W	Y	R/W	Выбор адресации: 1 – внутренний источник 0 - внешний	
38	VAR_STEPINPUTTYPE	W	Y	R/W	Режим работы входов адресации положения: 1 – входы квадратуры (A/B) 0 – тип Шаг и Направление	
39	VAR_MOTORTHERMALPROTECT	W	Y	R/W	Функция тепловой защиты двигателя: 0 - деактивирована 1 – активирована	
40	VAR_MOTORPTCRESISTANCE	F	Y	R/W	Сопротивления тепловой защиты двигателя PTC в Омах	[Ом]
41	VAR_SECONDENCODER	W	Y	R/W	Второй энкодер: 0 - деактивирован 1 – активирован	
42	VAR_REGENDUTY	W	Y	R/W	Ток регенерации ШИМ в % Диапазон: 0-100%	%
43	VAR_ENCODERREPEATSRC	W	Y	R/W	Выбор источника для буфера повторения: 0 – энкодер (терминал P4) 1 – Модуль обратной связи (если доступен в особенном модуле)	
44	VAR_VP_GAIN	W	Y	R/W	Коэффициент П-регулятора скорости Диапазон: 0 - 32767	
45	VAR_VI_GAIN	W	Y	R/W	Коэффициент И-регулятора скорости Диапазон: 0 - 16383	
46	VAR_PP_GAIN	W	Y	R/W	Коэффициент П-регулятора положения Диапазон: 0 – 32767	
47	VAR_PI_GAIN	W	Y	R/W	Коэффициент И-регулятора положения Диапазон: 0 – 16383	
48	VAR_PD_GAIN	W	Y	R/W	Коэффициент Д-регулятора положения Диапазон: 0 – 32767	
49	VAR_PI_LIMIT	W	Y	R/W	Предел И-регулятора положения Диапазон: 0 – 20000	
51	VAR_VREG_WINDOW	W	Y	R/W	Коэффициент мастабирования шкалы Диапазон: -5 – +4	
52	VAR_ENABLE	W	N	W	Активация/Деактивация ПО 0 - деактивировать 1 – активировать	
53	VAR_RESET	W	N	W	Перезагрузка привода (холодная перезагрузка) 0 – действия нет 1 – перезагрузить привод	
54	VAR_STATUS	W	N	R	Регистр состояния привода	
55	VAR_BCF_SIZE	W	Y	R	Размер байткода пользовательской программы	Байты
56	VAR_AUTOBOOT	W	Y	R/W	Флажок автозапуска пользовательской программы. 0 – ручной запуск программы (MotionView или интерфейс) 1 – автоматический запуск программы после загрузки привода	

Код	Имя	Формат	EPM	Доступ	Описание	Единицы
57	VAR_GROUPID	W	Y	R/W	ID сетевой группы Диапазон 1-32767	
58	VAR_VLIMIT_ZEROSPEED	F	Y	R/W	Значение нулевой скорости Диапазон: 0-100	Об
59	VAR_VLIMIT_SPEEDWND	F	Y	R/W	Диапазон скорости Диапазон: 10-10000	Об

60	VAR_VLIMIT_ATSPEED	F	Y	R/W	Конечная скорость для заданного диапазона Диапазон: -10000 -+10000	Об
61	VAR_PLIMIT_POSERROR	W	Y	R/W	Ошибка положения Диапазон: 1-32767	EC
62	VAR_PLIMIT_ERRORTIME	F	Y	R/W	Время выявления ошибки положения Диапазон 0,25-8000	мС
63	VAR_PLIMIT_SEPOSERROR	W	Y	R/W	Ошибка позиционирования 2-ого энкодера Диапазон: 1-32767	EC
64	VAR_PLIMIT_SEERRORTIME	F	Y	R/W	Время выявления ошибки положения 2-ого энкодера Диапазон: 0,25-8000	мС
65	VAR_INPUTS	W	N	R	Состояние цифровых входов A1- Bit0, A2- Bit1…C4- Bit11	
66	VAR_OUTPUTS	W	N	W	Состояния цифровых выходов. Запись на эти переменные задает значение/перезапускает цифровые выходы, кроме выходов, которым была присвоена специальная функция. Выход 1 Bit0 Выход 2 Bit 1 Выход 3 Bit 2 Выход 4 Bit 3 Выход 5 Bit 4	
67	VAR_IP_ADDRESS	W	Y	R/W	IP адрес Ethernet. IP адрес изменяется при следующей начальной загрузке. 32 бита	
68	VAR_IP_MASK	W	Y	R/W	Сетевая маска IP Ethernet. Маска меняется при следующей начальной загрузке. Значение 32 бита.	
69	VAR_IP_GATEWAY	W	Y	R/W	IP адрес входа Ethernet. IP адрес изменяется при следующей начальной загрузке. Значение 32 бита.	
70	VAR_IP_DHCP	W	Y	R/W	Использование DHCP 1 – ручное 2 – использование DHCP сервисов	
71	VAR_AIN1	F	N	R	Текущее значение аналогового входа AIN1	[V]olt
72	VAR_AIN2	F	N	R	Текущее значение аналогового входа AIN2	[V]olt
73	VAR_BUSVOLTAGE	F	N	R	Напряжение шины постоянного тока	[V]olt
74	VAR_HTEMP	F	N	R	Температура радиатора. 0 – для температур < 40C и фактическая температура для температур >40 C.	[C]
75	VAR_ENABLE_ACCELDECEL		Y	R/W	Активировать функцию ускорения / торможения для режима скорости 0-деактивировать 1-активировать	
76	VAR_ACCEL_LIMIT	F	Y	R/W	Величина ускорения для режима скорости Диапазон: 0,1- 5000000	Об/мин *с
77	VAR_DECEL_LIMIT	F	Y	R/W	Величина торможения для режима скорости Диапазон: 0,1- 5000000	Об/мин *с
78	VAR_FAULT_RESET	W	Y	R/W	Перезагрузить конфигурацию ошибок 1 – для деактивации входа (A3) Активировать/Запретить 0 - для активации входа (A3) Активировать/Запретить	
79	VAR_M2SRATIO_MASTER	W	Y	R/W	Коэффициент передачи Master к системе. Диапазон единиц отсчета Master -32767 - +32767	EC
80	VAR_M2SRATIO_SYSTEM	W	Y	R/W	Коэффициент передачи Master к системе. Диапазон системных единиц отсчета 1 – 32767	EC

Код	Имя	Формат	EPM	Доступ	Описание	Единицы
81	VAR_S2PRATIO_SECOND	W	Y	R/W	Отношение второго энкодера к первому. Числитель: -32767 - +32767	
82	VAR_S2PRATIO_PRIME	W	Y	R/W	Отношение второго энкодера к первому. Знаменатель: 1-32767	
83	VAR_EXSTATUS	W	N	R	Расширенное состояние. Копия низшего слова флагов статуса DSP.	
84	VAR_HLS_MODE	W	Y	R/W	Аппаратные концевые выключатели. 0 – не используются	

					1 – остановка и ошибка 2 – ошибка	
85	VAR_AOUT_FUNCTION	W	Y	R/W	Диапазон функции аналогового выхода 0 - не присвоен 1 – фазовый ток 2 – фазовый ток (пиковое значение) 3 – скорость двигателя 4 – фазовый ток R 5 – фазовый ток S 6 - фазовый ток T 7 – ток Iq	
86	VAR_AOUT_VELSCALE	F	Y	R/W	Масштаб аналогового выхода по скорости. Диапазон 0,1-5.	mV/Об/мин
87	VAR_AOUT_CURSCALE	F	Y	R/W	Масштаб аналогового выхода по току. Диапазон 0,1-10	V/A
88	VAR_AOUT	F	N	W	Значение аналогового выхода. (Активно при VAR #84 = 0 Диапазон 0-10.	V
89	VAR_AIN1_DEADBAND	F	Y	R/W	Зона нечувствительности аналогового входа #1. Применяется при адресации тока или скорости. Диапазон 0-50.	mV
90	VAR_AIN1_OFFSET		Y	R/W	Смещение аналогового входа #1. Применяется при адресации тока/ скорости Диапазон: -1000 -+1000	mV
91	VAR_SUSPEND_MOTION	W	N	R/W	Останавливает движение генератора траектории. Текущее движение завершается, перед остановкой движения 0 - движение активировано 1- движение деактивировано	
92	VAR_MOVEP	W	N	W	Целевое положение для абсолютного движения. Записанное значение выполняет перемещение в положение как MOVEP, используя фактические величины ускорения, торможения и максимальной скорости.	
93	VAR_MOVED	W	N	W	Инкрементное положение. Записанное значение <>0 выполняет инкрементное движение как MOVED, используя фактические величины ускорения, торможения и максимальной скорости.	
94	VAR_MDV_DISTANCE	F	N	W	Расстояние для MDV перемещения	UU
95	VAR_MDV_VELOCITY	F	N	W	Скорость для MDV перемещения. Запись в эту переменную выполняет MDV перемещение на расстояние, записанное последней в переменную#94.	UU
96	VAR_MOVE_PWI1	W	N	W	Величина записи выполняет перемещение в положительном направлении, пока вход активный. Величина определяется # входа	
97	VAR_MOVE_PWI0	W	N	W	Величина записи выполняет перемещение в положительном направлении, пока вход не активный. Определяет # входа.	
98	VAR_MOVE_NWI1	F	N	W	Значение выполняет перемещение в отрицательном направлении, пока вход активный. Определяет # входа.	
99	VAR_MOVE_NWI0	F	N	W	Величина записи выполняет перемещение в отрицательном направлении, пока вход «ложный» (не активный). Величина определяет номер входа.	
Код	**Имя**	**Формат**	**EPM**	**Доступ**	**Описание**	**Единицы**
100	VAR_V0	F	N	R/W	Пользовательская переменная. Определяется пользователем.	
101	VAR_V1	F	N	R/W	Пользовательская переменная. Определяется пользователем.	
102	VAR_V2	F	N	R/W	Пользовательская переменная. Определяется пользователем.	
103	VAR_V3	F	N	R/W	Пользовательская переменная. Определяется пользователем.	
104	VAR_V4	F	N	R/W	Пользовательская переменная. Определяется пользователем.	

105	VAR_V5	F	N	R/W	Пользовательская переменная. Определяется пользователем.	
106	VAR_V5	F	N	R/W	Пользовательская переменная. Определяется пользователем.	
107	VAR_V7	F	N	R/W	Пользовательская переменная. Определяется пользователем.	
108	VAR_V8	F	N	R/W	Пользовательская переменная. Определяется пользователем.	
109	VAR_V9	F	N	R/W	Пользовательская переменная. Определяется пользователем.	
110	VAR_V10	F	N	R/W	Пользовательская переменная. Определяется пользователем.	
111	VAR_V11	F	N	R/W	Пользовательская переменная. Определяется пользователем.	
112	VAR_V12	F	N	R/W	Пользовательская переменная. Определяется пользователем.	
113	VAR_V13	F	N	R/W	Пользовательская переменная. Определяется пользователем.	
114	VAR_V14	F	N	R/W	Пользовательская переменная. Определяется пользователем.	
115	VAR_V15	F	N	R/W	Пользовательская переменная. Определяется пользователем.	
116	VAR_V16	F	N	R/W	Пользовательская переменная. Определяется пользователем.	
117	VAR_V17	F	N	R/W	Пользовательская переменная. Определяется пользователем.	
118	VAR_V18	F	N	R/W	Пользовательская переменная. Определяется пользователем.	
119	VAR_V19	F	N	R/W	Пользовательская переменная. Определяется пользователем.	
120	VAR_V20	F	N	R/W	Пользовательская переменная. Определяется пользователем.	
121	VAR_V21	F	N	R/W	Пользовательская переменная. Определяется пользователем.	
122	VAR_V22	F	N	R/W	Пользовательская переменная. Определяется пользователем.	
123	VAR_V23	F	N	R/W	Пользовательская переменная. Определяется пользователем.	
124	VAR_V24	F	N	R/W	Пользовательская переменная. Определяется пользователем.	
125	VAR_V25	F	N	R/W	Пользовательская переменная. Определяется пользователем.	
126	VAR_V26	F	N	R/W	Пользовательская переменная. Определяется пользователем.	
127	VAR_V27	F	N	R/W	Пользовательская переменная. Определяется пользователем.	
128	VAR_V28	F	N	R/W	Пользовательская переменная. Определяется пользователем.	
129	VAR_V29	F	N	R/W	Пользовательская переменная. Определяется пользователем.	
130	VAR_V30	F	N	R/W	Пользовательская переменная. Определяется пользователем.	
131	VAR_V31	F	N	R/W	Пользовательская переменная. Определяется пользователем.	

Код	Имя	Формат	EPM	Доступ	Описание	Единицы
132	VAR_MOVEDR_DISTANCE	F	N		Расстояние зарегистрированного перемещения. Аналогично MOVEDR	UU
133	VAR_MOVEDR_ DISPLACEMENT	F	N		Смещение зарегистрированного перемещения. Запись в эту переменную выполняет перемещение MOVEDR с величиной заданной в #132	UU
134	VAR_MOVEPR_DISTANCE		N	W	Расстояние зарегистрированного перемещения. Аналогично MOVEPR	UU
135	VAR_MOVEPR_ DISPLACEMENT	F	N	W	Смещение зарегистрированного перемещения. Запись в эту переменную выполняет перемещение MOVEPR с величиной заданной в #134	UU

136	VAR_STOP_MOTION	W	N	W	Останавливает движение: 1 – останавливает движение 0 – нет действия	
137	VAR_START_PROGRAM	W	N	W	Запуск пользовательской программы 1 – запускает программу 0 – нет действия	
138	VAR_VEL_MODE_ON	W	N	W	Включает “профильную” скорость. (Аналогично VELOCITY ON) 0 – нормальное управление 1 – включает режим скорости	
139	VAR_IREF	F	N	R/W	Внутренняя переменная режимов тока или скорости. В режиме скорости В режиме тока	 RPS Amps
140	VAR_NV0	F	N	R/W	Определенная пользователем сетевая переменная (доступна через Ethernet).	
141	VAR_NV1	F	N	R/W	Определенная пользователем сетевая переменная (доступна через Ethernet).	
142	VAR_NV2	F	N	R/W	Определенная пользователем сетевая переменная (доступна через Ethernet).	
143	VAR_NV3	F	N	R/W	Определенная пользователем сетевая переменная (доступна через Ethernet).	
144	VAR_NV4	F	N	R/W	Определенная пользователем сетевая переменная (доступна через Ethernet).	
145	VAR_NV5	F	N	R/W	Определенная пользователем сетевая переменная (доступна через Ethernet).	
146	VAR_NV6	F	N	R/W	Определенная пользователем сетевая переменная (доступна через Ethernet).	
147	VAR_NV7	F	N	R/W	Определенная пользователем сетевая переменная (доступна через Ethernet).	
148	VAR_NV8	F	N	R/W	Определенная пользователем сетевая переменная (доступна через Ethernet).	
149	VAR_NV9	F	N	R/W	Определенная пользователем сетевая переменная (доступна через Ethernet).	
150	VAR_NV10	F	N	R/W	Определенная пользователем сетевая переменная (доступна через Ethernet).	
151	VAR_NV11	F	N	R/W	Определенная пользователем сетевая переменная (доступна через Ethernet).	
152	VAR_NV12	F	N	R/W	Определенная пользователем сетевая переменная (доступна через Ethernet).	
153	VAR_NV13	F	N	R/W	Определенная пользователем сетевая переменная (доступна через Ethernet).	
154	VAR_NV14	F	N	R/W	Определенная пользователем сетевая переменная (доступна через Ethernet).	
155	VAR_NV15	F	N	R/W	Определенная пользователем сетевая переменная (доступна через Ethernet).	
156	VAR_NV16	F	N	R/W	Определенная пользователем сетевая переменная (доступна через Ethernet).	
157	VAR_NV17	F	N	R/W	Определенная пользователем сетевая переменная (доступна через Ethernet).	
158	VAR_NV18	F	N	R/W	Определенная пользователем сетевая переменная (доступна через Ethernet).	

Код	Имя	Формат	EPM	Доступ	Описание	Единицы
159	VAR_NV19	F	N	R/W	Определенная пользователем сетевая переменная (доступна через Ethernet).	
160	VAR_NV20	F	N	R/W	Определенная пользователем сетевая переменная (доступна через Ethernet).	
161	VAR_NV21	F	N	R/W	Определенная пользователем сетевая переменная (доступна через Ethernet).	
162	VAR_NV22	F	N	R/W	Определенная пользователем сетевая переменная (доступна через Ethernet).	
163	VAR_NV23	F	N	R/W	Определенная пользователем сетевая переменная (доступна через Ethernet).	
164	VAR_NV24	F	N	R/W	Определенная пользователем сетевая переменная (доступна через Ethernet).	
165	VAR_NV25	F	N	R/W	Определенная пользователем сетевая переменная (доступна через Ethernet).	

166	VAR_NV26	F	N	R/W	Определенная пользователем сетевая переменная (доступна через Ethernet).	
167	VAR_NV27	F	N	R/W	Определенная пользователем сетевая переменная (доступна через Ethernet).	
168	VAR_NV28	F	N	R/W	Определенная пользователем сетевая переменная (доступна через Ethernet).	
169	VAR_NV29	F	N	R/W	Определенная пользователем сетевая переменная (доступна через Ethernet).	
170	VAR_NV30	F	N	R/W	Определенная пользователем сетевая переменная (доступна через Ethernet).	
171	VAR_NV31	F	N	R/W	Определенная пользователем сетевая переменная (доступна через Ethernet).	
172	VAR_SERIAL_ADDRESS	W	Y	R/W	ID привода (RS485). Диапазон: 0 - 31	
173	VAR_MODBUS_BAUDRATE	W	Y	R/W	Скорость передачи данных для ModBus 1 - 4800 2 - 9600 3 - 19200 4 - 38400 5 - 57600 6 – 115200	
174	VAR_MODBUS_DELAY	W	Y	R/W	Задержка ответа по ModBus в мс	мс
175	VAR_RS485_CONFIG	W	Y	R/W	Конфигурация RS485 0 – нормальный IP через PPP 1 – ModBus	
176	VAR_PPP_BAUDRATE	W	Y	R/W	Скорость передачи данных по RS232/485 1 – 4800 2 – 9600 3 – 19200 4 – 38400 5 – 57600 6 – 115200	
177	VAR_MOVEPS	F	N	W	Аналогична переменной #92, но с использованием S-профиля ускорения/торможения	
178	VAR_MOVEDS	F	N	W	Аналогична переменной #93, но с использованием S-профиля ускорения/торможения	
179	VAR_MDVS_VELOCITY		N	W	Значение скорости для MDV перемещения. Запись в эту переменную помещает MDV сегмент в стек движения, впоследствии вызывая выполнение движения (пока не будет приостановлено кодом #91). Расстояние взято из переменной #94, которая должна быть записана перед записью в эту переменную.	UU
180	VAR_MAXVEL	F	N	R/W	Макс. скорость для профиля движения.	UU/S
181	VAR_ACCEL	F	N	R/W	Значение ускорения для индексации.	UU/S^2
182	VAR_DECEL	F	N	R/W	Значение торможения для индексации.	UU/S^2
183	VAR_QDECEL	F	N	R/W	Значение быстрого торможения	UU/S^2

Код	Имя	Формат	EPM	Доступ	Описание	Единицы
184	VAR_INPOSLIM	W	N	R/W	Предел “По положению”	UU
185	VAR_VEL	F	N	R/W	Адресация скорости для “профилированной” скорости	UU/S
186	VAR_UNITS	F	Y	R/W	Пользовательские единицы	
187	VAR_MECOUNTER	W	N	R/W	Величина счетчика адресации входов A/B	Count
188	VAR_PHCUR	F	N	R	Фазовый ток	A
189	VAR_POS_PULSES	W	N	R/W	Целевое положение в импульсах энкодера.	EC
190	VAR_APOS_PULSES	W	N	R/W	Фактическое положение в импульсах энкодера.	EC
191	VAR_POSERROR_PULSES	W	N	R	Ошибка положения в импульсах энкодера.	EC
192	VAR_CURRENT_VEL_PPS	F	N	R	Фактическая скорость в PPS (импульсов за замер)	PPS
193	VAR_CURRENT_ACCEL_PPSS	F	N	R	Величина фактического ускорения (требуемая величина)	PPSS
194	VAR_IN0_DEBOUNCE	W	Y	R/W	Время дребезга входа в мс Диапазон: 0-1000	мс

195	VAR_IN1_DEBOUNCE	W	Y	R/W	Время дребезга входа в мс Диапазон: 0-1000	мс
196	VAR_IN2_DEBOUNCE	W	Y	R/W	Время дребезга входа в мс Диапазон: 0-1000	мс
197	VAR_IN3_DEBOUNCE	W	Y	R/W	Время дребезга входа в мс Диапазон: 0-1000	мс
198	VAR_IN4_DEBOUNCE	W	Y	R/W	Время дребезга входа в мс Диапазон: 0-1000	мс
199	VAR_IN5_DEBOUNCE	W	Y	R/W	Время дребезга входа в мс Диапазон: 0-1000	мс
200	VAR_IN6_DEBOUNCE	W	Y	R/W	Время дребезга входа в мс Диапазон: 0-1000	мс
201	VAR_IN7_DEBOUNCE	W	Y	R/W	Время дребезга входа в мс Диапазон: 0-1000	мс
202	VAR_IN8_DEBOUNCE	W	Y	R/W	Время дребезга входа в мс Диапазон: 0-1000	мс
203	VAR_IN9_DEBOUNCE	W	Y	R/W	Время дребезга входа в мс Диапазон: 0-1000	мс
204	VAR_IN10_DEBOUNCE	W	Y	R/W	Время дребезга входа в мс Диапазон: 0-1000	мс
205	VAR_IN11_DEBOUNCE	W	Y	R/W	Время дребезга входа в мс Диапазон: 0-1000	мс
206	VAR_OUT0_FUNCTION	W	Y	R/W	Индекс функции выхода	
207	VAR_OUT1_FUNCTION	W	Y	R/W	Индекс функции выхода	
208	VAR_OUT2_FUNCTION	W	Y	R/W	Индекс функции выхода	
209	VAR_OUT3_FUNCTION	W	Y	R/W	Индекс функции выхода	
210	VAR_HALLCODE	W	N	R	Текущий код Холла Bit 0 - Hall 1 Bit 1 - Hall 2 Bit 2 - Hall 3	
211	VAR_ENCODER	W	N	R	Текущее значение первого энкодера	EC
212	VAR_RPOS_PULSES	W	N	R	Регистрационное положение	EC
213	VAR_RPOS	F	N	R	Регистрационное положение	EU
214	VAR_POS	F	N	R/W	Целевое положение	UU
215	VAR_APOS	F	N	R/W	Фактическое положение	UU
216	VAR_POSERROR	W	N	R	Ошибка положения	EC
217	VAR_CURRENT_VEL	F	N	R	Фактическая скорость (требуемая величина)	UU/S
218	VAR_CURRENT_ACCEL	F	N	R	Фактическое ускорение (требуемая величина)	UU/S^2

Код	Имя	Формат	EPM	Доступ	Описание	Единицы
219	VAR_TPOS_ADVANCE	W	N	W	Опережение целевого положения. Каждая запись в эту переменную прибавляет значение к точке суммирования целевого положения. Величина добавляется единожды за запись. Эта переменная полезна, когда сигналы Masterэнкодеоа управляют циклом и пытаются корректировать фазу. Значение в единицах отсчета энкодера.	EC
220	VAR_IOINDEX	W	N	R/W	Такая же, как переменная индекса в пользовательской программе. См. “INDEX” в разделе языковые таблицы данного руководства.	
221	VAR_PSLIMIT_PULSES	W	Y	R/W	Положительная величина концевого выключателя ПО в единицах отсчета энкодера.	EC
222	VAR_NSLIMIT_PULSES	W	Y	R/W	Отрицательная величина концевого выключателя софта в единицах отсчета энкодера.	EC

223	VAR_ SLS_MODE	W	Y	R/W	Код действия концевого выключателя ПО 0 - нет действия 1 - ошибка 2 - Остановка и ошибка (Когда цикл управляется только генератором траектории. Со всеми остальными источниками то же действие, что и 1)	
224	VAR_PSLIMIT	F	Y	R/W	Аналогична переменной 221, но значение выражается в пользовательских единицах.	UU
225	VAR_NSLIMIT	F	Y	R/W	Аналогична переменной 222, но значение выражается в пользовательских единицах.	UU

AC Technology Corporation
member of the Lenze Group
630 Douglas Street
Uxbridge, MA 01569
Telephone: (508) 278-9100
Facsimile: (508) 278-7873

Официальный представитель в России
ООО «Промситех»
Адрес: Москва, 15-я
Парковая ул., д. 5, комн. 501
Тел.: (495) 788-1262
Факс.: (495) 463-8981
www.promsytex.ru
www.prst.ru
www.hubner.ru