Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение высшего образования
«Северо-Кавказский федеральный университет»

Выпускная квалификационная работа №_____

Студента Тарана Ивана Олеговича

Направления подготовки 13.04.02 «Электроэнергетика и электротехника»

Группы ЭЭТ-м-о-18-2

Защищена «25» июня 2020 г.

Тема: Совершенствование алгоритмов цифровых защит электродвигателей

Распоряжение об утверждении темы ВКР от «9» января 2020 г. № 01-р/14-03

Пояснительная записка_____страниц

Чертежи_____листов

Подпись и Ф.И.О. лица, принявшего документы на кафедру_____

Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт Инженерный
Кафедра Автоматизированных электроэнергетических систем и электроснабжения

Утверждена распоряжением по институту от «9» января 2020 г. № 01-р/14-03

Допущена к защите
«_____» _____ 2020 г.

Зав. кафедрой АЭСиЭ
профессор, доктор технических наук
Кононов Юрий Григорьевич

_____
(подпись зав. кафедрой)

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

Совершенствование алгоритмов цифровых защит электродвигателей
(название дипломной работы / дипломного проекта)

Рецензент:
Ярош Виктор Алексеевич
(ФИО)
доцент кафедры Электроснабжения и эксплуатации электрооборудования, Электроэнергетический факультет, ФГБОУ ВО СГАУ, канд. техн. наук
(ученая степень, звание, должность)

Нормоконтролер:
Мамаев Виктор Александрович
(ФИО)
канд. техн. наук, доцент кафедры АЭСиЭ
(ученая степень, звание, должность)

_____
(Подпись)

Дата защиты
«25» июня 2020 г.
Оценка _____

Выполнил (а):
Таран Иван Олегович
(ФИО)
студент(ка) 2 **курса, группы** ЭЭТ-м-о-18-2
направления подготовки 13.04.02 «Электроэнергетика и электротехника»
Направленность (профиль)
Энергосбережение и энергоэффективность
очной **формы обучения**

_____
(Подпись)

Руководитель:
Мамаев Виктор Александрович
(ФИО)
канд. техн. наук, доцент кафедры АЭСиЭ
(ученая степень, звание, должность)

_____
(Подпись)

Ставрополь, 2020 г.

Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт Инженерный
Кафедра Автоматизированных электроэнергетических систем и электроснабжения
Направление подготовки «Электроэнергетика и электротехника»
Направленность (профиль) Энергосбережение и энергоэффективность

«УТВЕРЖДАЮ»
Зав. кафедрой
_____Ю.Г. Кононов
подпись, инициалы, фамилия
«9» января 2020 г

ЗАДАНИЕ НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ (ДИПЛОМНУЮ РАБОТУ)

Студент Таран Иван Олегович группа ЭЭТ-м-о-18-2
фамилия, имя, отчество

1.Тема Совершенствование алгоритмов цифровых защит электродвигателей
Утверждена распоряжением по институту от «9» января 2020 г. № 01-р/14-03
2.Срок представления проекта к защите «25» июня 2019 г.
3.Исходные данные для проектирования Научно-техническая литература, описывающая проблемы функционирования цифровых защит асинхронных электродвигателей
Содержание пояснительной записки:
4.1 Анализ современного состояния проблемы функционирования защит электродвигателей
4.2 Алгоритмы измерительных органов цифровых защит
4.3 Инструментарий математического моделирования нормальных и аварийных режимов работы двигателей
4.4 Концепция функционирования защит электродвигателей на основе оценки мгновенного значения сопротивления
Приложение листинг скомпилированного кода модели PSCAD в компиляторе GFortran 95
5 Перечень графического материала (с точным указанием обязательных чертежей)
Цели и задачи; Обзор защит электродвигателей; Алгоритмы измерительных органов ЦРЗ; Модель участка сети с АД и защитой; Программное обеспечение; Концепция функционирования РЗ АД.
Дата выдачи задания 17.01.2020

Руководитель проекта _____ В.А. Мамаев
подпись *инициалы, фамилия*

Консультанты по разделам _____ П.А. Звада
подпись *инициалы, фамилия*

Задание принял к исполнению _____ И.О. Таран
подпись, дата *инициалы, фамилия*

Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт Инженерный
Кафедра Автоматизированных электроэнергетических систем и электроснабжения
Направление подготовки 13.04.02 «Электроэнергетика и электротехника»
Направленность (профиль) Энергосбережение и энергоэффективность

КАЛЕНДАРНЫЙ ПЛАН

Фамилия, имя, отчество (полностью) Таран Иван Олегович
Тема ВКР Совершенствование алгоритмов цифровых защит электродвигателей
Руководитель Мамаев Виктор Александрович
Консультанты: Звада Павел Александрович

№	Наименование этапов выполнения выпускной квалификационной работы	Срок выполнения работы	Примечание
1	Выбор темы выпускной квалификационной работы. Получение задания ВКР. Сбор информации по теме ВКР.	**28.01.2020 г.**	Начало проектирования – 28.01.2020 г
2	Анализ современного состояния проблемы функционирования защит электродвигателей	15.02.2020 г.	Готовность раздела 1 настоящей работы
3	Алгоритмы измерительных органов цифровых защит	15.03.2020 г.	Готовность раздела 2 настоящей работы
4	Инструментарий математического моделирования нормальных и аварийных режимов работы двигателей. Концепция функционирования защит электродвигателей на основе оценки мгновенного значения сопротивления	**20.05.2020 г**	Готовность раздела 3, 4 настоящей работы **Контрольная проверка**
5	Графическая часть	**16.06.2020 г.**	Готовность ВКР – **Контрольная проверка**
6	Подготовка документов ВКР к защите		–
7	Предзащита	**17.06.2020 г**	**–**
8	Защита ВКР	**25.06.2020 г**	**–**

Руководитель _____ В.А. Мамаев

подпись. Ф.И.О.

Зав. кафедрой _____ Ю.Г. Кононов

подпись. Ф.И.О.

«19» января 2020 г.

Выпускная квалификационная работа содержит 93 стр., 22 илл., 0 табл.

Ключевые слова: асинхронный двигатель, цифровые защиты, алгоритм адаптивный, селективность защит.

Настоящая работа посвящена исследованию и разработке алгоритма защиты электродвигателей в условиях ненормальных и аварийных режимах работы.

Направления исследований.

1. Анализ современного состояния проблемы функционирования защит электродвигателей.

2. Анализа состояния вопроса повышения селективности защит двигателей в трехфазных электрических сетях на основе современных цифровых устройств релейной защиты.

3. Поиск и разработка новых алгоритмов функционирования существующих терминалов защиты электродвигателей.

4. Определения требований к применению рассматриваемого алгоритма функционирования защит двигателей.

Содержание

Введение

Актуальность работы: Релейная защита и противоаварийная автоматика (РЗА) осуществляют автоматическую ликвидацию повреждений и ненормальных режимов в электрической части энергосистем и являются важнейшей системой, обеспечивающей их надежную и устойчивую работу [1]. Рост нагрузок, увеличение протяжённости линий электропередачи, ужесточение требований к устойчивости работы энергосистем усложняют условия работы релейной защиты. В то же время повышаются требования к эффективности её функционирования, поэтому идёт непрерывный процесс развития техники релейной защиты, направленный на создание более совершенных устройств, отвечающих требованиям современной электроэнергетики.

Системы РЗА играют значительную роль в обеспечении управляемости и надёжности работы энергосистем. До настоящего времени основную долю находящейся в эксплуатации аппаратуры РЗА составляют отечественные электромеханические и микроэлектронные устройства. При этом парк технических средств РЗА физически и морально стареет. Указанные обстоятельства диктуют необходимость проведения реконструкции и модернизации РЗА в направлении применения аппаратных и программных средств, основанных на использовании микропроцессорных систем, интегрированных в АСУ ТП.

Целью работы является анализ и разработка алгоритма защиты двигателей в трехфазных схемах.

Для достижения поставленной цели решались следующие задачи:

- анализ современного состояния проблемы функционирования защит электродвигателей;

- обзор существующих защит электродвигателей и алгоритмов их реализации в различных ненормальных и аварийных режимах работы электрод двигателей;

- разработка алгоритма адаптивной зашиты электродвигателя на базе расчета мгновенного сопротивления на зажимах его питания;

- математическое моделирование электродвигателя и предлагаемого алгоритма функционирования защиты;

- разработка программы и алгоритма функционирования цифровой защиты устройств на его основе.

При решении поставленных задач использовались: элементы теории электрических цепей, методы математического моделирования, методы математического эксперимента. Экспериментально-расчетные исследования выполнены с использованием средств математического моделирования переходных режимов PSCAD и Multisim.

Научная новизна работы заключается в следующем:

- разработана концепция функционирования защит электродвигателей основанная на оценке мгновенного значения сопротивления;

- разработаны программы и алгоритма функционирования цифровой защиты устройств на его основе;

- предлагаемый алгоритм функционирования цифровых защит позволяет повысить селективность и чувствительность защит электродвигателей.

На защиту выносятся положения, составляющие научную новизну, а также результаты анализа современного состояния проблемы функционирования защит электродвигателей.

Реализации и внедрение результатов работы:

Основные результаты работы планируются к внедрению в учебный процесс при изучении дисциплины «Цифровые устройства релейной защиты» по направлению подготовки 13.04.02 Электроэнергетика и электротехника.

Личный вклад автора заключается в обработке результатов математического эксперимента, в разработке математической модели и алгоритма защиты электродвигателя, а также в разработке программной реализации алгоритма защиты электродвигателя.

Асинхронные электродвигатели составляют основу наиболее массовых приемников электрической энергии в современных электрических системах Стремление максимально полно использовать их нагрузочные возможности в условиях все более динамичных и разнообразных режимов работы привело к повышению риска возникновения внутренних повреждений Ежегодно 10-20% общего парка электродвигателей выходит из строя Каждый день возникают несколько тысяч повреждений в электродвигателях, которые вызывают нарушения технологических процессов, создают опасные возмущения в электрических системах и приводят к угрожающему росту опасности их развития в крупные аварии с катастрофическими последствиями

Традиционные средства релейной защиты электродвигателей развивались исторически параллельно с электромеханическими измерительными механизмами измерительных приборов па базе общей теории применительно к стационарным входным сигналам Поэтому большинство алгоритмов традиционных защит основано на контроле интегральных (действующих или средних) значений токов и напряжений Этот стационарный подход требует длительного наблюдения за процессами в аварийных ситуациях для принятия правильною решения о состоянии контролируемого объекта

Дефицит времени, отводимого для выявления повреждений в современных электрических системах, вызывает необходимость выполнять анализ состояния контролируемого объекта в условиях не завершившихся переходных процессов При этом требования к средствам защиты электродвигателей повышаются, и традиционные решения часто оказываются не приемлемыми Требуется новый нестационарный подход к построению

средств релейной защиты электродвигателей, основанный на представлении контролируемых объектов более точными математическими моделями, пригодными для реализации динамического контроля

В этой связи совершенствование зашит наиболее массовых приемников электрической энергии в электрических системах на основе нового нестационарного подхода играет важную роль в достижении высокой надежности электроснабжения и представляет собой крупную и актуальную научно-техническую задачу

1 АНАЛИЗ СОВРЕМЕННОГО СОСТОЯНИЯ ПРОБЛЕМЫ ФУНКЦИОНИРОВАНИЯ ЗАЩИТ ЭЛЕКТРОДВИГАТЕЛЕЙ

1.1 Обзор проблемы функционирования защит АД

Координация защиты имеет важное значение в распределительной сети с растущей долей распределенной генерации различных типов генерирующих установок, фотоэлектрических электростанций, электростанций на биомассе, биогазовых установок и ветряных электростанций, и они были рассмотрены в [1-3]. Практические рекомендации и обзору координации защиты в радиальных распределительных сетях с реле максимального тока приведены в [4-6]. В прошлом координация защиты в высоковольтных сетях передачи и в промышленности осуществлялась вручную с использованием правил временных диаграмм и настроечных таблиц реле тока устройства. Трудоемкий процесс занимал много времени и приводил к ошибкам в настройке защитных реле, которые вызывали ненужные перебои с потребителем в распределительных и промышленных технологических процессах, вызывая дальнейшее прямое повреждение оборудования и косвенные затраты в технологическом процессе из-за нереализованных производство.

С развитием электроники защиты все больше строятся с использованием микроэлектронных компонентов и в современном исполнении представляют собой цифровые устройства (терминалы) релейной защиты, несущие в своем программном и аппаратном обеспечении дополнительные функции автоматики и сигнализации.

Принципы построения устройств релейной защиты, в том числе и микропроцессорных, весьма разнообразны. Однако в подавляющем большинстве эти устройства являются автономными и выполняются с использованием электрических воздействующих величин – токов и напряжений промышленной частоты защищаемых элементов системы. Иногда, в качестве дополнительной информации, могут использоваться некоторые физические явления неэлектрического характера, сопровождающие короткие замыкания (КЗ) и ненормальные режимы защищаемого элемента электрической системы. В частности, может

использоваться световая вспышка (при дуговых КЗ), изменение скорости выделения газов (газовые реле), повышение температуры элементов защищаемого объекта, вибрация электрической машины и т.д.

В общем случае устройства релейной защиты имеют две основные части – измерительную и логическую (рисунок 1.1).

Измерительная часть (U3), включающая релейный измерительный орган (РИО), непрерывно контролирует состояние защищаемого объекта и определяет условия срабатывания в соответствии со значениями входных величин. Таковыми являются вторичные токи измерительных трансформаторов тока (TA1, …) и вторичные напряжения измерительных трансформаторов напряжения (TV1, ...). Логическая часть (U5) формирует управляющие воздействия в зависимости от комбинации и последовательности поступления на неё сигналов от РИО. Логическая часть действует на выключатель не непосредственно, а через исполнительные органы (электромеханические реле KL1, …).

Для защит с абсолютной селективностью измерительная и логическая части могут получать также информацию с другой стороны защищаемого элемента (другой электроустановки) по специальным каналам связи.

Сигнальные органы (H1) информируют о срабатывании комплекта защиты в целом, иногда и отдельных её частей. Для питания измерительных, исполнительных и сигнальных органов, логической части предусматривается источник питания.

На вход устройства релейной защиты непрерывно подаются аналоговые величины, пропорциональные значениям напряжений и токов защищаемого объекта.

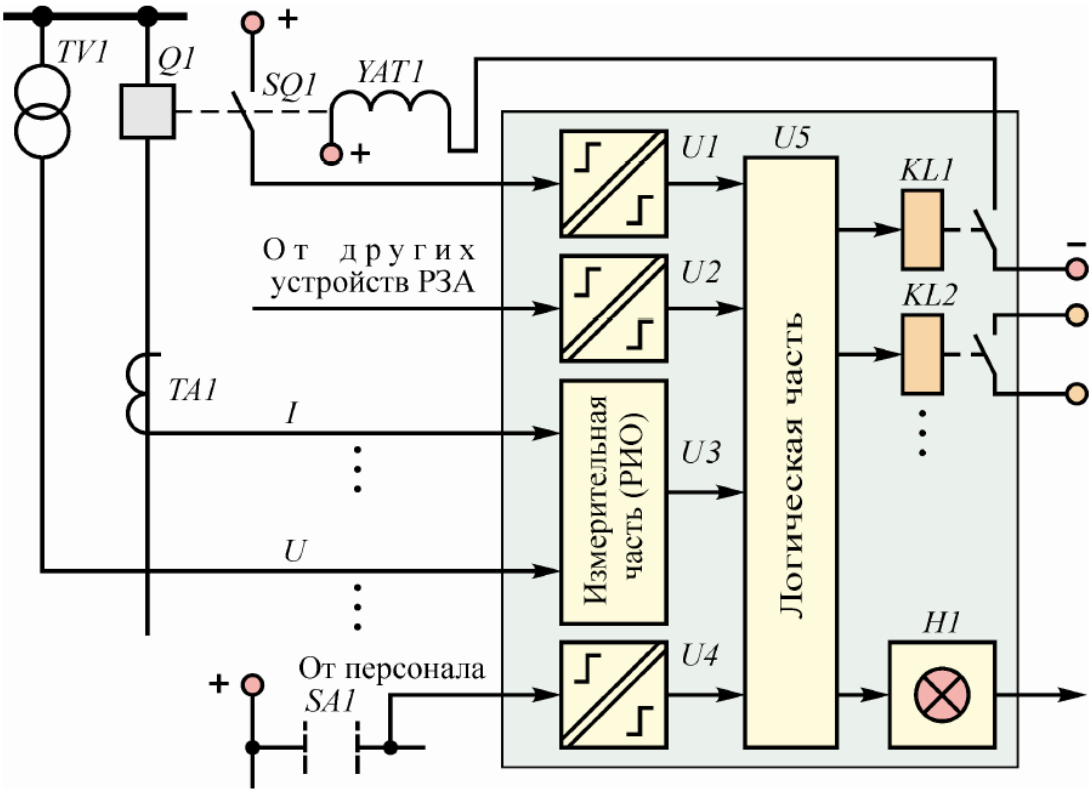


Рисунок 1.1 – Обобщённая структура устройства релейной защиты

На рисунке 1.1 элементы имеют следующие обозначения: TV1 – измерительный трансформатор напряжения; TA1 – измерительный трансформатор тока; Q1 – выключатель; SQ1 – контакт выключателя; YAT1 – катушка отключения выключателя; SA1 – ключ управления.

Определённый выходной сигнал (например, сигнал "Отключить") должен быть автоматически получен лишь в том случае, если входные величины удовлетворяют некоторым условиям. Наличие этих условий фиксируется РИО, который подает сигнал одного вида, если определённое условие удовлетворяется, и иной сигнал, если оно не удовлетворяется.

Таким образом, измерительный орган преобразовывает аналоговый сигнал на входе, например, напряжение, в дискретный сигнал на выходе.

К измерительным органам относятся:

– реле тока (напряжения), действие которого зависит от мгновенного или интегрального значения входной воздействующей величины;

– реле направления мощности, зона действия которого определяется

фазовым углом между векторами напряжения и тока;

– реле сопротивления, действие которого определяется от векторного отношения напряжения и тока;

– реле частоты, действие которого зависит от частоты и др.

Входные сигналы РИО несут определенную информацию о режиме работы защищаемого объекта. Для выявления аварийного режима из подведенных напряжений и токов с помощью измерительной части могут формироваться специальные непрерывные величины, характерные только для аварии. Так, из трёх фазных напряжений и токов выделяются симметричные составляющие обратной и нулевой последовательностей, на основе анализа которых можно определить вид КЗ и его "место".

В схеме сравнения обрабатываются подготовленные измерительной схемой непрерывные величины. В зависимости от соотношения этих величин на выходе схемы сравнения появляется или не появляется определенный стандартный сигнал. Именно схема сравнения преобразует непрерывные сигналы на входе в дискретный сигнал на выходе. Например, функциональная схема алгоритма максимальной токовой защиты МТЗ, реализующейся в логической части устройства показана на рисунке 1.2. Данный вариант логики как правило, реализуется программным путем.

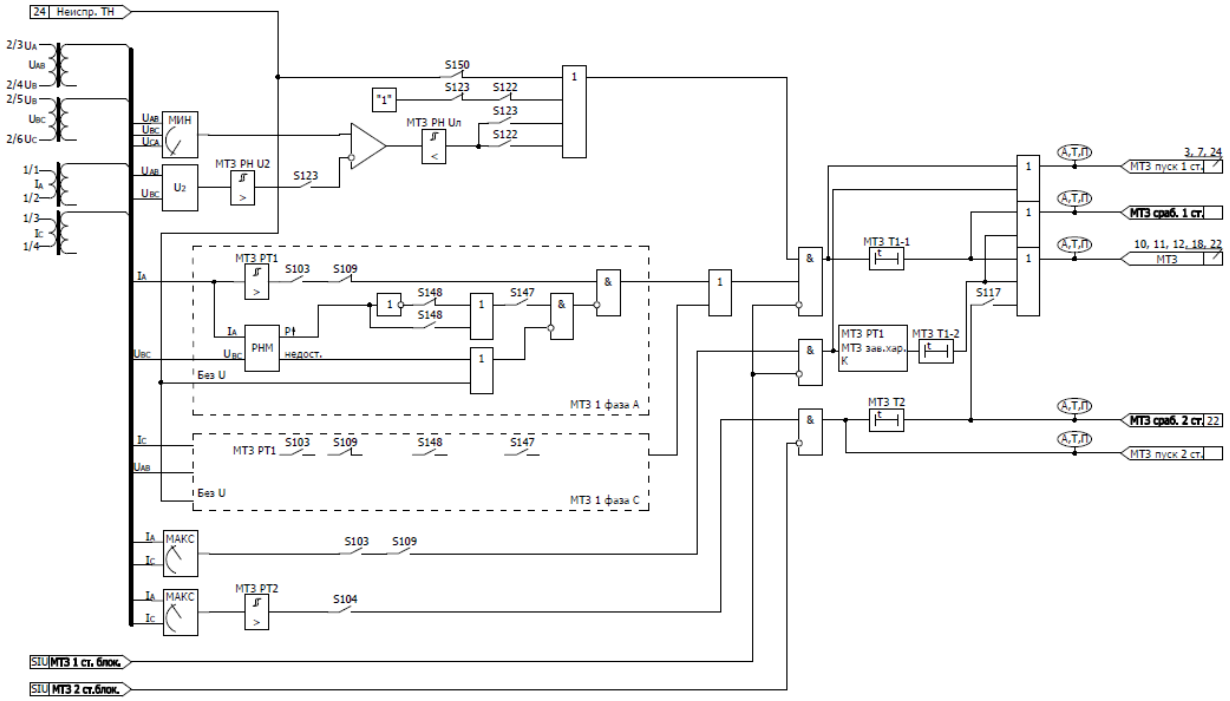


Рисунок 1.2 – Функциональная схема алгоритма максимальной токовой защиты

1.2 Цифровые защиты электродвигателей

Для обеспечения непрерывных процессов функционирования требуются системы управления двигателем и его защиты. Необходимо рассмотреть различные варианты проектирования защит в том числе и защиты от замыканий на землю в низковольтных сетях. Системы управления и защиты двигателя применяются как для сетей с изолированной нейтралью, с гухозаземленной и заземленные через сопротивление. Типичные методы защиты от замыканий на землю в центрах управления низковольтными двигателями и общие схемные проблемы защиты заземления, включаю: системы обнаружения импульсов напряжения, напряжения нулевой последовательности и методы, основанные на заземлении через высокое сопротивление для быстрого обнаружения неисправностей.

В основном все микропроцессорные устройства релейной защиты двигателей устанавливаются в релейных отсеках КРУ, КРУН и КСО, на

панелях и в шкафах в релейных залах и пультах управления электростанций и подстанций 6–35 кВ и являются комбинированными микропроцессорными терминалами релейной защиты и автоматики.

Применение в устройстве модульной мультипроцессорной архитектуры наряду с современными технологиями поверхностного монтажа обеспечивают высокую надежность, большую вычислительную мощность и быстродействие, а также высокую точность измерения электрических величин и временных интервалов, что дает возможность снизить ступени селективности и повысить чувствительность терминала.

Возможны варианты применения защиты элементов распределительных сетей как в самостоятельном функционировании устройства, так и совместно с другими устройствами РЗА (например, дуговой защитой, защитой от однофазных замыканий на землю, защитой шин и т.д.).

Все устройства обеспечивают как правило следующий перечень эксплуатационных возможностей:

выполнение функций защит, автоматики и управления, определенных ПУЭ и ПТЭ;

задание внутренней конфигурации (ввод/вывод защит и автоматики, выбор защитных характеристик и т.д.);

ввод и хранение уставок защит и автоматики;

контроль и индикацию положения выключателя, а также контроль исправности его цепей управления;

определение вида повреждения;

передачу параметров аварии, ввод и изменение уставок по линии связи;

непрерывный оперативный контроль работоспособности (самодиагностику) в течение всего времени работы;

блокировку всех выходов при неисправности устройства для исключения ложных срабатываний;

получение дискретных сигналов управления и блокировок, выдачу команд управления, аварийной и предупредительной сигнализации;

гальваническую развязку всех входов и выходов, включая питание, для обеспечения высокой помехозащищенности;

высокое сопротивление и прочность изоляции входов и выходов относительно корпуса и между собой для повышения устойчивости устройства к перенапряжениям, возникающим во вторичных цепях КРУ.

Функции защиты, выполняемые устройством:

двухступенчатая дифференциальная токовая защита двигателя (токовая отсечка и защита с торможением от сквозного тока и блокировкой при броске тока намагничивания);

трехступенчатая максимальная токовая защита (МТЗ) от междуфазных повреждений с контролем двух или трех фазных токов (любая ступень может быть выполнена направленной, а также может иметь комбинированный пуск по напряжению);

защита от обрыва фазы питающего фидера (ЗОФ);

защита от однофазных замыканий на землю (ОЗЗ) по сумме высших гармоник;

защита от однофазных замыканий на землю по току основной частоты (может быть выполнена направленной);

защита синхронных двигателей от асинхронного хода в ступени МТЗ-2;

минимальная токовая защита (ЗМТ);

защита минимального напряжения (ЗМН);

защита от перегрева электродвигателя;

защита от затянутого пуска (ЗЗП);

защита от блокировки ротора (ЗБР);

защита обратной мощности (ЗОМ);

выдача сигнала пуска МТЗ для организации логической защиты шин.

Различают повреждения, связанные с пробоем изоляции в статорной обмотке и требующие максимально быстрого отключения двигателя, а также разного рода ненормальные режимы, при которых действие защиты возможно

с выдержкой времени, а в некоторых случаях может ограничиваться сигнализацией.

Наиболее опасными являются многофазные и витковые короткие замыкания в обмотке статора, приводящие к разрушению двигателя. Для двигателей с напряжением от 6 кВ и выше работающих в сети с изолированной нейтралью возможно возникновение двойных КЗ, для которых одна точка повреждения может находиться во внешней сети, а вторая в двигателе. Такие повреждения также являются опасными для двигателя и требуют отключения с минимально достижимой выдержкой времени. Достаточно отключения одной точки двойного КЗ, которое переводит двойное короткое замыкание в однофазное замыкание с малым током замыкания на землю.

Для защиты двигателей от многофазных КЗ согласно ПУЭ [1] в случаях, когда не используются предохранители, должна применяться токовая защита мгновенного действия – отсечка. На двигателях мощностью менее 2000 кВт отсечка выполняется с одним реле тока, включенным на разность токов двух фаз, для двигателей мощностью от 2000 до 5000 кВт используется двухрелейная отсечка при условии, что на двигателях установлена отдельная защита от двойных, а также однофазных замыканий на землю с действием на отключение. При отсутствии указанных защит токовая отсечка выполняется трехрелейной с контролем трех фазных токов.

Защита блоков двигатель-трансформатор при мощности двигателя менее 2000 кВт выполняется токовой отсечкой.

Для двигателей мощностью более 5000 кВт, а также в случае недостаточной чувствительности токовой отсечки для двигателей меньшей мощности, необходимо использование продольной дифференциальной токовой защиты. В терминалах серии «Сириус-Д» такая защита не предусмотрена, что определяет область ее использования. Соответственно, дифференциальные токовые защиты далее не рассматриваются.

Микропроцессорные терминалы релейной защиты в подавляющем большинстве выполняются с возможностью контроля всех трех фазных токов,

что позволяет выполнить токовую отсечку по трехрелейной схеме на двигателях любой мощности, даже если Правила допускают упрощенное исполнение защиты. Контроль первичным измерительным трансформатором тока только двух фаз не является препятствием для выполнения трехрелейной схемы, поскольку ток третьей фазы «восстанавливается» алгоритмом защиты путем суммирования токов двух контролируемых фаз.

Токовые защиты в составе терминалов, как правило, выполняются трехступенчатыми. Первая ступень защиты реализует функции токовой отсечки от КЗ, последняя ступень может быть использована в качестве защиты от перегрузки, при этом, вторая ступень в случае защиты двигателей оказывается избыточной и не вводится в работу.

Например, терминал «Сириус-21-Д» разработан как универсальное устройство с возможностью защиты не только двигателей, но и иных элементов сети напряжением до 35 кВ. Поэтому в его состав, в том числе, введены измерительные органы, не используемые при выполнении защиты двигателей. Применительно к ступенчатой токовой защите предусмотрены следующие элементы, полезные при защите сетей, но не используемые для защиты двигателей:

обеспечение направленности с помощью измерительного органа направления мощности;

пусковые органы напряжения, включая комбинированный пуск по напряжению;

зависимые время-токовые характеристики срабатывания.

1.3 Алгоритмы токовой защиты

Приводится общее описание алгоритмов, в котором отражены только основные положения, общие для большинства терминалов. В зависимости от модификации терминала конкретные алгоритмы, реализующие функции защиты и автоматики, имеют некоторые не отмеченные здесь особенности, в

связи с чем для более детального изучения алгоритмов рекомендуется обратиться к Руководству по эксплуатации соответствующего терминала.

Ступенчатая токовая защита контролирует фазные токи и предназначена для действия при междуфазных КЗ. Защита выполнена в трехфазном варианте, т.е. с контролем трех фазных токов, при наличии ТТ в двух фазах ток третьей фазы «восстанавливается» по выражению.

Ступени токовой защиты в зависимости от их назначения и, соответственно, расчетных условий для выбора тока срабатывания, традиционно разделяют на «отсечку» без выдержки и с выдержкой времени (первая и вторая ступени) и «максимальную токовую защиту» (последняя ступень). Такое деление устоялось для традиционных защит аналогового исполнения, когда назначение ступени было фиксировано схемным исполнением ступени в составе панели. В микропроцессорных защитах назначение ступени не фиксировано и может изменяться в зависимости от условий применения терминала защиты. Терминологическое разделение ступеней на «отсечку» и «максимальную токовую защиту» оказывается достаточно условным и в ряде случаев неоднозначным. Применительно к микропроцессорным защитам все более часто используется термин «максимальная токовая защита» (МТЗ) для любых ступеней, в том числе, и первых, ранее именовавшихся «отсечкой». Далее под МТЗ понимается любая ступень защиты, реагирующая на увеличение тока в условиях срабатывания.

Максимальная токовая защита может быть выполнена в простейшем варианте, не предусматривающем контроль напряжений, в связи с чем она не обладает направленностью и не содержит пусковые органы напряжения

В канале каждой ступени включена логическая схема «И», обеспечивающая настройку ступени, что позволяет: ввести ступень в действие: замыкается программный ключ «ввод», что обеспечивает постоянную подачу логического сигнала «1» на соответствующий ввод схемы И; поставить действие ступени под контроль от пускового органа напряжения. При верхнем положении ключа ПОН пуск ступени от ПОН не предусмотрен,

на вход схемы И постоянно поступает логический сигнал «1», при нижнем положении ключа логический сигнал «1» поступает от ПОН только при его срабатывании; ввести направленность действия ступени ключом «напр.» с контролем направления мощности от органа направления мощности (ОНМ).

Выводы к 1-й главе

В данной главе проведен анализ современного состояния проблемы функционирования защит электродвигателей. Традиционные средства релейной защиты электродвигателей развивались исторически параллельно с электромеханическими измерительными механизмами измерительных приборов па базе общей теории применительно к стационарным входным сигналам Поэтому большинство алгоритмов традиционных защит основано на контроле интегральных (действующих или средних) значений токов и напряжений Этот стационарный подход требует длительного наблюдения за процессами в аварийных ситуациях для принятия правильною решения о состоянии контролируемого объекта

Установлено, что наиболее опасными являются многофазные и витковые короткие замыкания в обмотке статора, приводящие к разрушению двигателя. Для двигателей с напряжением от 6 кВ и выше работающих в сети с изолированной нейтралью возможно возникновение двойных КЗ, для которых одна точка повреждения может находиться во внешней сети, а вторая в двигателе. Такие повреждения также являются опасными для двигателя и требуют отключения с минимально достижимой выдержкой времениОбзор проблемы функционирования защит АД

В основном все микропроцессорные устройства релейной защиты двигателей устанавливаются в релейных отсеках КРУ, КРУН и КСО, на панелях и в шкафах в релейных залах и пультах управления электростанций и подстанций 6–35 кВ и являются комбинированными микропроцессорными терминалами релейной защиты и автоматики

2. АЛГОРИТМЫ ИЗМЕРИТЕЛЬНЫХ ОРГАНОВ ЦИФРОВЫХ ЗАЩИТ

2. Алгоритмы измерительных органов цифровых защит

2.1 Общие положения

В аналоговых устройствах защиты осуществляется измерение параметров контролируемых синусоидальных величин электрической системы путем воздействия непрерывных аналоговых сигналов, зависящих от входных токов и напряжений, на физическую систему (индукционную, электромагнитную, полупроводниковую и т.д.). По результатам этого воздействия оценивается нахождение контролируемых параметров в заданной области, на основании чего принимается решение, соответствующее алгоритму работы защиты. Например, срабатывание измерительного органа максимальной токовой защиты, дающего команду реле времени на отсчёт выдержки времени, соответствует при синусоидальном токе превышению его амплитудой заранее заданного значения (действующее значение превышает уставку).

Основные операции в ЦИО производятся не с аналоговыми сигналами, а с чередующимися во времени с периодом $T_д$ последовательностями двоичных чисел, соответствующими дискретным сигналам, полученным при цифровой обработке входных аналоговых сигналов.

Алгоритмом ЦИО называется последовательность операций с цифровыми отсчётами (выборками), зависящими от входных аналоговых сигналов, обеспечивающая измерение контролируемых параметров электрической системы или оценку их нахождения в заданной области.

Все рассматриваемые алгоритмы обеспечивают необходимые характеристики ЦИО при условии, что производятся операции с цифровыми последовательностями вида (2.1), обусловленными цифровой обработкой только синусоидальных сигналов частоты 50 Гц (w1 = 314 с-1), т.е. составляющие других частот практически отсутствуют.

В основу рассматриваемых алгоритмов положена возможность определения амплитуды U_m и фазы у входного синусоидального сигнала

$u(t) = U_m\sin(\omega t + \psi)$, изменяющегося с частотой $\omega = \omega_1$, по известным в произвольный момент времени t мгновенным значениям сигнала $u(t)$ и его первой $du/dt = \omega U_m cos(\omega t + \psi)$ и второй $d^2u/dt^2 = -\omega^2 U_m\sin(\omega t + \psi)$ производных.

2.2. Вычисление векторов по мгновенным значениям синусоидальных величин

Синусоидальный сигнал $u(t) = U_m sin(\omega t + \psi)$ в векторном представлении записывается как $U_{me}\, j(\omega t + \psi) = U_{me}\, j\omega t = U_x + jU_y$, чему соответствует геометрическое отображение, показанное на рисунке 2.1, а. Амплитуда синусоидальной величины Um и фаза у, как это следует из известных тригонометрических соотношений, иллюстрируемых рисунком 2.1, а, могут быть вычислены по формулам

$$U_m = \sqrt{u^2(t) + \left(\frac{1}{\omega}\frac{du}{dt}\right)^2}\,; \psi = arctg\frac{\omega u(t)}{du/dt} - \omega t\,. \qquad (2.1)$$

Этим формулам соответствует векторное соотношение

$$\underline{U}(t) = \underline{U}_m e^{j\omega t} = \frac{1}{\omega}\frac{du}{dt} + ju(t) \qquad (2.2)$$

Из уравнений (2.1, 2.2) относительно легко получаются алгоритмы, определяющие вектор $U(nT_д) = U_{me}\, ^{j\omega nT_д}$, соответствующий цифровой форме дискретизированного синусоидального сигнала $u(nT_д) = U_m sin(\omega nT_д + \psi)$.

Если для вычисления первой производной использовать выборки $u(nT_д)$ и $u[(n\text{-}1)T_д]$ дискретизированного синусоидального сигнала (рисунок 2.1, б), то можно принять

$$u(t) = u(nT_д);\ \frac{du}{dt} \approx \frac{1}{T_д}\left\{u(nT_д) - u\left[(n-1)T_д\right]\right\}, \qquad (2.3)$$

откуда с учётом (2.2) и соотношения $\omega T_д = \omega T/N = 2\pi/N$, где T – период частоты напряжения $u(t)$, а $N = T/T_д$ – число выборок за период, получается алгоритм вычисления вектора $U(nT_д)$:

$$\underline{U}(nT_д) = \left(\frac{N}{2\pi} + j\right) u(nT_д) - \frac{N}{2\pi} u\left[(n-1)T_д\right] \qquad (6)$$

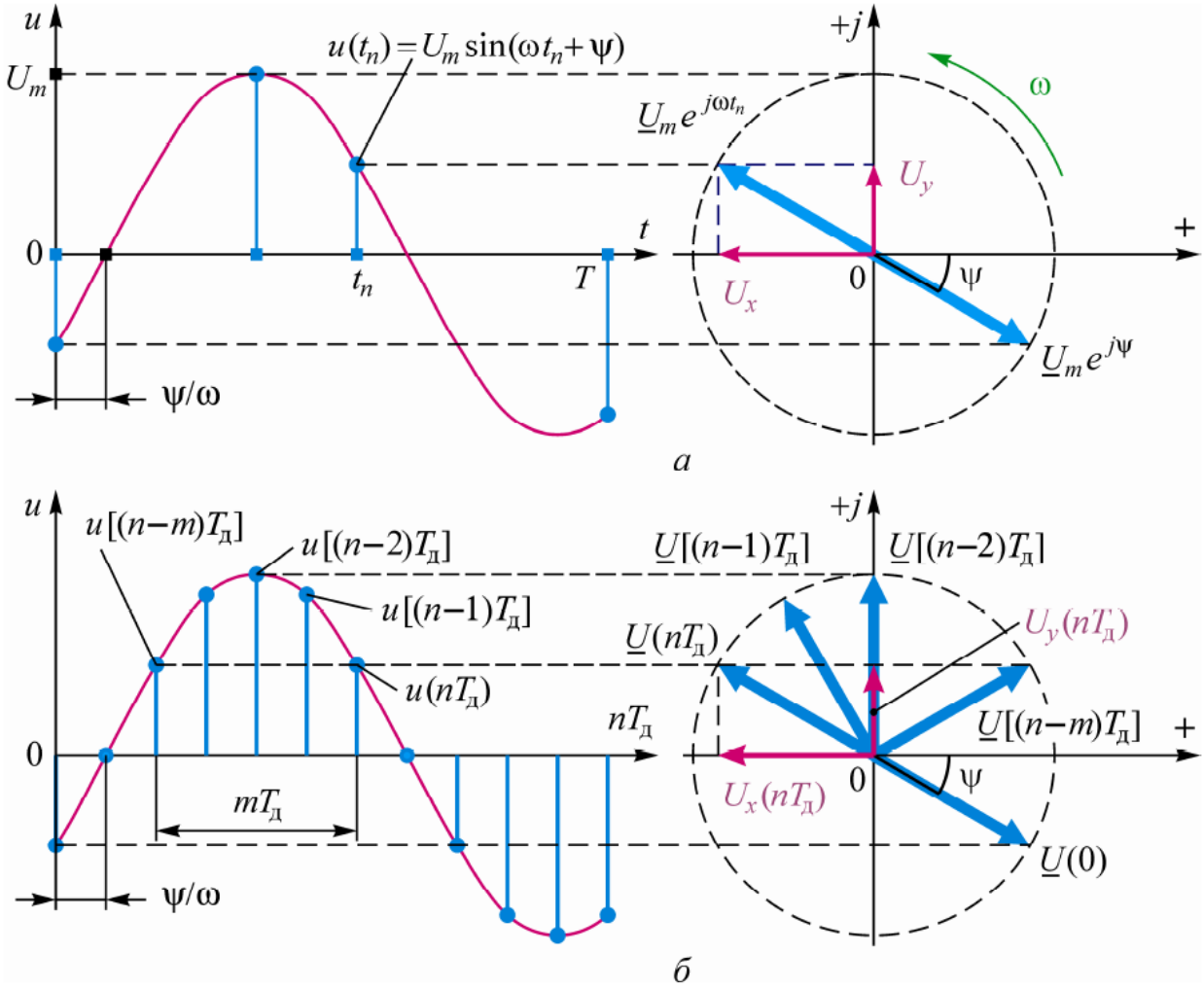


Рисунок 2.1 – Векторное представление синусоидальных величин:

а – напряжение и соответствующий ему вектор;

б – дискретизированное напряжение

Алгоритм не является точным из-за приближённого вычисления производной по по двум выборкам.

Вычисление вектора U(nТд) на основе соотношения (2.3) возможно и посредством усреднения замера мгновенного значения на отрезке Тд:

$$u(t) \approx \frac{1}{2}\left\{u(nT_д) + u\left[(n-1)T_д\right]\right\}; \quad \frac{du}{dt} \approx \frac{1}{T_д}\left\{u(nT_д) - u\left[(n-1)T_д\right]\right\}$$

Тогда вектор $U(nT_д)$ вычисляется по формуле

$$\underline{U}(nT_д) = \frac{1}{2}\left(\frac{N}{\pi} + j\right)u(nT_д) + \frac{1}{2}\left(j - \frac{N}{\pi}\right)u\left[(n-1)T_д\right] \qquad (2.4)$$

Ещё более точное вычисление вектора получается, если используются три выборки – $u(nT_д)$, $u[(n\text{-}1)T_д]$, $u[(n\text{-}2)T_д]$.

В этом случае можно принять

$$u(t) \approx u\left[(n-1)T_д\right]; \ \frac{du}{dt} \approx \frac{1}{2T_д}\left\{u(nT_д) - u\left[(n-2)T_д\right]\right\}$$

следовательно, вектор $U(nT_д)$ вычисляется по формуле

$$\underline{U}(nT_д) = \frac{N}{4\pi}\left\{u(nT_д) - u\left[(n-2)T_д\right]\right\} + ju\left[(n-1)T_д\right] \qquad (2.5)$$

Отметим, что алгоритмы имеют зависящие от числа выборок за период N даже при неизменной частоте измеряемых сигналов погрешности, характеризующие их параметры точности и частотные свойства.

Для лучшей отстройки от апериодических составляющих входного сигнала возможно применение алгоритма вычисления амплитуды U_m и фазы ψ гладкой синусоидальной функции, использующего первую и вторую производные [9]:

2.3 Использование выборок мгновенных значений для получения требуемых характеристик ЦИО

Рассмотрим возможность получения характеристик ЦИО заданной конфигурации на основе непосредственных операций с выборками величин без вычисления векторов, соответствующих напряжению и току.

Предварительно в цифровой форме найдём вектор, сдвинутый относительно вектора, соответствующего входному синусоидальному сигналу, на угол $\gamma = l\omega T_д$. Для этого заменим в выражении n на $(n - l)$:

$$U(nT_д)e^{-j\gamma} = u\left[(n-l)T_д\right](ctg\omega mT_д + j) - \frac{u\left[(n-m-l)T_д\right]}{\sin\omega mT_д} \qquad (2.6)$$

т.е. для поворота вектора на угол ±γ необходимо сместить последовательность на число циклов

$$l = \pm\frac{\gamma}{m\omega T_{д}} = \pm\frac{\gamma N}{2\pi} \qquad (2.7)$$

Применим соотношение (17) к принципу сравнения синусоидальных величин по фазе, используемому в электронных реле с двумя и более основными воздействующими величинами.

В простейшем случае сравниваемых величин две – e_1 и e_2. Пусть $\underline{E}_1$ и $\underline{E}_2$ – векторы, соответствующие аналоговым синусоидальным величинам e_1 и e_2 частоты ω. Как известно [9], указанный принцип требует, чтобы алгоритм преобразований, осуществляемых в измерительном органе, соответствовал условию, которое записывается как

$$0 \le \arg\frac{\underline{E}_2}{\underline{E}_1} \le \pi\,. \qquad (2.8)$$

Применительно к ЦИО это условие реализуется, если известны выборки величин e_1 и e_2, взятые соответственно в моменты времени $t_n = nT_{д}$ и $t_m = (n-m)T_{д}$: $e_1(nT_{д})$, $e_2(nT_{д})$, $e_1[(n-m)T_{д}]$, $e_2[(n-m)T_{д}]$. В соответствии с выражением (13) векторы $E_1(nT_{д})$ и $E_2(nT_{д})$ определяются как

$$\underline{E}_1(nT_{д}) = e_1(nT_{д})(\operatorname{ctg}\omega m T_{д} + j) - \frac{e_1[(n-m)T_{д}]}{\sin\omega m T_{д}};$$

$$\underline{E}_2(nT_{д}) = e_2(nT_{д})(\operatorname{ctg}\omega m T_{д} + j) - \frac{e_2[(n-m)T_{д}]}{\sin\omega m T_{д}}.$$

Если ввести новую переменную $\underline{W} = \underline{E}_1(nT_{д})/\underline{E}_1(nT_{д})$, то после преобразования уравнений, описывающих две указанные векторные величины, получается

$$\underline{W} = W_{\mathrm{Re}} + jW_{\mathrm{Im}}, \qquad (2.9)$$

где

$$W_{Re}=\langle e_1(nT_д)\cdot e_2(nT_д)+e_2[(n-m)T_д]\cdot e_1[(n-m)T_д]-$$
$$-\cos\omega mT_д\{e_1(nT_д)\cdot e_2[(n-m)T_д]+e_2(nT_д)\cdot e_1[(n-m)T_д]\}\rangle:$$
$$:\langle[e_1(nT_д)]^2-2e_1(nT_д)\cdot e_2(nT_д)\cos\omega mT_д+\{e_1[(n-m)T_д]\}^2\rangle;$$
$$W_{Im}=\langle\{e_1(nT_д)\cdot e_2[(n-m)T_д]-e_2(nT_д)\cdot e_1[(n-m)T_д]\}\sin\omega mT_д\rangle:$$
$$:\langle[e_1(nT_д)]^2-2e_1(nT_д)\cdot e_2(nT_д)\cos\omega mT_д+\{e_1[(n-m)T_д]\}^2\rangle.$$

Условие $0\le\arg(\underline{E}_1/\underline{E}_2)\le\pi$, соответствующее так называемой синусной схеме сравнения фаз [9], выполняется, если $\underline{W}$ находится в верхней полуплоскости. Следовательно, для реализации указанного условия необходимо и достаточно, чтобы в (2.9) числитель мнимой части W_{Im} величины $\underline{W}$ был положителен:

$$\{e_1(nT_д)\cdot e_2[(n-m)T_д]-e_2(nT_д)\cdot e_1[(n-m)T_д]\}\sin\omega mT_д\ge 0.$$

Для числа циклов между выборками $m<N/2$ ($m\omega T_д < \pi$) всегда $\sin m\omega T_д>0$, поэтому достаточно только выполнения условия

$$e_1(nT_д)\cdot e_2[(n-m)T_д]-e_2(nT_д)\cdot e_1[(n-m)T_д]\ge 0. \qquad (2.10)$$

С учётом выражений (16) и (20) для построения схемы сравнения фаз с характеристикой, соответствующей неравенству

$$\gamma\le\arg\frac{\underline{E}_2}{\underline{E}_1}\le\pi+\gamma, \qquad (2.11)$$

необходимо выполнение условия

$$e_1(nT_д)\cdot e_2[(n-m-l)T_д]-e_2[(n-l)T_д]\cdot e_1[(n-m)T_д]\ge 0, \qquad (2.12)$$

получаемое при замене величины $\underline{E}_2$ в неравенстве (18) на $\underline{E}_2e^{-j\gamma}$.

По условию (2.12), используя две выборки мгновенных значений сигналов e_1 и e_2, можно выполнить ЦИО с произвольно расположенными в плоскости сопротивлений характеристиками в виде прямых, окружностей и их комбинаций. Для этого формируются величины $\underline{E}_1$ и $\underline{E}_2$, являющиеся линейными функциями токов и напряжений защищаемого объекта:

$$\left.\begin{aligned} \underline{E}_1 &= \underline{k}_1\underline{U} + \underline{k}_2\,\underline{I} = \underline{k}_1\underline{I}\,(\underline{Z} - \underline{Z}_1);\ \underline{Z}_1 = -\underline{k}_2/\underline{k}_1\ ; \\ \underline{E}_2 &= \underline{k}_3\,\underline{U} + \underline{k}_4\,\underline{I} = \underline{k}_3\,\underline{I}\,(\underline{Z} - \underline{Z}_2);\ \underline{Z}_2 = -\underline{k}_4/\underline{k}_2\,. \end{aligned}\right\} \qquad (23)$$

Получение необходимых характеристик производится выбором соответствующих значений коэффициентов $\underline{k}_1 \ldots \underline{k}_4$.

Условие (22) обеспечивается операциями с двумя выборками сформированных величин e_1 и e_2, соответствующих векторам $\underline{E}_1$ и $\underline{E}_2$. При формировании величин $\underline{E}_1$ и $\underline{E}_2$ комплексные коэффициенты $\underline{k}_1 \ldots \underline{k}_4$ реализуются умножением соответствующих выборок тока *i* и напряжения *u* на числа, равные модулям коэффициентов $\underline{k}_1 \ldots \underline{k}_4$, а фазовые сдвиги – на основе выражения (2.6) – сдвигом номеров выборок.

2.4 Цифровые реле тока и напряжения

Алгоритмы вычисления модуля вектора могут быть реализованы простейшим способом путём непосредственного вычисления действующего значения, например:

$$I^2 = \frac{1}{T_1}\int_{t-T_1}^{t} i^2\,dt.$$

С учётом дискретизации сигнала в АЦП формуле вычисления действующего значения соответствует алгоритм

$$I^2 = \frac{1}{N}\sum_{n-N+1}^{n} i[(nT_д)]^2. \qquad (2.13)$$

Для вычисления амплитуды по среднему значению можно использовать формулу

$$I = \frac{1}{T_1}\int_{t-T_1}^{t} |i(t)|\,dt.$$

Дискретизация формулы вычисления среднего значения даёт следующий результат:

$$I_{ср}=\frac{1}{N}\sum_{n-N+1}^{n}\left|i(nT_д)\right|. \qquad (2.14)$$

Цифровые реле тока и напряжения могут быть построены на основе структур, показанных на рис. 2.1, б. В частности, алгоритм цифрового реле тока с уставкой $I_{уст}$, подключаемого к измерительному ТТ фазы А, при вычислении ортогональных составляющих имеет вид

$$I_{Ax}^2(nT_д)+I_{Ay}^2(nT_д)\geq I_{уст}^2, \qquad (2.15)$$

где ортогональные составляющие вычисляются в соответствии с принятым алгоритмом измерения параметров векторов, например, по выражению (2.13).

Если организовать вычисления так, чтобы они начинались в момент, когда кривая тока или напряжения проходит через нуль, т.е. $u(nT_д) = 0$ при $nT_д = 0$, то измеряемый вектор $\underline{U}_{m1}$ имеет нулевую фазу. Тогда мнимая составляющая $U_{ym1}= 0$, а вещественная составляющая имеет амплитудное значение: $U_{xm1}= U_{m1}$. Для реле с одной входной воздействующей величиной это упрощает вычисления. В частности, выражение (2.15) для реле тока становится таким:

$$I_{Ax}(nT_д)\geq I_{уст}. \qquad (2.16)$$

2.5 Цифровые реле сопротивления

Синтез алгоритмов способами вычисления комплексного сопротивления.

Для синтеза алгоритмов цифровых реле сопротивления можно использовать способы, основанные на вычислении комплексного сопротивления как отношения амплитуды напряжения $u(t)=U_m sin(\omega_1 t + \varphi)$ к амплитуде тока $i(t) =I_m sin\omega_1 t$, подводимых к реле от измерительных ТТ и ТН в качестве основных воздействующих величин.

Рассмотрим основные способы "цифрового" вычисления комплексного сопротивления.

Получение сопротивления Z по ортогональным составляющим тока и напряжения. Данный способ основан на использовании для вычисления комплексного сопротивления ортогональных составляющих векторов $\underline{U}_m = U_x + jU_y$ и $\underline{I}_m = I_x + jI_y$, методы нахождения которых рассмотрены ранее:

$$\underline{Z} = \frac{\underline{U}_m}{\underline{I}_m} = \frac{U_x + j\,U_y}{I_x + j\,I_y} = R + jX. \qquad (2.17)$$

В выражении (3.9) вещественная и мнимая части вычисляются по формулам

$$R = \frac{U_x I_x + U_y I_y}{I_x^2 + I_y^2};\; X = \frac{U_y I_x - U_x I_y}{I_x^2 + I_y^2}.$$

Получение сопротивления Z по интегральным значениям величин. Вычислим интеграл произведения мгновенных значений тока $i(t) = I_m \sin\omega_1 t$ и напряжения $u(t) = U_m \sin(\omega_1 t + \varphi)$:

$$\frac{2}{T_1}\int_0^{T_1} U_m \sin(\omega_1 t + \varphi) \times I_m \sin\omega_1 t\, dt \Rightarrow UI\cos\varphi,$$

что эквивалентно вычислению активной мощности P. Реализуя данную операцию дискретным образом, получаем

$$P = \frac{2}{N}\sum_{n-N+1}^{n} u(nT_д)\, i(nT_д). \qquad (2.18)$$

Аналогичным образом можно определить реактивную мощность $Q = U_m I_m \sin\varphi$:

$$Q = \frac{2}{N}\sum_{n-N+1}^{n} u\left[\left(n - \frac{N}{4}\right)T_д\right] i(nT_д), \qquad (2.19)$$

где выборки напряжения $u(nT_д)$ сдвигаются во времени на четверть периода.

Используя для вычисления вещественной $R = P/I^2$ и мнимой $X = Q/I^2$ частей комплексного сопротивления соотношения (3.10), (3.11) и алгоритм (2.19), получаем

$$R=\frac{\sum_{n-N+1}^{n}u(nT_д)\,i(nT_д)}{\sum_{n-N+1}^{n}[i(nT_д)]^2};\quad X=\frac{\sum_{n-N+1}^{n}u\left[\left(n-\frac{N}{4}\right)T_д\right]i(nT_д)}{\sum_{n-N+1}^{n}[i(nT_д)]^2}. \qquad (2.20)$$

Получение сопротивления Z непосредственной обработкой сигналов с использованием алгоритма двух выборок

Если m – число периодов дискретизации между выборками, по которым производятся вычисления, то текущее значение сопротивления

$$\underline{Z}=\frac{\underline{U}(nT_д)}{\underline{I}(nT_д)}=\frac{u(nT_д)\cos\omega_1 mT_д-u[(n-m)T_д]+ju(nT_д)\sin\omega_1 mT_д}{i(nT_д)\cos\omega_1 mT_д-i[(n-m)T_д]+ji(nT_д)\sin\omega_1 mT_д}. \qquad (2.21)$$

Обозначим дискретизированные значения тока и напряжения как $u_1 = u[(n - m)T_д]$, $u_2 = u(nT_д)$, $i_1 = u[(n - m)T_д]$, $i_2 = u(nT_д)$. Считая, что в тригонометрических функциях синуса и косинуса произведение $\omega_1 mT_д = \beta_\mu$, после преобразования выражения (2.21) получаем формулы вычисления вещественной и мнимой частей комплексного сопротивления:

$$R=\frac{u_1i_1-(u_1i_2+u_2i_1)\cos\beta_\mu+u_2i_2}{i_1^2-2i_1i_2\cos\beta_\mu+i_2^2};\quad X=\frac{(u_1i_2-u_2i_1)\sin\beta_\mu}{i_1^2-2i_1i_2\cos\beta_\mu+i_2^2}. \qquad (2.22)$$

В том случае, когда $m= N/4$ и $\beta_\mu = \pi/2$, выражения (2.22) несколько упрощаются:

$$R=\frac{u_1i_1+u_2i_2}{i_1^2+i_2^2};\quad X=\frac{u_1i_2+u_2i_1}{i_1^2+i_2^2}. \qquad (2.23)$$

Вычисление сопротивления Z на основе уравнения короткозамкнутой линии.

Данный алгоритм предполагает, что производятся операции с синусоидальными величинами. Соотношения между напряжением u(t) и i(t), подведёнными к линии, замещаемой активно-индуктивным звеном, в любой момент времени описываются дифференциальным уравнением 1-го порядка:

$$u = Ri + L\frac{di}{dt} \qquad (2.24)$$

где R, L – активное сопротивление и индуктивность линии. Индуктивность линии для синусоидальных сигналов основной частоты $L = X/\omega_1$. Тогда для моментов времени t_1 и t_2 имеем систему из двух уравнений:

$$\left.\begin{aligned} u_1 &= Ri_1 + \frac{X}{\omega_1}\frac{di}{dt}\Big|_{t=t_1}; \\ u_2 &= Ri_2 + \frac{X}{\omega_1}\frac{di}{dt}\Big|_{t=t_2}, \end{aligned}\right\} \qquad (2.25)$$

где $u_1 = u(t_1)$, $i_1 = i(t_1)$, $u_2 = u(t_2)$, $i_2 = i(t_2)$; $di/dt\big|_{t=t_1}$, $di/dt\big|_{t=t_2}$ – первые производные функции $i(t)$ в указанные моменты времени.

Решая систему (2.25), получаем вычисления вещественной и мнимой частей комплексного сопротивления:

$$R = \frac{u_1\frac{di}{dt}\Big|_{t=t_2} - u_2\frac{di}{dt}\Big|_{t=t_1}}{i_1\frac{di}{dt}\Big|_{t=t_2} - i_2\frac{di}{dt}\Big|_{t=t_1}}; \quad X = \omega_1\frac{i_1u_2 - i_2u_1}{i_1\frac{di}{dt}\Big|_{t=t_2} - i_2\frac{di}{dt}\Big|_{t=t_1}} \qquad (2.25)$$

На основе соотношений (2.25) возможно определение Z для синусоидальных величин – ток $i(t)=I_m sin(\omega_1 t + \varphi_i)$ и напряжение $u(t)=I_m sin(\omega_1 t + \varphi_u)$, не обязательно имеющихся в простой RL–цепи. При этом считается, что напряжение $u(t)$ подводится к фиктивной RL–цепи, где R и L могут быть как положительными, так и отрицательными в зависимости от значения $\varphi_u = \varphi_u - \varphi_i$, т.е. квадранта комплексной плоскости, в котором располагается сопротивление Z [9].

2.6 Алгоритм токовой отсечки

Согласно ПУЭ для электродвигателей мощностью менее 2 МВт должна быть обязательно предусмотрена однорелейная токовая отсечка, защищающая от многофазных замыканий.

Если однорелейная токовая отсечка не удовлетворяет требованиям чувствительности, то для защиты электродвигателей мощностью менее 2 М Вт можно использовать двухрелейную токовую отсечку.

Сразу необходимо отметить, что при коротком замыкании между фазами *AB* или *BC* на вход схемы однорелейной токовой отсечки от двух трансформаторов тока, соединенных на разность токов (рисунок 2.1), поступает сигнал в 3 раза меньший, чем при симметричной нагрузке или при трехфазном коротком замыкании.

В технической литературе [22] принято говорить, что такая схема имеет в $\sqrt{3}$ раз худшую чувствительность, чем двухрелейная схема с двумя трансформаторами тока [1], однако правильнее говорить, выходной ток в такой схеме в $\sqrt{3}$ раз меньше.

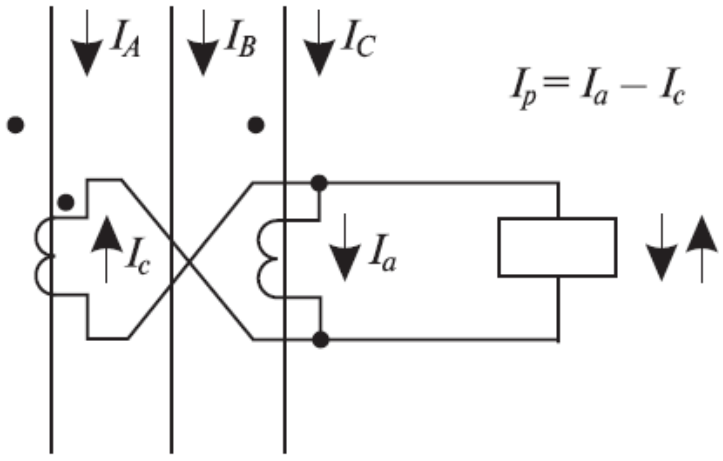


Рисунок 2.2 – Соединение трансформаторов тока на разность токов двух фаз

ПУЭ рекомендует применять двухрелейную токовую отсечку для защиты электродвигателей мощностью 2 МВт и более, имеющих защиту от однофазных замыканий на землю, действующую на отключение.

Если же защита от однофазных замыканий на землю отсутствует, то для электродвигателей мощностью 2 МВт и более следует применять трехрелейную токовую отсечку с тремя трансформаторами тока.

ПУЭ допускает применять и двухрелейную токовую отсечку для защиты электродвигателей мощностью 2 МВт и более, не имеющих защиты от однофазных замыканий на землю. Однако в этом случае необходимо дополнительно предусмотреть защиту от двойных замыканий на землю.

Наиболее просто и полно все требования, изложенные в ПУЭ, реализуются при использовании серийно выпускаемых устройств БМРЗ и «Сириус-Д», предназначенных для защиты синхронных и асинхронных электродвигателей.

В ряде случаев для защиты асинхронных двигателей можно применять устройства БМРЗ и БМРЗ-100, рассчитанные на защиту воздушных и кабельных линий.

Во всех этих блоках для выполнения токовой отсечки используют первую ступень алгоритма максимальной токовой защиты МТЗ с нулевой выдержкой времени (рис. 2.3).

При превышении любым из фазных токов $I_л$, I_B, I_C уставки соответствующего компаратора *1 – 3* возникает сигнал *«Пуск* /.» и при отсутствии блокирующих сигналов элемент выдержки времени *5* начинает отсчет.

При использовании первой ступени МТЗ в качестве токовой отсечки выдержка времени устанавливается равной нулю, поэтому сигнал **Откл. />*» на выходе алгоритма появляется после сигнала *«Пуск />>* без временно2й задержки.

Блокирование срабатывания любой ступени МТЗ выполняется элементом *4.* Сигнал блокирования поступает на элемент *13.*

В связи с тем, что в данном алгоритме устанавливается нулевое значение выдержки времени, то необходимость ускорения срабатывания защиты (при ручном включении выключателя или в цикле АПВ) отсутствует.

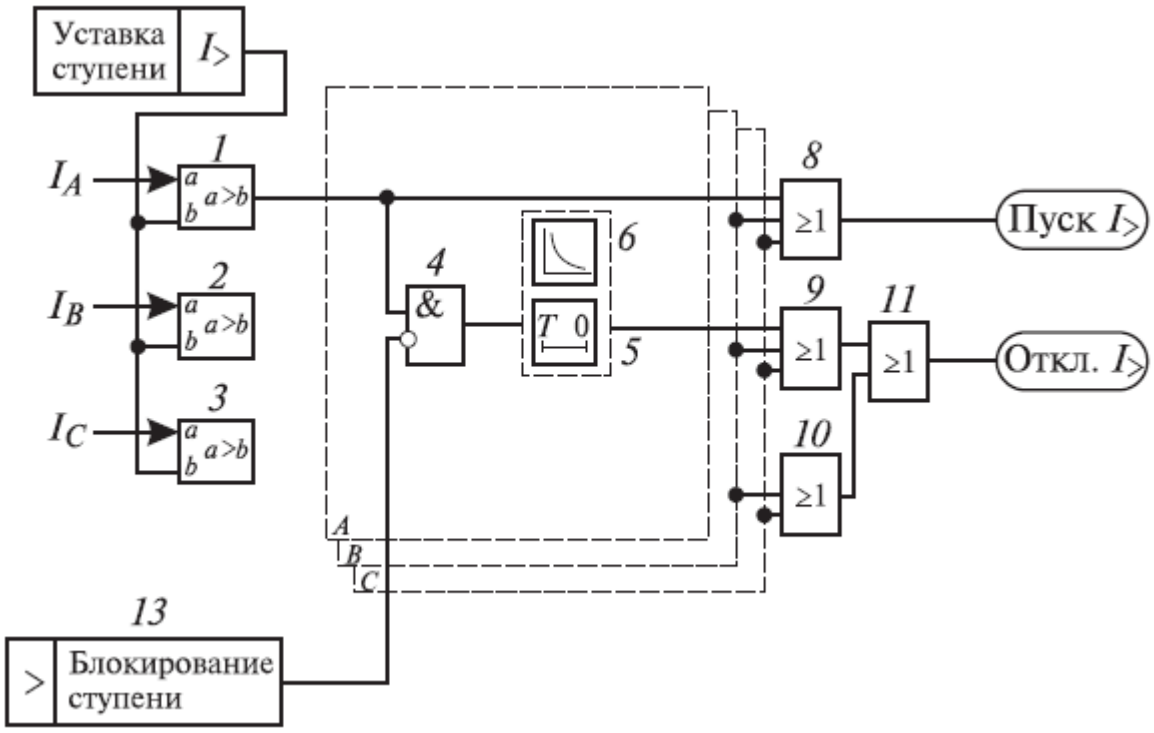


Рисунок 2.3 – Структурная схема алгоритма максимальной токовой зашиты (ТО – первая ступень МТЗ)

В устройствах БМРЗ и «Сириус-Д» предусмотрено необходимое количество цифровых реле максимального тока для каждой фазы, поэтому применение предусмотренной в ПУЭ отсечки в виде однорелеиной схемы нецелесообразно.

Выводы к 2-й главе

Во данной главе выполнен анализ алгоритмов измерительных органов цифровых защит, алгоритмов вычисления векторов по мгновенным значениям синусоидальных величин. Также проведен анализ функционирования алгоритмов использования выборок мгновенных значений для получения требуемых характеристик ЦИО и использования при построении алгоритма повышения чувствительности.

Рассмотрены принципы функционирования цифровых реле тока и напряжения, цифровых реле сопротивления, алгоритм токовой отсечки.

3. ИНСТРУМЕНТАРИЙ МАТЕМАТИЧЕСКОГО МОДЕЛИРОВАНИЯ НОРМАЛЬНЫХ И АВАРИЙНЫХ РЕЖИМОВ РАБОТЫ ДВИГАТЕЛЕЙ

3.1 Среда моделирования PSCAD

PSCAD (Power Systems Computer Aided Design) – это мощный и гибкий графический пользовательский интерфейс для всемирно известного модуля электромагнитного моделирования переходных процессов EMTDC. PSCAD позволяет пользователю схематически построить схему, запустить симуляцию, проанализировать результаты и управлять данными в полностью интегрированной графической среде. Также включены функции построения графиков, элементы управления и счетчики, позволяющие пользователю изменять параметры системы во время симуляции и, таким образом, просматривать эффекты во время симуляции.

PSCAD поставляется с библиотекой предварительно запрограммированных и протестированных имитационных моделей, начиная от простых пассивных элементов и функций управления и заканчивая более сложными моделями, такими как электрические машины, полнофункциональные устройства FACTS, линии передачи и кабели. Если требуемая модель не существует, PSCAD предоставляет возможности для создания пользовательских моделей. Например, пользовательские модели могут быть созданы путем объединения существующих моделей в модуль или путем создания элементарных моделей с нуля в гибкой среде проектирования.

Ниже приведены некоторые распространенные модели, которые можно найти в главной библиотеке PSCAD:

Резисторы, катушки индуктивности, конденсаторы

Взаимосвязанные обмотки, такие как трансформаторы

Частотно-зависимые линии передачи и кабели (включая самую точную модель линий во временной области в мире!)

Источники тока и напряжения

Выключатели и выключатели

Защита и ретрансляция

Диоды, тиристоры и гто

Аналоговые и цифровые функции управления

Машины переменного и постоянного тока, возбудители, регуляторы, стабилизаторы и инерционные модели

Метры и измерительные функции

Общий контроль постоянного и переменного тока

HVDC, SVC и другие контроллеры FACTS

Источник ветра, турбины и регуляторы

PSCAD и его механизм моделирования EMTDC развивались почти 40 лет, вдохновляясь идеями и предложениями постоянно растущей всемирной пользовательской базы. Эта философия разработки помогла сделать PSCAD одним из самых мощных и интуитивно понятных пакетов программного обеспечения САПР.

PSCAD был впервые концептуализирован в 1988 году и начал свою разработку в качестве графического интерфейса для программы моделирования электромагнитных переходных процессов EMTDC. В своей предварительной коммерческой форме PSCAD был в значительной степени экспериментальным; тем не менее, это стало гигантским скачком в производительности, поскольку пользователи EMTDC могли проектировать свои системы схематично, а не вводить данные через текстовые списки. Графические аспекты PSCAD улучшали общее восприятие моделируемой системы, значительно ускоряя сборку схемы и сводя к минимуму ошибки.

PSCAD используется для планирования, проектирования, эксплуатации, подготовки тендерной документации, при преподавании и проведении научных исследований. Ниже приведены некоторые примеры исследований, для которых обычно используется PSCAD:

– исследования аварийных ситуаций в сетях переменного тока, включающих в себя вращающиеся электрические машины и возбудители, регуляторы частоты вращения, турбины, трансформаторы, воздушные и кабельные линии электропередачи, различные виды нагрузок;

– анализ работы релейной защиты;

– исследования процессов насыщения трансформаторов;

– согласование изоляции трансформаторов, выключателей и разрядников;

– импульсное тестирование трансформаторов;

– исследование подсинхронного резонанса сети и электрических машин, линий электропередачи постоянного тока (HVDC);

– анализ гармонических составляющих и выбор оптимального фильтра;

– разработка системы управления и согласование силовых полупроводниковых установок: FACTS, HVDC, STATCOM, VSC и преобразователей частоты;

– исследование новых решений и концепций систем управления;

– исследование ударов молний, неисправностей при работе выключателя;

– изучение сверхбыстрых процессов с крутыми фронтами;

– разработка полностью электрических кораблей;

– исследование эффекта пульсации в сетях с дизельными генераторами и ветровыми турбинами.

Прикладная среда.

Термин «среда приложения» относится не только к тому, как PSCAD организован визуально, но также к соглашениям об именах, утилитам и другим функциям, которые облегчают его использование. Много усилий постоянно затрачивается на рабочую среду – основными целями являются постоянство внешнего вида предыдущих версий, а также улучшение существующих функций и своевременное добавление новых.

Наборы моделирования / Несколько EMTDC

Можно запустить и запустить несколько симуляций EMTDC одновременно. Как последовательные, так и параллельные прогоны

симуляции возможны посредством определения так называемых «наборов симуляции» в главном окне рабочей области.

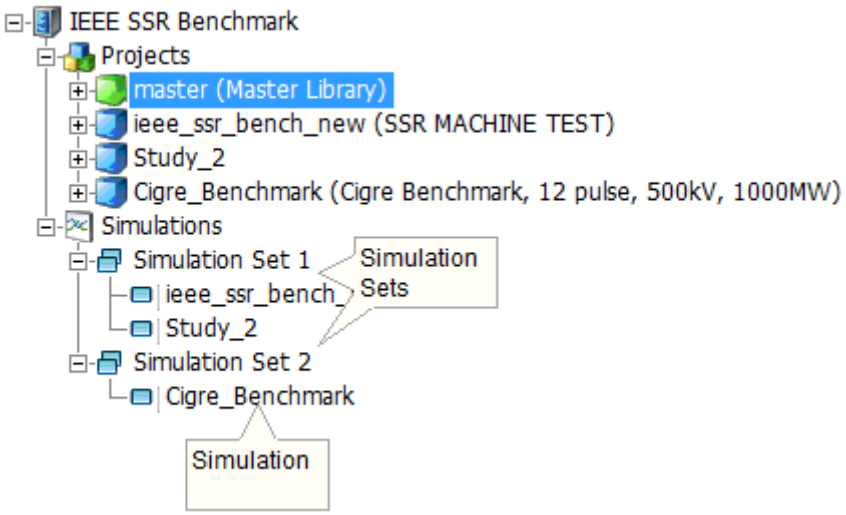


Рисунок 3.1 – Главное окно рабочей области

Только проекты, загруженные под веткой Projects в главном окне рабочей области, могут быть добавлены как Задача моделирования в Набор моделирования. Все моделирования в определенном наборе будут запущены одновременно, используя все доступные ресурсы процессора. Каждый набор запускается последовательно: на изображении выше, например, Simulation Set 1 запустит и запустит проекты ieee_ssr_bench_new и Study_2 одновременно. После завершения Simulation Set 2 запустит и запустит проект Cigre_Benchmark.

Управление мульти-EMTDC можно найти как во всплывающих меню Simulation по щелчку правой кнопкой мыши, так и в кнопке Run на ленте.

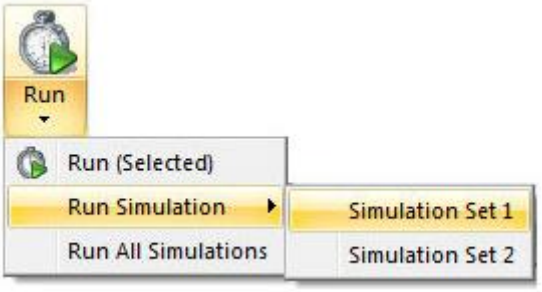


Рисунок 3.2 – Всплывающее меню Simulation

Визуальное построение моделей.

Построение и изменение модели осуществляется максимально наглядно и выполняется в графическом виде как привычная электрическая схема. Характеристики и свойства каждого элемента модели задаются в виде табличных данных. Готовая модель наглядна, понятна для анализа и проверки.

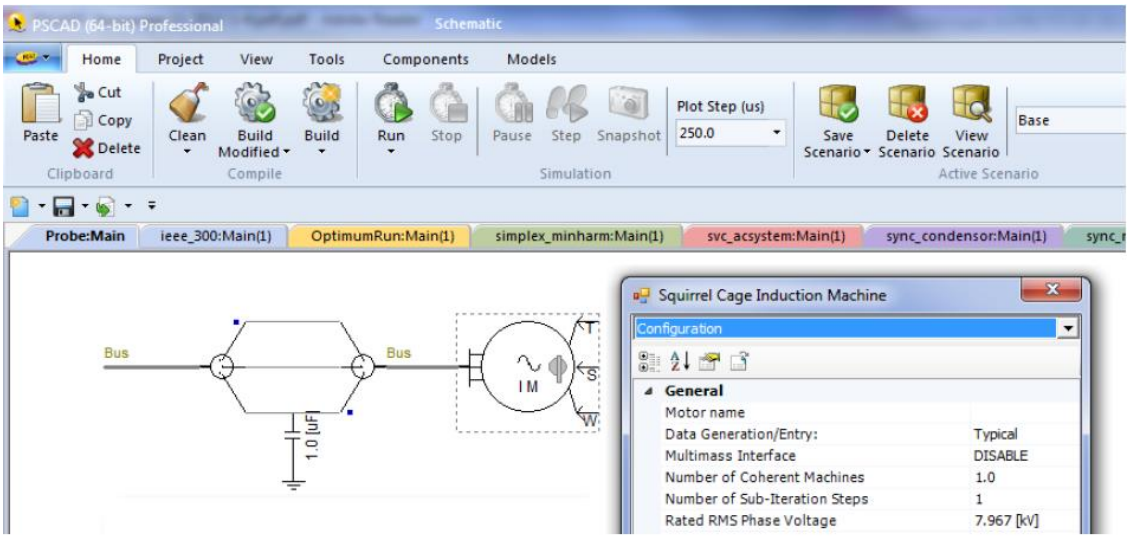


Рисунок 3.3 – Наглядная форма представления модели

Послойная структура модели.

Для удобства работы и наглядности, модель может быть многослойной: на главном слое сложные объекты представлены в виде укрупненных объектов, входя в которые можно увидеть детальное строение объекта

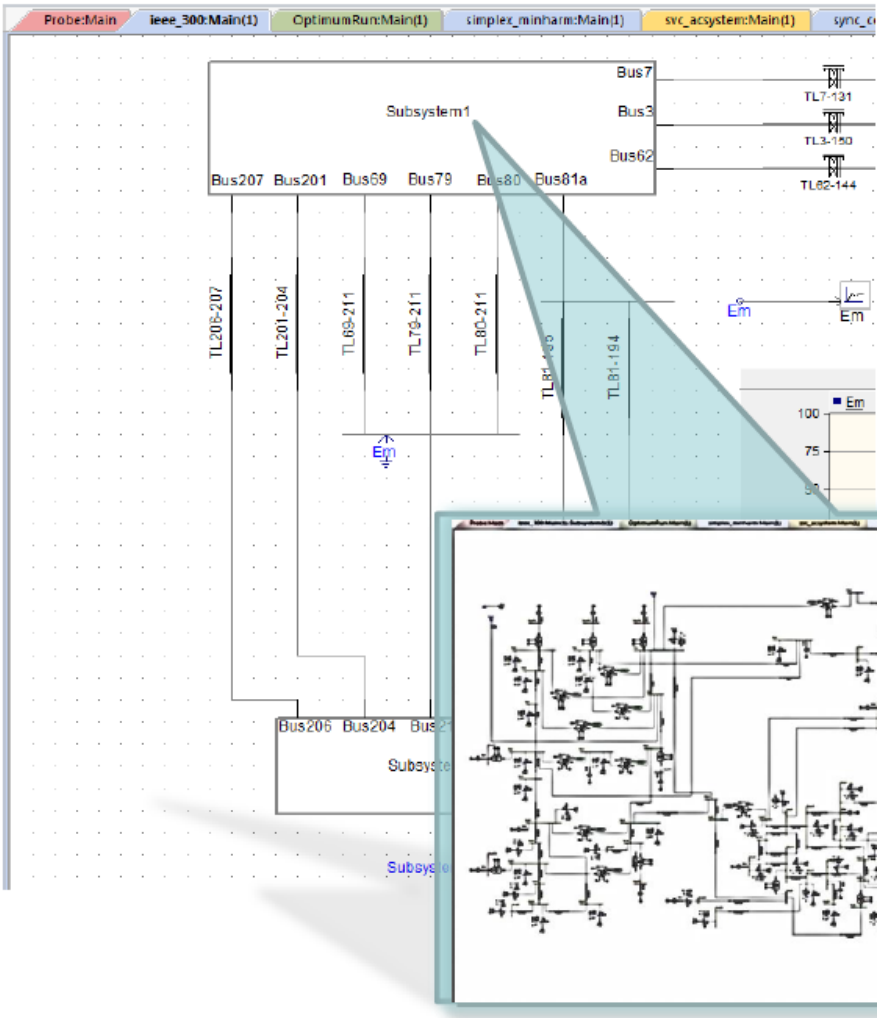


Рисунок 3.4 – Многослойная модель

PSCAD содержит около 300 компонентов:

– Пассивные элементы;

– Источники электроэнергии;

– Выключатели;

– Силовая электроника;

– Трансформаторы;

– Электрические машины;

– ВЛ и КЛ;

– Измерители;

– Релейная защита;

– Экспорт и импорт данных;

– Логические функции.

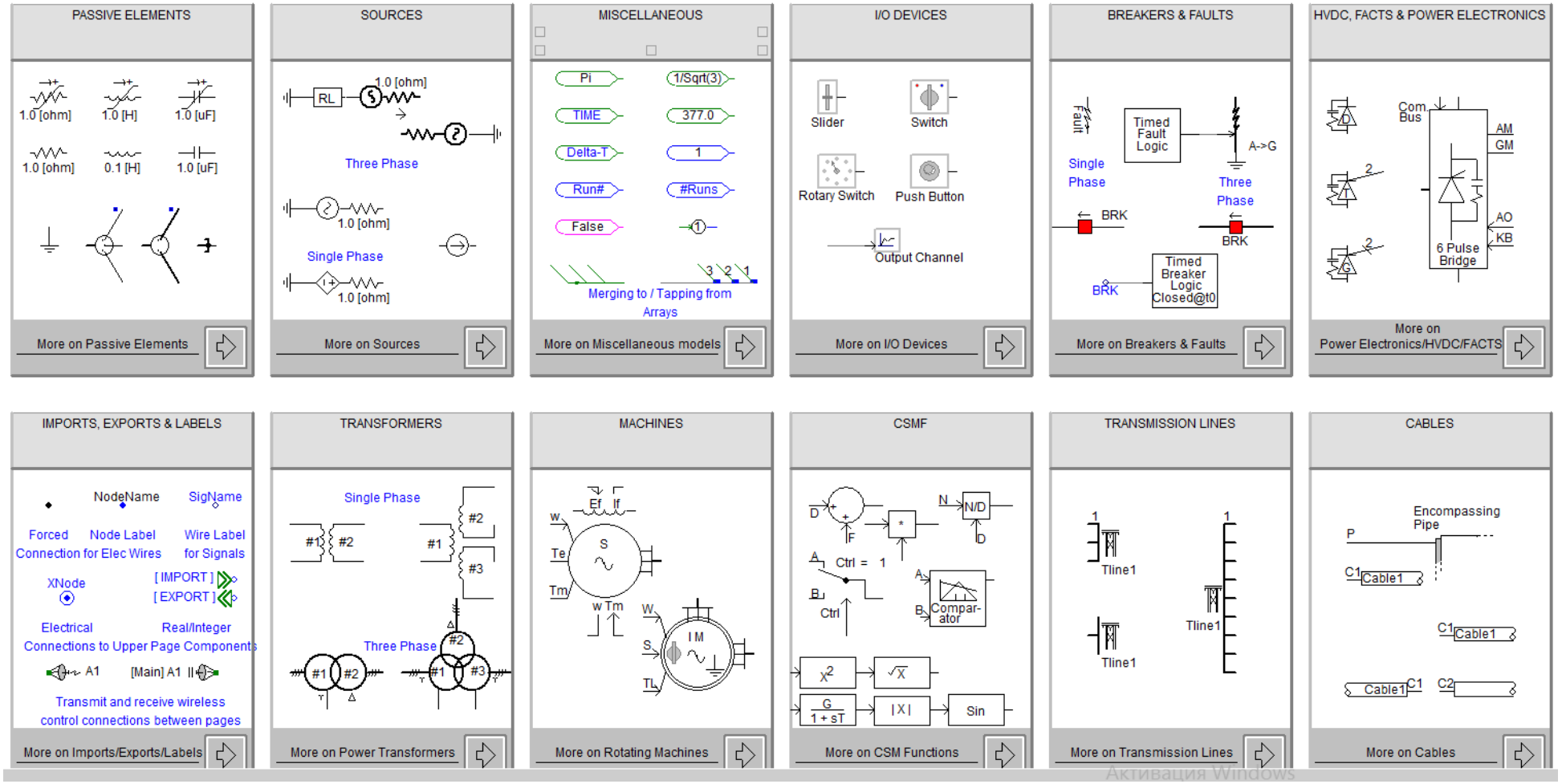


Рисунок 3.5 – Главное меню с компонентами

3.2 Описание экспериментальной схемы

Пример запуска асинхронного двигателя:

Машина намотана ротором.

Двигатель запускается с нулевой скорости.

Применяемый механический момент, T_m, изменяется в зависимости от скорости, то есть:

$T_m = T_{load} = k * w * w + b$

w = скорость

k, b = постоянные.

Машина ускоряется, если $T_e > T_m$

Через 3 с механический момент переключается на 1,8 о. е. с помощью «управляемого» переключателя. Машина проходит переходное состояние и стабилизируется на новой скорости.

Характеристики запуска зависят от внешнего сопротивления ротора, подключенного к ротору. Это значение можно изменить с помощью «ползунка». Такие параметры, как инерция машины, демпфирование, также могут быть изменены внутри компонента, чтобы изучить их влияние на запуск.

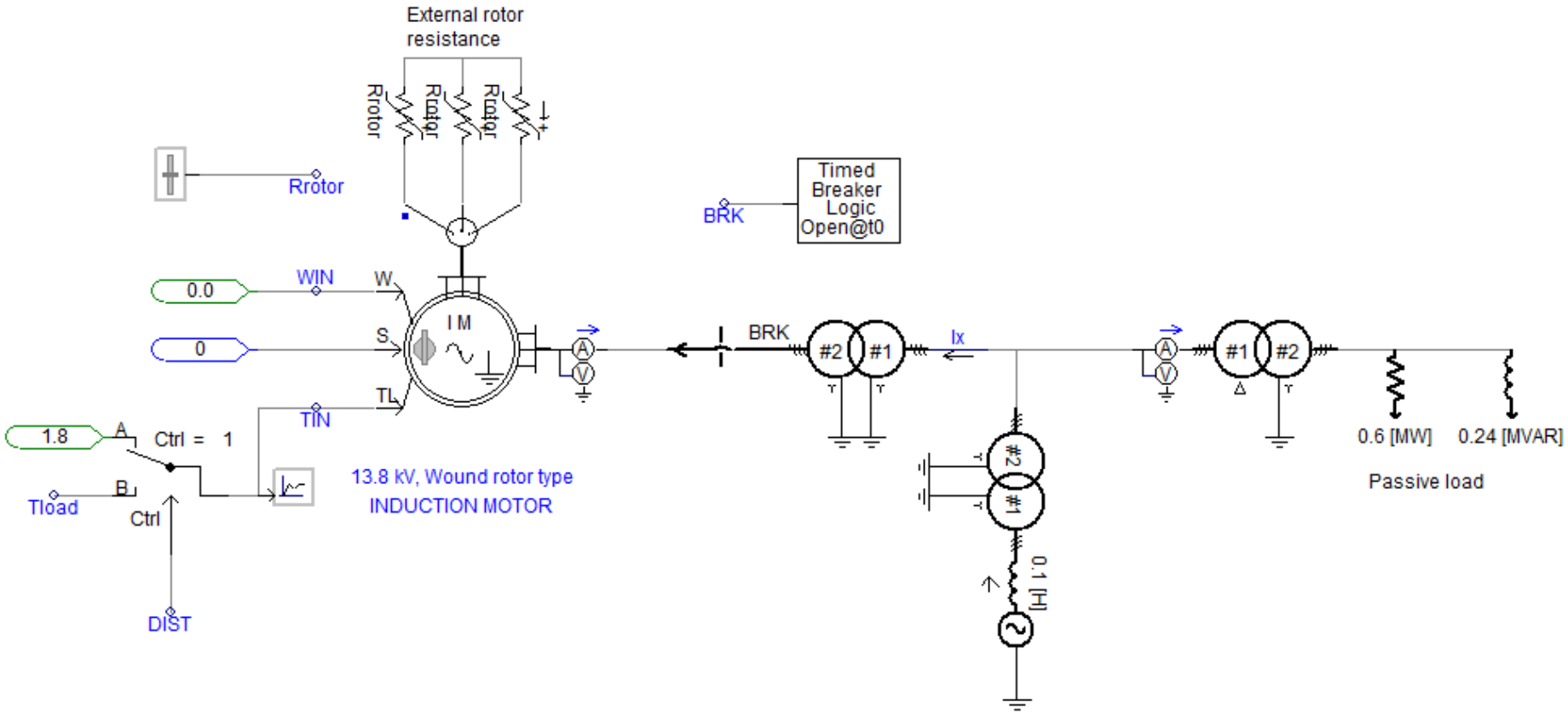


Рисунок 3.6 – Модель участка сети и асинхронного электродвигателя в среде PSCAD

Параметры модели электродвигателя приведены на рисунках

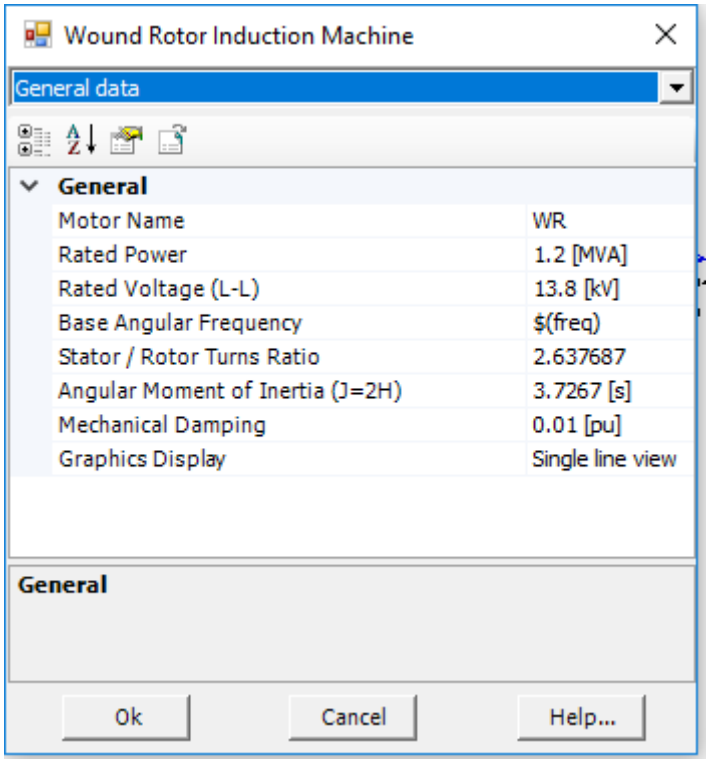


Рисунок 3.7 – Основные параметры АД

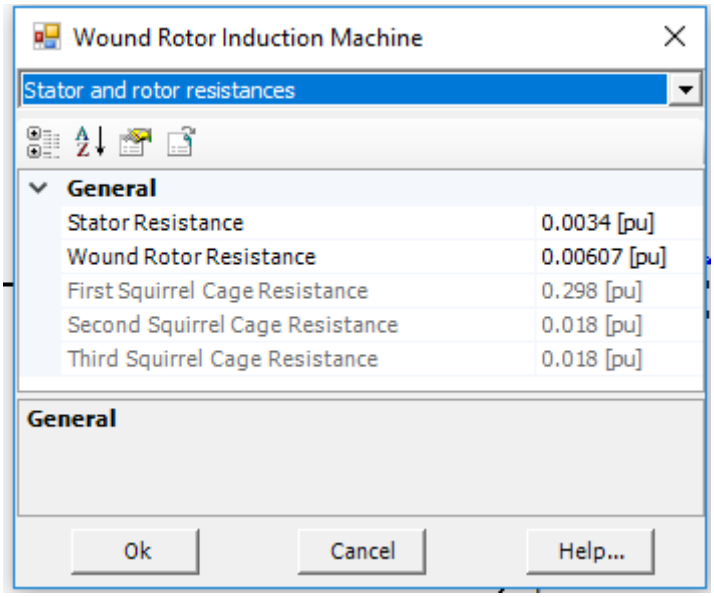


Рисунок 3.8 – Сопротивления обмоток статора и ратора

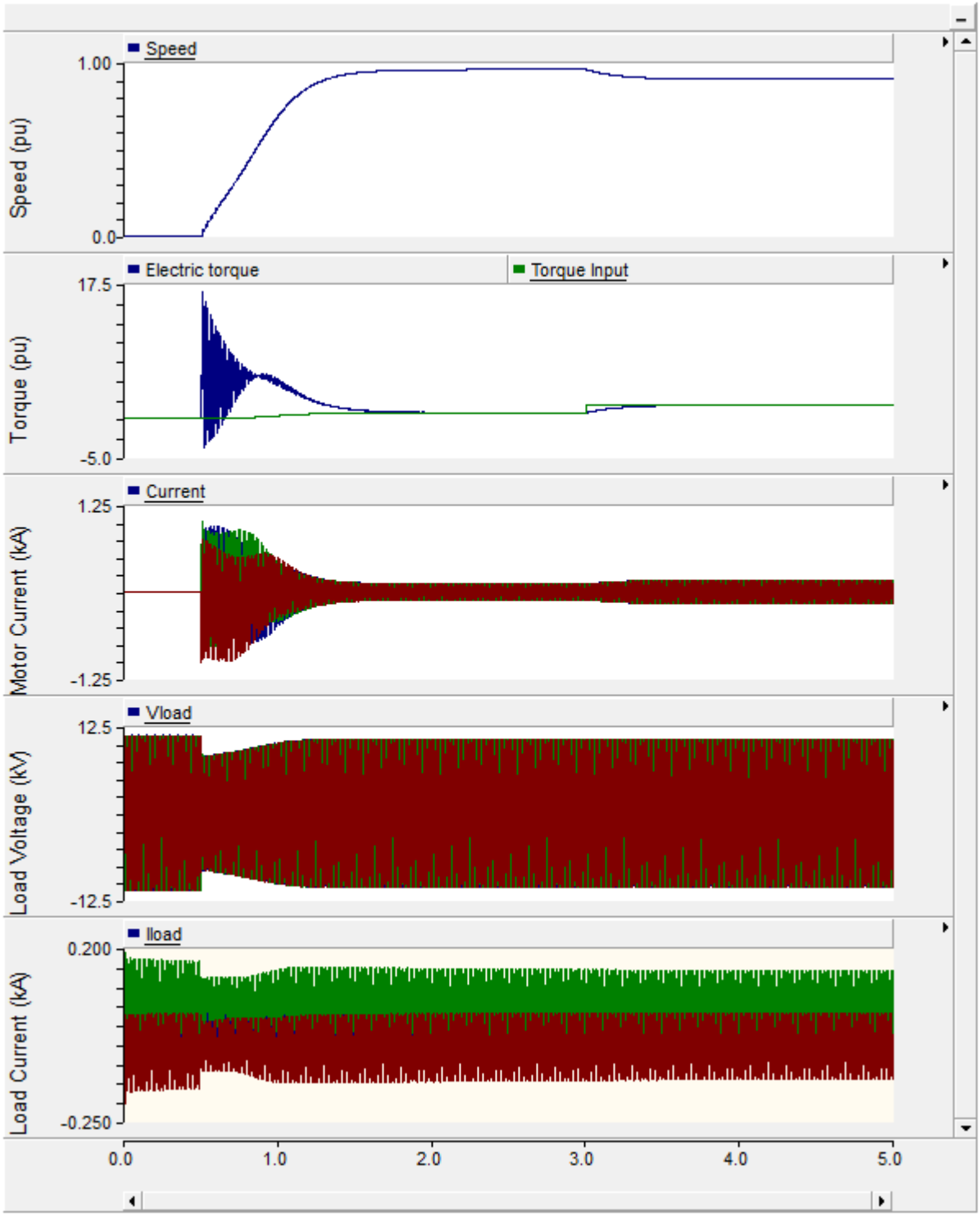


Рисунок 3.9 – Осциллограммы параметров асинхронного двигателя при запуске: 1) угловая скорость двигателя в относительных единицах, 2) Момент на валу в момент запуска и при подключении нагрузки, 3) Фазные токи двигателя, 4) Напряжение на нагрузке, 5) Ток на нагрузке.

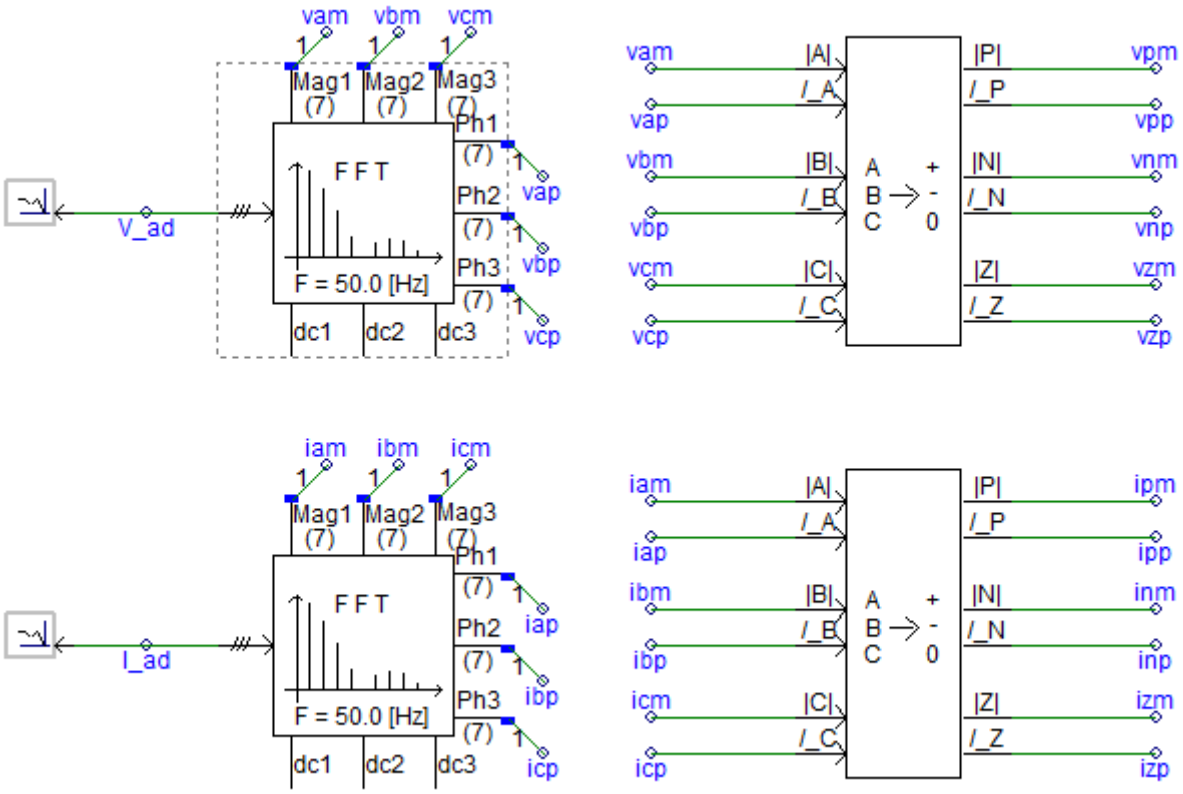


Рисунок 3.10 – Элементы алгоритма цифровой защиты электродвигателя

На рисунке 3.10 в левой части представлены блоки быстрого преобразования Фурье для сигналов тока и напряжения в месте подключения двигателя. Для выделения коэффициентов a_{m1} и b_{m1} 1-й гармоники $u_1(t) = a_{m1}\cos\omega_1 t + b_{m1}\sin\omega_1 t$ несинусоидального напряжения $u(t)$, представленного в виде ряда:

$$f(t) = \frac{a_0}{2} + \sum_{q=1}^{\infty}(a_{mq}\cos q\omega_1 t + b_{mq}\sin q\omega_1 t)$$

преобразователь Фурье в соответствии с формулами вычисления интегралов должен выполнить к моменту времени t цифровые преобразования по алгоритму

$$a_{m1} = \frac{2}{T_1}\int_{t-T_1}^{t} u(t)\cos\omega_1 t\,dt\ ;\ b_{m1} = \frac{2}{T_1}\int_{t-T_1}^{t} u(t)\sin\omega_1 t\,dt$$

Также в правой части рисунка 3.10 представлены блоки алгоритма разложения на симметричные составляющие сигналы тока и напряжения.

Получив в цифровом виде значения симметричных составляющих, несложно выполнить фильтровые ЦИО с одной входной воздействующей величиной (тока, напряжения) или с двумя входными величинами (направления мощности, сопротивления).

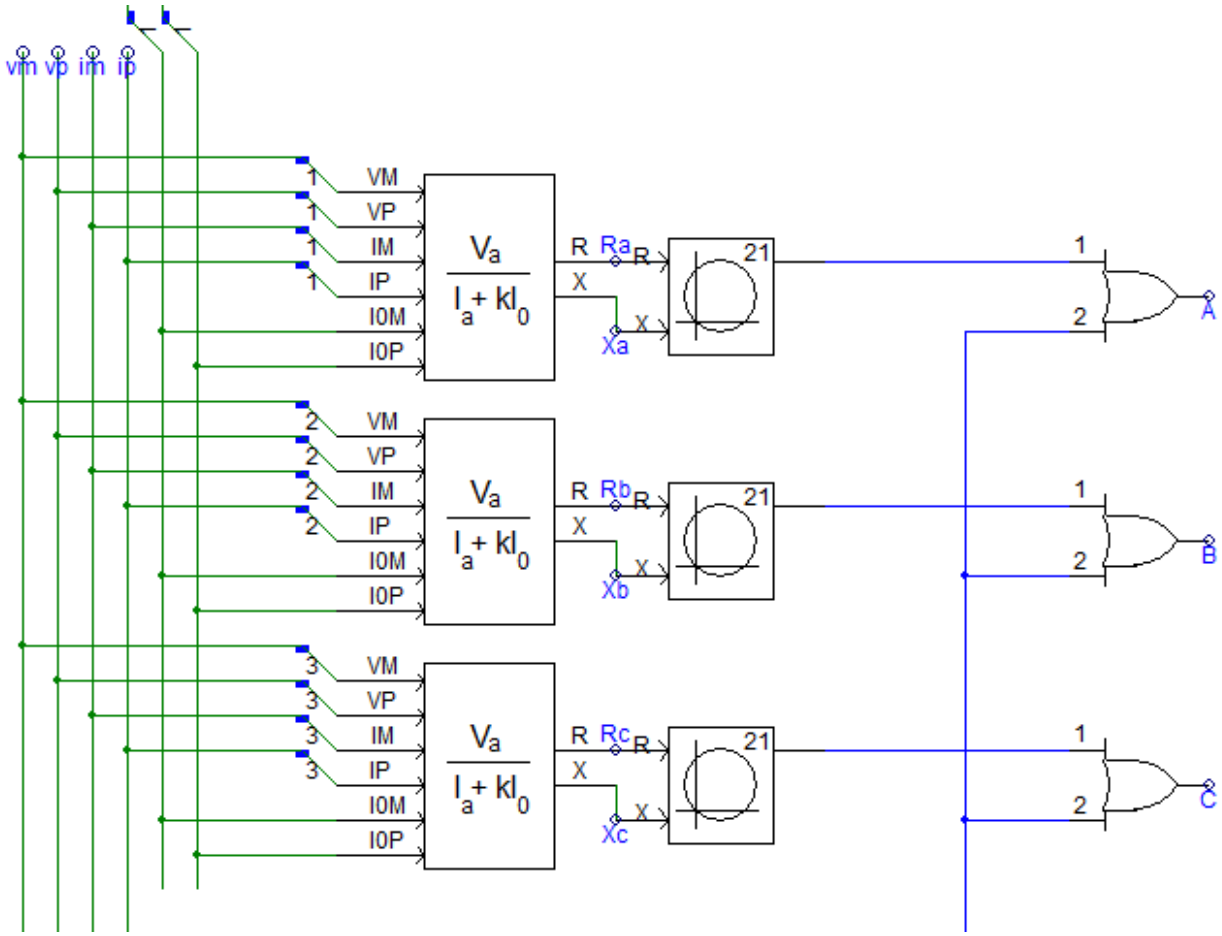


Рисунок 3.11 – Элементы алгоритма цифровой защиты электродвигателя

На рисунках 3.11 и 3.12 приведены элементы вычисления ортогональных составляющих сопротивлений фазных и линейных контуров соответственно.

Наличие вычисленных параметрах векторов напряжения $U(nT_д) = U_x(nT_д) + jU_y(nT_д)$ и тока $I(nT_д) = I_x(nT_д) + jI_y(nT_д)$ позволяет получить требуемые характеристики в плоскости сопротивлений двумя способами:

а) вычислением комплексной величины $Z(nT_д)$ с последующим сравнением этой величины с заданной областью значений в плоскости Z;

б) операциями непосредственно с векторами $U(nT_д)$ и $I(nT_д)$.

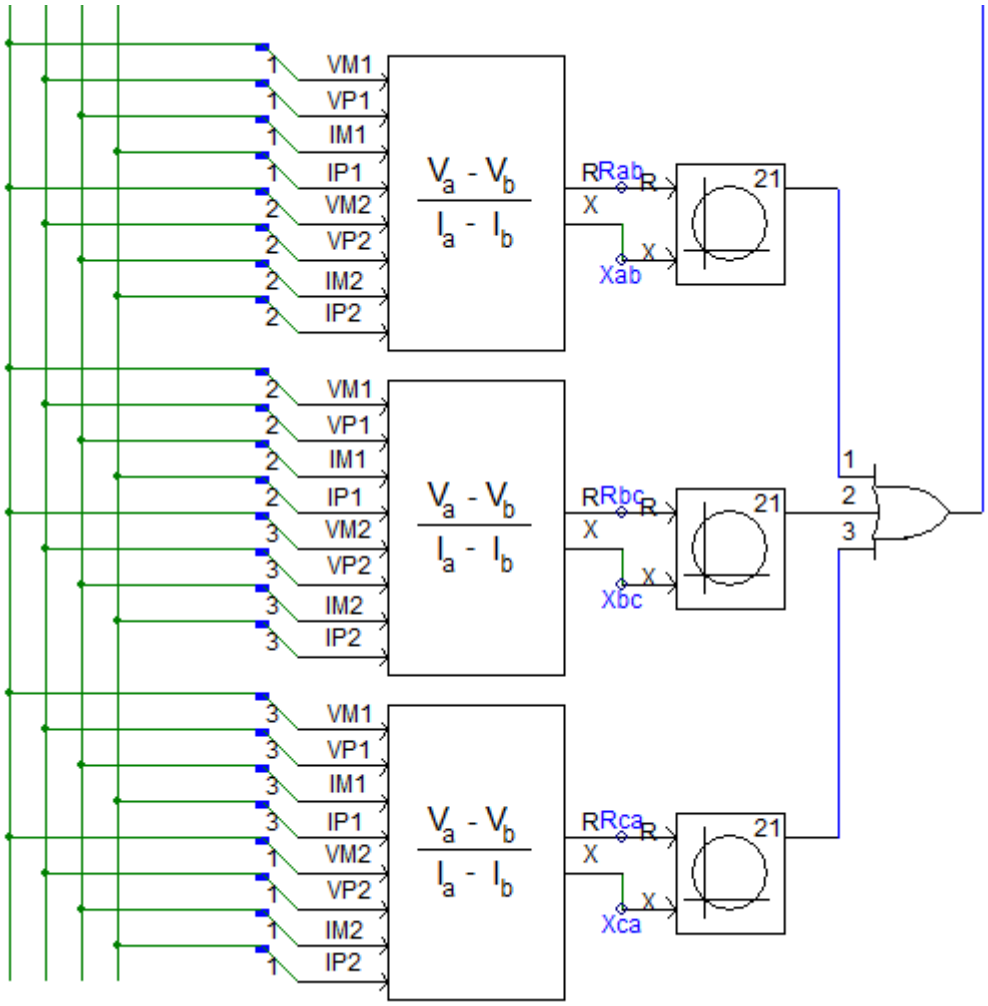


Рисунок 3.12 – Элементы алгоритма цифровой защиты электродвигателя

Реализация первого способа требует вычисления значения Z(nТд) на основе соотношения

$$\underline{Z} = \frac{U_x(nT_д) + jU_y(nT_д)}{I_x(nT_д) + jI_y(nT_д)} = \frac{U_x(nT_д)\cdot I_x(nT_д) + U_y(nT_д)\cdot I_y(nT_д)}{[I_x(nT_д)]^2 + [I_y(nT_д)]^2} +$$

$$+ j\,\frac{U_y(nT_д)\cdot I_x(nT_д) - U_x(nT_д)\cdot I_y(nT_д)}{[I_x(nT_д)]^2 + [I_y(nT_д)]^2} = R(nT_д) + jX(nT_д)\,.$$

При реализации второго способа получение характеристик ЦИО в виде прямых, окружностей и их комбинаций является результатом непосредственных операций с ортогональными составляющими векторов $U(nT_д)$ и $I(nT_д)$ без вычисления Z.

Так же присутствуют элемент задания уставок дистанционного органа в виде круговых характеристик (блоки 21 на рисунках). Результат работы виден на рисунке 3.13. Можно заметить некорректную работу защиты в части попадания сопротивления одной из фаз в область срабатывания, заданную круговой диаграммой.

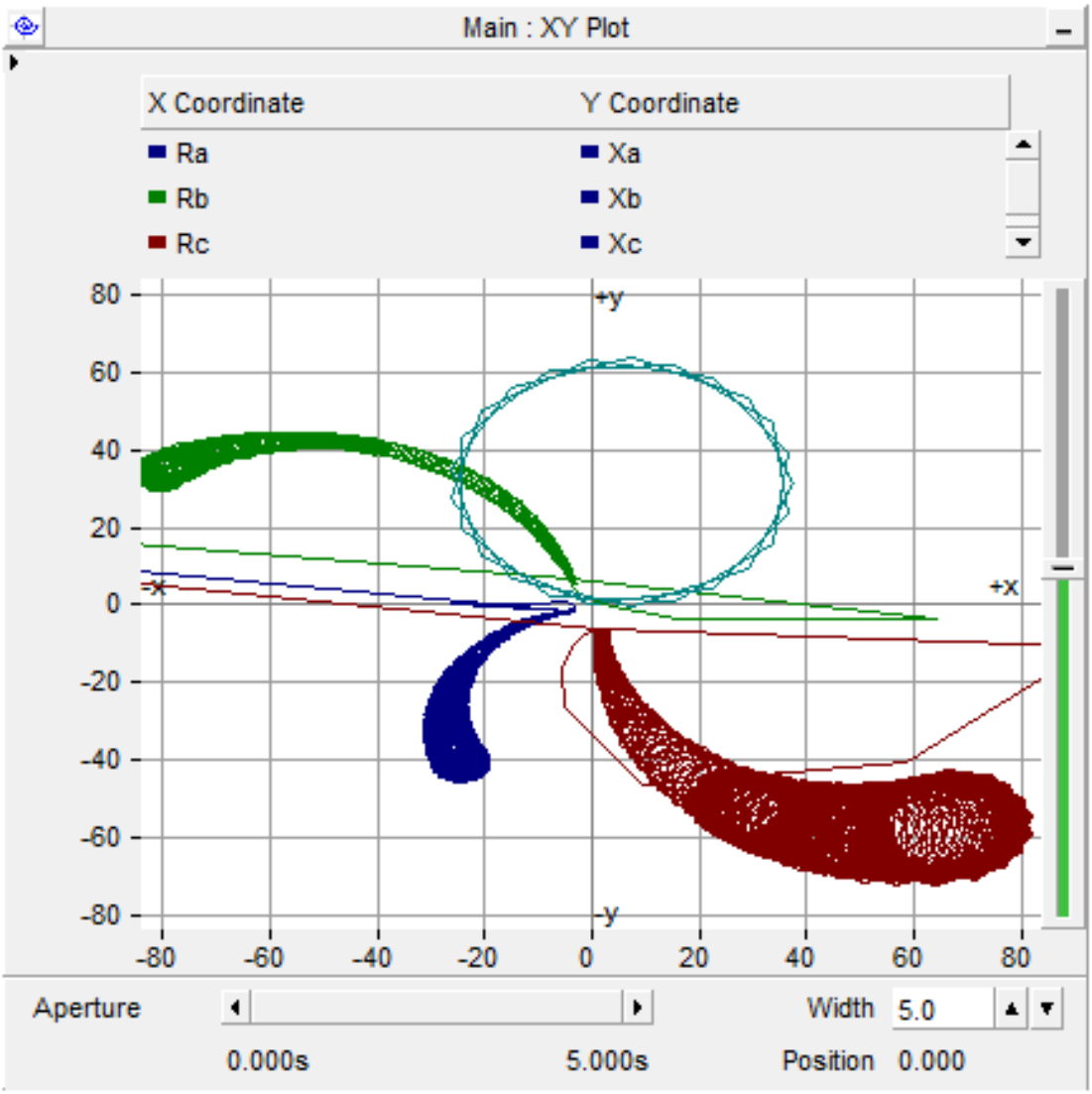


Рисунок 3.13 – Осциллограммы сопротивлений фазных контуров AN, BN и CN электродвигателя с круговой защитной характеристикой

Для создания и симуляции проектов в PSCAD необходим компилятор FORTRAN. В настоящее время поддерживаются следующие коммерчески доступные компиляторы: Intel Visual Fortran 9.x, 10.x, 11.x, 12.x & 13.x 1; GFortran 95.

Скомпилированный код модели в компиляторе GFortran 95 представлен в приложении А.

Выводы к 3-й главе

В данной главе рассмотрен инструментарий для моделирования как участка сети, содержащего асинхронный электродвигатель, так и средства измерения и защиты. Выбрана среда моделирования PSCAD - это мощный и гибкий графический пользовательский интерфейс для всемирно известного модуля электромагнитного моделирования переходных процессов.

Выполнено подробное описание экспериментальной Модель участка сети и асинхронного электродвигателя, составленной в среде PSCAD.

В результате выполненного моделирования работы электродвигателя в различных режимах получены Осциллограммы токов и напряжений на зажимах электродвигателя и осциллограммы мгновенных сопротивлений фазных контуров AN, BN и CN и линейных контуров AB, BC и CA электродвигателя с круговой защитной характеристикой.

По данным эксперимента фиксированы значения токов и сопротивлений, нарушающих рассчитанные по методикам значения уставок токовой защиты и моделируемой дистанционной защиты электродвигателя. Данный факт подтверждает факт некорректности функционирования защит электродвигателей в периоды стартов.

4. КОНЦЕПЦИЯ ФУНКЦИОНИРОВАНИЯ ЗАЩИТ ЭЛЕКТРОДВИГАТЕЛЕЙ НА ОСНОВЕ ОЦЕНКИ МГНОВЕННОГО ЗНАЧЕНИЯ СОПРОТИВЛЕНИЯ

4.1 Оценка мгновенного значения сопротивления

Пуск двигателя сопровождается как правили повышением силы тока в зависимости от мощности двигателя и условий пуска. Такое повышение тока выше номинального нагрузочного тока может составлять от 2 до 15 номинальных токов двигателя. Защиты двигателя как правило блокируются в момент пуска на определенный временной промежуток. Или могут иметь защитную характеристику повторяющую график изменения пускового тока во времени с некоторым запасом на погрешности измерительных трансформаторов тока и температурные изменения сопротивлений обмоток двигателя и других проводящих частей контура.

Любые изменения условий пуска или самого двигателя, например, после ремонта или перемотки предполагают корректировку уставок защит и дополнительную настройку.

Предлагаемая концепция совершенствования алгоритма защиты электродвигателей состоит в оперативной корректировке уставки защитной характеристики во время пуска с характерным снижением сопротивлением двигателя на величину пропорциональную мгновенному значению сопротивления, умноженную на постоянный калибровочный коэффициент.

Таким образом защита адаптируется к изменению сопротивления двигателя в пусковых режимах и всегда остается в работе, отслеживая возможные ситуации замыканий в течение процедуры пуска.

Для проверки данной концепции используем данные из модельного математического эксперимента в PSCAD, позволяющего получить осциллограммы токов и напряжений трехфазного асинхронного двигателя большой мощности. Применяемого на станциях и других энергоемких производствах и требующих не только высокой производительности от таких машин, но и чувствительности и селективности от защит электродвигателей.

Разработаем алгоритм функционирования защиты по предлагаемой концепции и реализуем его в среде графического программирования Labview.

Алгоритм представлен на рисунке 4.1.

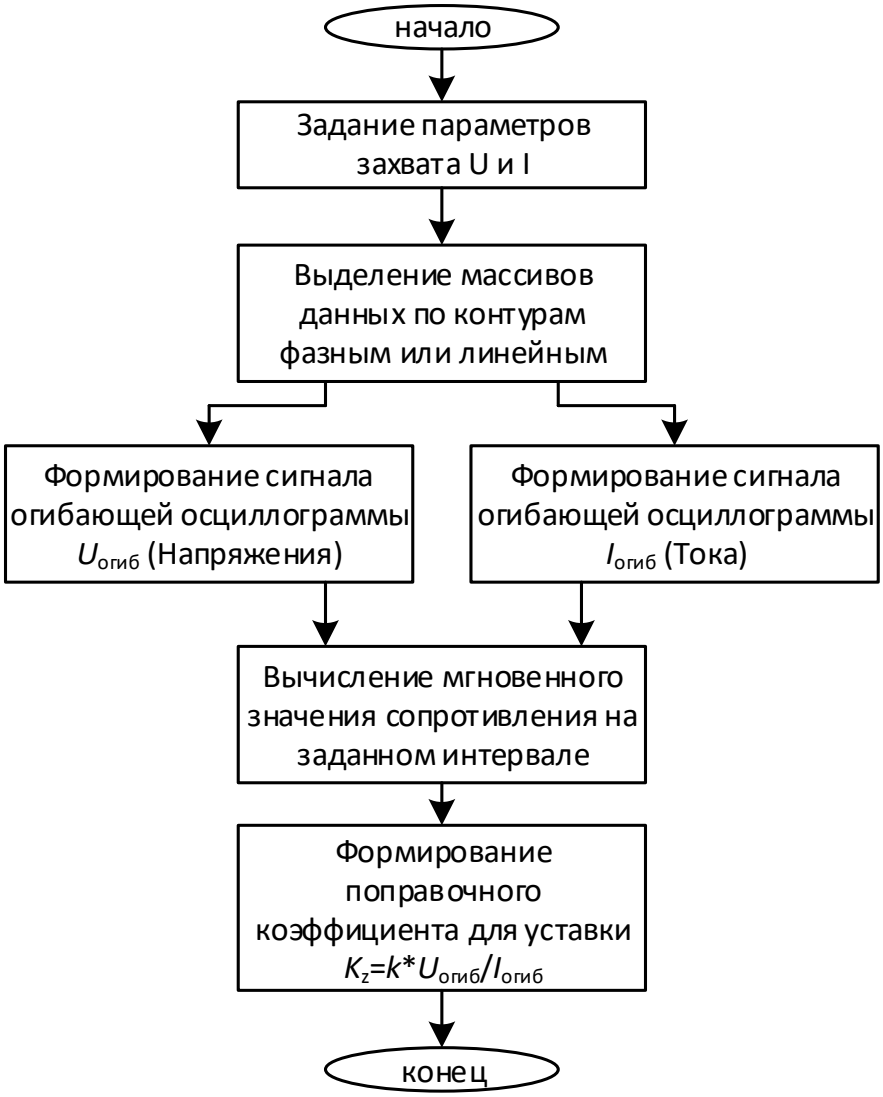


Рисунок 4.1 – Алгоритм вычисления корректирующего коэффициента

Для поддержки компьютерного моделирования были разработаны различные пакеты прикладной математики, например, Mathcad, MATLAB, LabVIEW, VisSim и другие. Они позволяют создавать блочные и формальные модели простых или же сложных процессов и устройств, легко меняя параметры моделей в ходе моделирования. Модели чаще всего представлены графическими блоками, соединение и набор, которых задаются диаграммой модели.

Программа LabVIEW состоит из двух частей:

- блочной диаграммы, которая описывает логику работы виртуального прибора;
- лицевой панели, которая описывает внешний интерфейс виртуального прибора.

Для построения виртуальных приборов могут использоваться комплект более простых приборов. В конечном итоге из базовых элементов можно создать большой, сложный и многофункциональный прибор, который может отвечать за целые системы.

Лицевая панель виртуального прибора содержит средства ввода-вывода: кнопки, переключатели, шкалы, верньеры, информационные табло, и так далее. Эти средства ввода используются пользователем для контроля и управления прибором.

Блочная диаграмма содержит функциональные узлы, которые являются источниками, приемниками и средствами обработки данных. Также компонентами блочной диаграммы являются терминалы («задние контакты» объектов лицевой панели) и управляющие структуры. Функциональные узлы и терминалы объединены в общую систему схему линиями связей.

Далее на рисунке 4.2 представлена блок –диаграмма программы реализующая алгоритм вычисления корректирующего коэффициента, приведенного на рисунке 4.1.

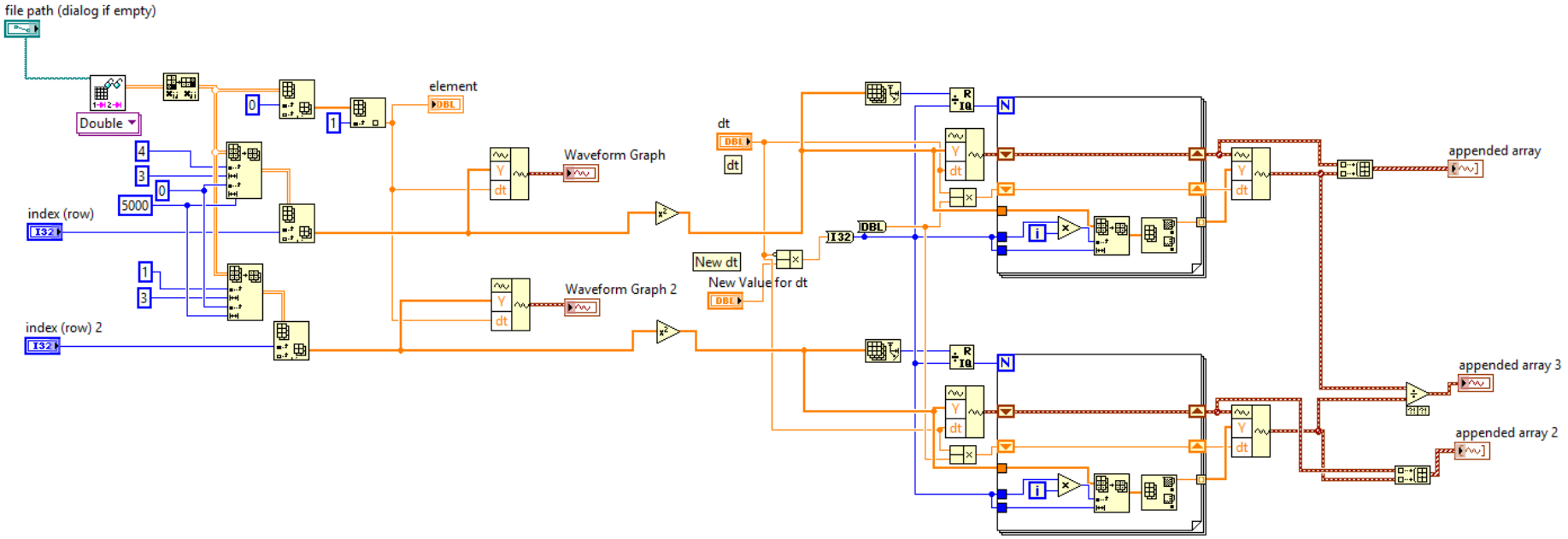


Рисунок 4.2 – Блок диаграмма программы реализации алгоритма вычисления корректирующего коэффициента, приведенного на рисунке 4.1

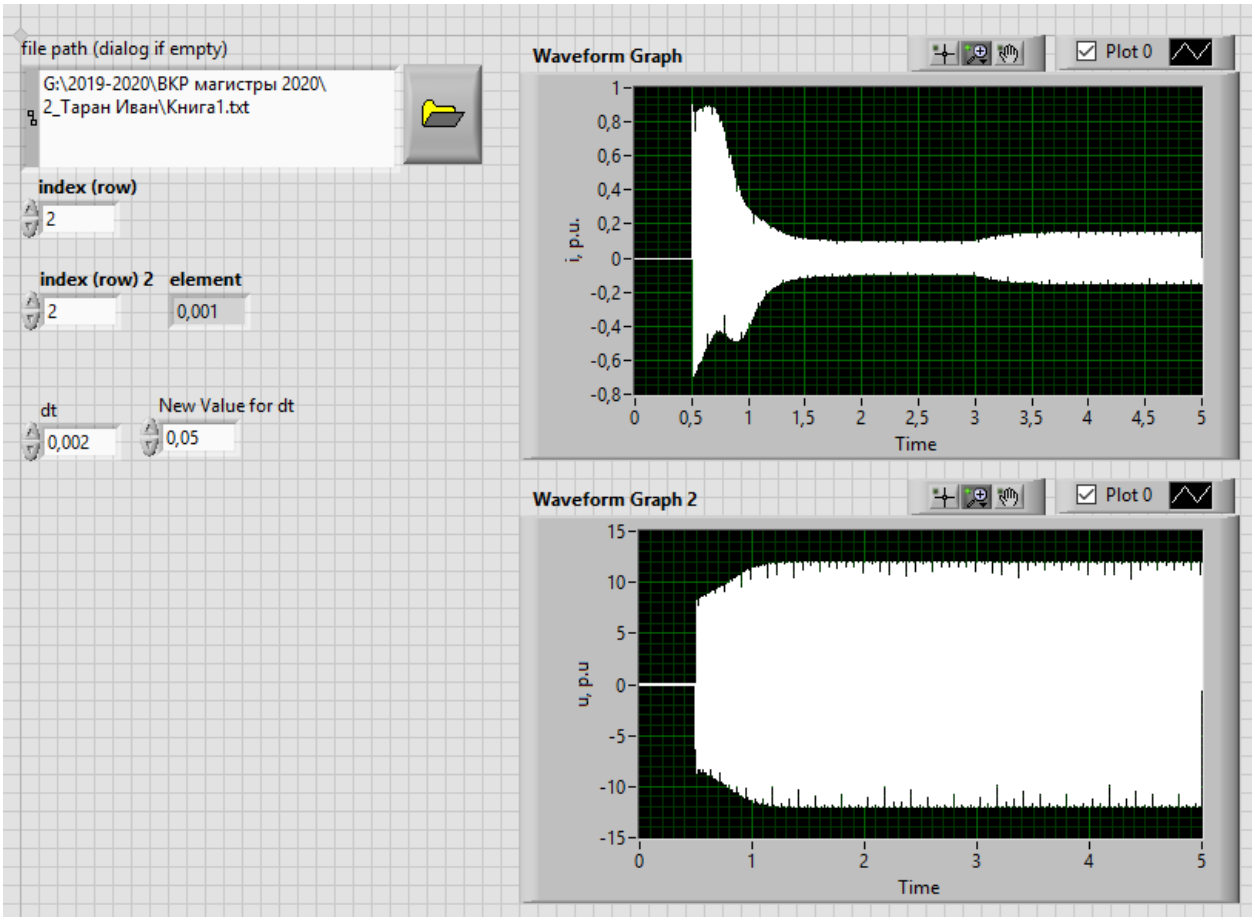


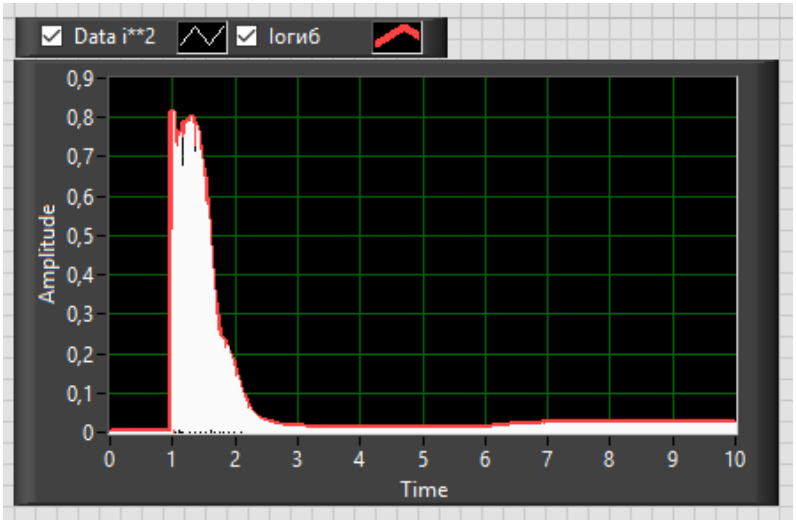
Рисунок 4.3 – Лицевая панель программы реализации алгоритма вычисления корректирующего коэффициента

Работа программы представляет собой параллельную обработку сигналов тока и напряжения в онлайн режиме независимо от прочих алгоритмов защиты и диагностики двигателя. Предполагает задание интервала времени или отсчетов измерений для вычисления значений огибающего сигнала на заданном интервале для сигналов тока и напряжения по заданному контуру в онлайн режиме.

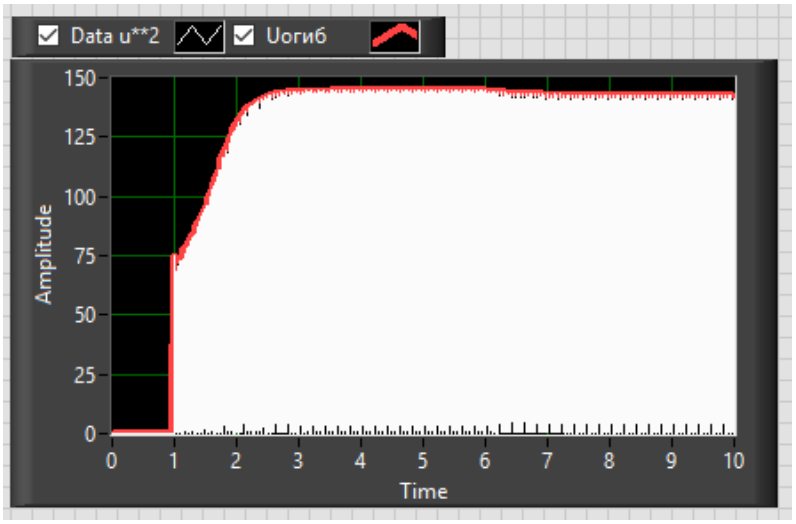
На рисунках 4.4 а, б приведены результаты работы программы по формированию огибающих сигналов тока и напряжения соответственно. Рисунок 4.4 в иллюстрирует изменение вычисляемого адаптивного коэффициента коррекции защитной характеристики двигателя по выражению:

$$K_z = k * U_{огиб} / I_{огиб},$$

Где k – постоянный калибровочный коэффициент, $U_{огиб}$, $I_{огиб,}$ – огибающие.



а)



б)

в)

Рисунок 4.4 – изменение вычисляемого адаптивного коэффициента коррекции защитной характеристики двигателя

4.2 Требования к применению разработанного алгоритма функционирования защит двигателей

Реализация данного алгоритма формирования адаптивного коэффициента коррекции защитной характеристики двигателя возможна на любом современном терминале релейной защиты и автоматики с применением стандартных технологий формирования программных кодов, функционирующих в условиях реального времени.

Так же данный алгоритм возможно реализовать на терминалах, поддерживающих технологию свободно-программируемой логики.

Функции (СПЛ) свободно-программируемой логики для терминалов защиты электродвигателя реализуют функции свободно-программируемой логики

Свободно-программируемая логика (СПЛ) позволяет реализовать дополнительные цепи взаимодействия основных функций РЗА, организовать специальное взаимодействие дискретных входов и выходных реле, а также реализовать новые функции, требуемые по условиям эксплуатации.

Так, например, для терминалов БЭМП РУ программа свободно-программируемой логики LD editor распространяется бесплатно, а также поставляется совместно с терминалами БЭМП РУ в составе программы настройки и мониторинга BempExplorer.

Выводы к 4-й главе

В данной главе разработана концепция функционирования защит электродвигателей на основе оценки мгновенного значения сопротивления.

Данная концепция состоит в оперативной корректировке уставки защитной характеристики во время пуска с характерным снижением сопротивлением двигателя на величину пропорциональную мгновенному значению сопротивления, умноженную на постоянный калибровочный коэффициент.

Таким образом защита адаптируется к изменению сопротивления двигателя в пусковых режимах и всегда остается в работе, отслеживая возможные ситуации замыканий в течение процедуры пуска.

Разработан алгоритм вычисления корректирующего коэффициента, и его программная реализация в среде моделирования Labview.

Работа программы представляет собой параллельную обработку сигналов тока и напряжения в онлайн режиме независимо от прочих алгоритмов защиты и диагностики двигателя. Предполагает задание интервала времени или отсчетов измерений для вычисления значений огибающего сигнала на заданном интервале для сигналов тока и напряжения по заданному контуру в онлайн режиме.

Разработаны требования к применению разработанного алгоритма функционирования защит двигателей.

Выводы

Выводы к 1-й главе

В данной главе проведен анализ современного состояния проблемы функционирования защит электродвигателей. Традиционные средства релейной защиты электродвигателей развивались исторически параллельно с электромеханическими измерительными механизмами измерительных приборов па базе общей теории применительно к стационарным входным сигналам Поэтому большинство алгоритмов традиционных защит основано на контроле интегральных (действующих или средних) значений токов и напряжений Этот стационарный подход требует длительного наблюдения за процессами в аварийных ситуациях для принятия правильною решения о состоянии контролируемого объекта

Установлено, что наиболее опасными являются многофазные и витковые короткие замыкания в обмотке статора, приводящие к разрушению двигателя. Для двигателей с напряжением от 6 кВ и выше работающих в сети с изолированной нейтралью возможно возникновение двойных КЗ, для которых одна точка повреждения может находиться во внешней сети, а вторая в двигателе. Такие повреждения также являются опасными для двигателя и требуют отключения с минимально достижимой выдержкой времениОбзор проблемы функционирования защит АД

В основном все микропроцессорные устройства релейной защиты двигателей устанавливаются в релейных отсеках КРУ, КРУН и КСО, на панелях и в шкафах в релейных залах и пультах управления электростанций и подстанций 6–35 кВ и являются комбинированными микропроцессорными терминалами релейной защиты и автоматики

Во 2-й главе выполнен анализ алгоритмов измерительных органов цифровых защит, Алгоритмов вычисления векторов по мгновенным значениям синусоидальных величин. Также проведен анализ

функционирования алгоритмов использования выборок мгновенных значений для получения требуемых характеристик ЦИО и использования при построении алгоритма повышения чувствительности.

Рассмотрены принципы функционирования цифровых реле тока и напряжения, цифровых реле сопротивления, алгоритм токовой отсечки.

В 3-й главе рассмотрен инструментарий для моделирования как участка сети, содержащего асинхронный электродвигатель, так и средства измерения и защиты. Выбрана среда моделирования PSCAD - это мощный и гибкий графический пользовательский интерфейс для всемирно известного модуля электромагнитного моделирования переходных процессов.

Выполнено подробное описание экспериментальной Модель участка сети и асинхронного электродвигателя, составленной в среде PSCAD.

В результате выполненного моделирования работы электродвигателя в различных режимах получены Осциллограммы токов и напряжений на зажимах электродвигателя и осциллограммы мгновенных сопротивлений фазных контуров AN, BN и CN и линейных контуров AB, BC и CA электродвигателя с круговой защитной характеристикой.

По данным эксперимента фиксированы значения токов и сопротивлений, нарушающих рассчитанные по методикам значения уставок токовой защиты и моделируемой дистанционной защиты электродвигателя. Данный факт подтверждает факт некорректности функционирования защит электродвигателей в периоды стартов.

В 4-й главе разработана концепция функционирования защит электродвигателей на основе оценки мгновенного значения сопротивления.

Данная концепция состоит в оперативной корректировке уставки защитной характеристики во время пуска с характерным снижением сопротивлением двигателя на величину пропорциональную мгновенному

значению сопротивления, умноженную на постоянный калибровочный коэффициент.

Таким образом защита адаптируется к изменению сопротивления двигателя в пусковых режимах и всегда остается в работе, отслеживая возможные ситуации замыканий в течение процедуры пуска.

Разработан алгоритм вычисления корректирующего коэффициента, и его программная реализация в среде моделирования Labview.

Работа программы представляет собой параллельную обработку сигналов тока и напряжения в онлайн режиме независимо от прочих алгоритмов защиты и диагностики двигателя. Предполагает задание интервала времени или отсчетов измерений для вычисления значений огибающего сигнала на заданном интервале для сигналов тока и напряжения по заданному контуру в онлайн режиме.

Разработаны требования к применению разработанного алгоритма функционирования защит двигателей.

Список использованных источников

1. С. Николовский, П. Марич, и Lj. Майданджич, «Интеграция солнечной электростанции в распределительную сеть», Международный журнал по электротехнике и вычислительной технике - IJECE, vol. 5, нет 4, с. 656-668, август 2015 г.

2. С. Николовский, П. Марич, Г. Кнежевич, «Компьютерное моделирование и имитация уставок реле максимального тока солнечной электростанции», Журнал фундаментальных и прикладных исследований, International Knowledge Press, vol.11, no. 1, 2015

3. Х. Фунмилайо, Дж. Сильва и К. Батлер-Пурри, «Защита от перегрузки по току для радиального испытательного устройства подачи на 34 узла», IEEE Transactions on Power Delivery, vol. 27, нет. 2 апреля 2012 г.

4. Д. Бирла, Р. П. Махешвари и Х. О. Гупта, "Реле времени сверхтоковой координации: обзор", Международный журнал развивающихся электроэнергетических систем, вып. 2, нет 2 апреля. 2005.

5. A. Zamani, T. Sidhu, A. Yazdani, "Стратегия координации защиты в радиальных распределительных сетях с распределенными генераторами", IEEE Общее собрание энергетического общества, том. 10, с. 263-270, 2010.

6. С. Николовский, М. Гавранек и П. Марич, «Координация числовой релейной защиты с использованием программного обеспечения для моделирования», в материалах MIPRO Computers in Technical Systems, 2013, стр. 1107-1111.

7. Правила устройства электроустановок. 7-е издание. - М.: Энергоатомиздат, 2003.

8. Общие технические требования к микропроцессорным устройствам защиты и автоматики энергосистем. РД 34.35.310-97, - М.: ОРГРЭС, 1997.

9. Шнеерсон Э.М. Дистанционные защиты/ Э.М. Шнеерсон. – М.: Энерго-

атомиздат, 1986. – 448 с.

10. Микропроцессорные устройства защиты: руководство по эксплуатации / Радиус-Автоматика. – М.: 2006 ÷ 2010.

11. Корогодский В.И., Кужеков С.Л., Паперно Л.Б. Релейная защита электродвигателей напряжением выше 1 кВ, - М.: Энергоатомиздат, 1987.

12. Федосеев А.М. Релейная защита электроэнергетических систем. Релейная защита сетей: учеб. пособие для вузов /А.М. Федосеев. – М.: Энергоатомиздат, 1984. – 520 с.

13. Шмурьев В.Я. Цифровые реле: учеб. пособие / В.Я. Шмурьев. – СПб.: Изд-во ПЭИпк, 1998. – 81 с.

14. RELPOL S.A.: Electromagnetic relays / Catalogue 2001/2002. – 137 с.

15. Протон-Импульс: оптоэлектронные компоненты коммутации и контроля. – М.: Издат. дом "Додэка-XXI", 2001. – 64 с.

16. Пуляев В.И. Цифровая регистрация аварийных событий в энергосисте-

мах / В.И. Пуляев, Ю.В. Усачёв. – М.: НТФ "Прогресс", 1999. – 72 с.

17. Ерофеев Ю.Н. Импульсные устройства: учеб. пособие для вузов по спец. "Радиотехника"/Ю.Н. Ерофеев. – М.: Высш. шк., 1989. – 527 с.

18. Браммер Ю. А. Импульсные и цифровые устройства: Учеб. пособие для студентов электроприборостроительных сред. учеб. завед. / Ю. А. Браммер, И. Н. Пащук. 7-е изд., перераб. и доп. – М.: Высш. шк., 2003. – 351 с.

19. DESIGN-IN REFERENCE MANUAL: DATA CONVERTERS, AMPLIFIERS, SPECIAL LINER PRODUCTS, SUPPORT COMPONENTS. Analog Devices, Inc., 1994. – С. 2-375-2-382.

20. Гришанов В.Г. Основы микропроцессорной техники: учеб. пособие / В.Г. Гришанов, А.А. Никитин. – Чебоксары: Изд-во Чуваш. ун-та, 2003. – 108 с.

21. Лебедев Е.К. Микропроцессорные устройства и системы: учеб. пособие / Е.К. Лебедев. – Чебоксары: Изд-во Чуваш. ун-та, 2000. – 240 с.

22. Лачин В.И. Электроника: учеб. пособие / В.И. Лачин, Н.С. Савёлов. 3-е изд., перераб. и доп. – Ростов н/Д.: Феникс, 2002. – 576 с.

23. Найвельт Г.С. Источники вторичного электропитания: справ. / Г.С.Найвельт, К.Б. Мазель, Ч.И. Хусаинов и др.; под ред. Г.С. Найвельта. – М.: Радио и связь, 1986. – 576 с.

24. Интегральные микросхемы: перспективные изделия. Вып. 3. – М.: Издат. дом "Доддэка", 1997. – 96 с.

25. Никитин А.А. Аналоговые интегральные элементы электронных и микропроцессорных электрических аппаратов: учеб. пособие / А.А. Никитин. – Чебоксары: Изд-во Чуваш. ун-та. 2011. – 258 с.

26. Никитин А.А. Микропроцессорные реле: учеб. пособие /А.А. Никитин. – Чебоксары: Изд-во Чуваш. ун-та, 2006. – 448 с.

27. Атабеков Г.И. Основы теории цепей: Учебник для вузов / Г.И. Атабеков. – М.: Энергия, 1969. – 424 с.

28. Ванин В.К. Релейная защита на элементах вычислительной техники / В.К. Ванин, Г.М. Павлов. – Л.: Энергоатомиздат. Ленингр. отд-е, 1983. – 206 с.

29. Никитин А.А. Аппараты релейной защиты: учеб. пособие /А.А. Никитин, Э.М. Шнеерсон. – Чебоксары: Изд-во Чуваш. ун-та. 2008. – 524 с.

```
!=======================================================================
! Generated by  : PSCAD v4.5.0.0
!
! Warning:  The content of this file is automatically generated.
!           Do not modify, as any changes made here will be lost!
!-----------------------------------------------------------------------
! Component     : Main
! Description   :  EXAMPLE 7
!-----------------------------------------------------------------------


!=======================================================================

      SUBROUTINE MainDyn()

!---------------------------------------
! Standard includes
!---------------------------------------

      INCLUDE 'nd.h'
      INCLUDE 'emtconst.h'
      INCLUDE 'emtstor.h'
      INCLUDE 's0.h'
      INCLUDE 's1.h'
      INCLUDE 's2.h'
      INCLUDE 's4.h'
      INCLUDE 'branches.h'
      INCLUDE 'pscadv3.h'
      INCLUDE 'fnames.h'
      INCLUDE 'radiolinks.h'
      INCLUDE 'matlab.h'
      INCLUDE 'rtconfig.h'

!---------------------------------------
! Function/Subroutine Declarations
!---------------------------------------

!     SUBR    EMTDC_X2COMP  ! 'Comparator with Interpolation'

!---------------------------------------
! Variable Declarations
!---------------------------------------


! Subroutine Arguments

! Electrical Node Indices
      INTEGER  NT_11(3)

! Control Signals
      INTEGER  DIST, IT_1, BRK, IT_2, IT_3, IT_4
      INTEGER  IT_5, IT_6, IT_7, StoT, A, B, C
      INTEGER  IT_8
      REAL     RT_1, RT_2, Tload, Ix(3), RT_3
      REAL     RT_4, Rrotor, Te, Vload(3)
      REAL     Iload(3), Vrms, V_ad(3), I_ad(3)
      REAL     RT_5, RT_6, vp(3), vm(3), ip(3)
      REAL     im(3), Xa, Xb, Xc, Ra, Rb, Rc, Xab
      REAL     Xbc, Xca, Rab, Rbc, Rca, RT_7
      REAL     iseqm(3), iseqp(3), WIN, W, TIN
      REAL     vam, vbm, vcm, vap, vbp, vcp, vpm
      REAL     vnm, vzm, vpp, vnp, vzp, iam, ibm
      REAL     icm, iap, ibp, icp, ipm, inm, izm
      REAL     ipp, inp, izp, vseqm(3), vseqp(3)
      REAL     Rcircle, Xcircle, RT_8, RT_9
      REAL     RT_10, RT_11, RT_12, RT_13(7)
      REAL     RT_14(7), RT_15(7), RT_16, RT_17
      REAL     RT_18, RT_19(7), RT_20(7)
      REAL     RT_21(7), RT_22(7), RT_23(7)
      REAL     RT_24(7), RT_25, RT_26, RT_27
```

```
      REAL     RT_28(7), RT_29(7), RT_30(7)
      REAL     RT_31, RT_32, RT_33, RT_34, RT_35
      REAL     RT_36, RT_37, RT_38, RT_39, RT_40
      REAL     RT_41, RT_42, RT_43, RT_44, RT_45
      REAL     RT_46, RT_47, RT_48, RT_49, RT_50
      REAL     RT_51, RT_52, RT_53, RT_54, RT_55
      REAL     RT_56, RT_57, RT_58, RT_59, RT_60
      REAL     RT_61, RT_62, RT_63, RT_64, RT_65
      REAL     RT_66, RT_67, RT_68, RT_69, RT_70
      REAL     RT_71, RT_72, RT_73, RT_74, Ib, Ic

! Internal Variables
      INTEGER  IVD1_1, IVD1_2, IVD1_3, IVD1_4
      REAL     RVD1_1, RVD1_2, RVD2_1(2), RVD1_3
      REAL     RVD1_4, RVD1_5, RVD1_6, RVD1_7

! Indexing variables
      INTEGER ICALL_NO                          ! Module call num
      INTEGER ISTOI, ISTOF, IT_0                ! Storage Indices
      INTEGER ICX, IPGB                         ! Control/Monitoring
      INTEGER ISUBS, SS(1), IBRCH(1), INODE     ! SS/Node/Branch/Xfmr
      INTEGER IXFMR


!---------------------------------------
! Local Indices
!---------------------------------------

! Dsdyn <-> Dsout transfer index storage

      NTXFR = NTXFR + 1

      TXFR(NTXFR,1) = NSTOL
      TXFR(NTXFR,2) = NSTOI
      TXFR(NTXFR,3) = NSTOF
      TXFR(NTXFR,4) = NSTOC

! Increment and assign runtime configuration call indices

      ICALL_NO  = NCALL_NO
      NCALL_NO  = NCALL_NO + 1

! Increment global storage indices

      ISTOI     = NSTOI
      NSTOI     = NSTOI + 14
      ISTOF     = NSTOF
      NSTOF     = NSTOF + 232
      IPGB      = NPGB
      NPGB      = NPGB + 34
      ICX       = NCX
      NCX       = NCX + 3
      INODE     = NNODE + 2
      NNODE     = NNODE + 30
      IXFMR     = NXFMR
      NXFMR     = NXFMR + 9

! Initialize Subsystem Mapping

      ISUBS = NSUBS + 0
      NSUBS = NSUBS + 1

      DO IT_0 = 1,1
         SS(IT_0) = SUBS(ISUBS + IT_0)
      END DO

! Initialize Branch Mapping.

      IBRCH(1)     = NBRCH(SS(1))
      NBRCH(SS(1)) = NBRCH(SS(1)) + 63
!---------------------------------------
```

```
! Transfers from storage arrays
!--------------------------------------

      Tload    = STOF(ISTOF + 3)
      Te       = STOF(ISTOF + 10)
      Vrms     = STOF(ISTOF + 17)
      RT_5     = STOF(ISTOF + 24)
      RT_6     = STOF(ISTOF + 25)
      IT_2     = STOI(ISTOI + 4)
      IT_3     = STOI(ISTOI + 5)
      IT_4     = STOI(ISTOI + 6)
      IT_5     = STOI(ISTOI + 7)
      IT_6     = STOI(ISTOI + 8)
      Xa       = STOF(ISTOF + 38)
      Xb       = STOF(ISTOF + 39)
      Xc       = STOF(ISTOF + 40)
      Ra       = STOF(ISTOF + 41)
      Rb       = STOF(ISTOF + 42)
      Rc       = STOF(ISTOF + 43)
      Xab      = STOF(ISTOF + 44)
      Xbc      = STOF(ISTOF + 45)
      Xca      = STOF(ISTOF + 46)
      Rab      = STOF(ISTOF + 47)
      Rbc      = STOF(ISTOF + 48)
      Rca      = STOF(ISTOF + 49)
      W        = STOF(ISTOF + 58)
      vam      = STOF(ISTOF + 60)
      vbm      = STOF(ISTOF + 61)
      vcm      = STOF(ISTOF + 62)
      vap      = STOF(ISTOF + 63)
      vbp      = STOF(ISTOF + 64)
      vcp      = STOF(ISTOF + 65)
      vpm      = STOF(ISTOF + 66)
      vnm      = STOF(ISTOF + 67)
      vzm      = STOF(ISTOF + 68)
      vpp      = STOF(ISTOF + 69)
      vnp      = STOF(ISTOF + 70)
      vzp      = STOF(ISTOF + 71)
      iam      = STOF(ISTOF + 72)
      ibm      = STOF(ISTOF + 73)
      icm      = STOF(ISTOF + 74)
      iap      = STOF(ISTOF + 75)
      ibp      = STOF(ISTOF + 76)
      icp      = STOF(ISTOF + 77)
      ipm      = STOF(ISTOF + 78)
      inm      = STOF(ISTOF + 79)
      izm      = STOF(ISTOF + 80)
      ipp      = STOF(ISTOF + 81)
      inp      = STOF(ISTOF + 82)
      izp      = STOF(ISTOF + 83)
      RT_16    = STOF(ISTOF + 118)
      RT_17    = STOF(ISTOF + 119)
      RT_18    = STOF(ISTOF + 120)
      RT_25    = STOF(ISTOF + 163)
      RT_26    = STOF(ISTOF + 164)
      RT_27    = STOF(ISTOF + 165)
      RT_31    = STOF(ISTOF + 187)
      RT_32    = STOF(ISTOF + 188)
      RT_33    = STOF(ISTOF + 189)
      RT_34    = STOF(ISTOF + 190)
      RT_35    = STOF(ISTOF + 191)
      RT_36    = STOF(ISTOF + 192)
      RT_37    = STOF(ISTOF + 193)
      RT_38    = STOF(ISTOF + 194)
      RT_39    = STOF(ISTOF + 195)
      RT_40    = STOF(ISTOF + 196)
      RT_41    = STOF(ISTOF + 197)
      RT_42    = STOF(ISTOF + 198)
      IT_8     = STOI(ISTOI + 14)
      RT_43    = STOF(ISTOF + 199)
      RT_44    = STOF(ISTOF + 200)
```

```
      RT_45    = STOF(ISTOF + 201)
      RT_46    = STOF(ISTOF + 202)
      RT_47    = STOF(ISTOF + 203)
      RT_48    = STOF(ISTOF + 204)
      RT_49    = STOF(ISTOF + 205)
      RT_50    = STOF(ISTOF + 206)
      RT_51    = STOF(ISTOF + 207)
      RT_52    = STOF(ISTOF + 208)
      RT_53    = STOF(ISTOF + 209)
      RT_54    = STOF(ISTOF + 210)
      RT_55    = STOF(ISTOF + 211)
      RT_56    = STOF(ISTOF + 212)
      RT_57    = STOF(ISTOF + 213)
      RT_58    = STOF(ISTOF + 214)
      RT_59    = STOF(ISTOF + 215)
      RT_60    = STOF(ISTOF + 216)
      RT_61    = STOF(ISTOF + 217)
      RT_62    = STOF(ISTOF + 218)
      RT_63    = STOF(ISTOF + 219)
      RT_64    = STOF(ISTOF + 220)
      RT_65    = STOF(ISTOF + 221)
      RT_66    = STOF(ISTOF + 222)
      Ib       = STOF(ISTOF + 231)
      Ic       = STOF(ISTOF + 232)

! Array (1:3) quantities...
      DO IT_0 = 1,3
         Ix(IT_0) = STOF(ISTOF + 3 + IT_0)
         Vload(IT_0) = STOF(ISTOF + 10 + IT_0)
         Iload(IT_0) = STOF(ISTOF + 13 + IT_0)
         V_ad(IT_0) = STOF(ISTOF + 17 + IT_0)
         I_ad(IT_0) = STOF(ISTOF + 20 + IT_0)
         vp(IT_0) = STOF(ISTOF + 25 + IT_0)
         vm(IT_0) = STOF(ISTOF + 28 + IT_0)
         ip(IT_0) = STOF(ISTOF + 31 + IT_0)
         im(IT_0) = STOF(ISTOF + 34 + IT_0)
         iseqm(IT_0) = STOF(ISTOF + 50 + IT_0)
         iseqp(IT_0) = STOF(ISTOF + 53 + IT_0)
         vseqm(IT_0) = STOF(ISTOF + 83 + IT_0)
         vseqp(IT_0) = STOF(ISTOF + 86 + IT_0)
      END DO

! Array (1:7) quantities...
      DO IT_0 = 1,7
         RT_13(IT_0) = STOF(ISTOF + 96 + IT_0)
         RT_14(IT_0) = STOF(ISTOF + 103 + IT_0)
         RT_15(IT_0) = STOF(ISTOF + 110 + IT_0)
         RT_19(IT_0) = STOF(ISTOF + 120 + IT_0)
         RT_20(IT_0) = STOF(ISTOF + 127 + IT_0)
         RT_21(IT_0) = STOF(ISTOF + 134 + IT_0)
         RT_22(IT_0) = STOF(ISTOF + 141 + IT_0)
         RT_23(IT_0) = STOF(ISTOF + 148 + IT_0)
         RT_24(IT_0) = STOF(ISTOF + 155 + IT_0)
         RT_28(IT_0) = STOF(ISTOF + 165 + IT_0)
         RT_29(IT_0) = STOF(ISTOF + 172 + IT_0)
         RT_30(IT_0) = STOF(ISTOF + 179 + IT_0)
      END DO

!---------------------------------------
! Electrical Node Lookup
!---------------------------------------


! Array (1:3) quantities...
      DO IT_0 = 1,3
         NT_11(IT_0) = NODE(INODE + 19 + IT_0)
      END DO

!---------------------------------------
! Configuration of Models
!---------------------------------------
```

```
      IF ( TIMEZERO ) THEN
         FILENAME = 'Main.dta'
         CALL EMTDC_OPENFILE
         SECTION = 'DATADSD:'
         CALL EMTDC_GOTOSECTION
      ENDIF
!---------------------------------------
! Generated code from module definition
!---------------------------------------


! 10:[var] Variable Input Slider 'Rrotor'
      Rrotor = CX(CXMAP(ICX+1))

! 20:[tbreakn] Timed Breaker Logic
! Timed breaker logic
      IF ( TIMEZERO ) THEN
         BRK = 1
      ELSE
         BRK = 1
         IF ( TIME .GE. 0.5 ) BRK = (1-1)
      ENDIF

! 30:[const] Real Constant
      WIN = 0.0

! 40:[consti] Integer Constant
      IT_1 = 0

! 60:[breaker3] 3 Phase Breaker 'BRK'
      IVD1_4 = NSTORI
      NSTORI = NSTORI + 3
! Three Phase Breaker
      CALL EMTDC_BREAKER1(SS(1), (IBRCH(1)+31),0.1,1000000.0,RTCF(NRTCF)&
     &,0,NINT(1.0-REAL(BRK)))
      CALL EMTDC_BREAKER1(SS(1), (IBRCH(1)+32),0.1,1000000.0,RTCF(NRTCF)&
     &,0,NINT(1.0-REAL(BRK)))
      CALL EMTDC_BREAKER1(SS(1), (IBRCH(1)+33),0.1,1000000.0,RTCF(NRTCF)&
     &,0,NINT(1.0-REAL(BRK)))
!
      IVD1_1 = 2*E_BtoI(OPENBR( (IBRCH(1)+31),SS(1)))
      IVD1_2 = 2*E_BtoI(OPENBR( (IBRCH(1)+32),SS(1)))
      IVD1_3 = 2*E_BtoI(OPENBR( (IBRCH(1)+33),SS(1)))
      NRTCF = NRTCF + 1
      IF (FIRSTSTEP .OR. (STORI(IVD1_4+0) .NE. IVD1_1)) THEN
         CALL PSCAD_AGI(423392219,IVD1_1,"BOpen1")
      ENDIF
      IF (FIRSTSTEP .OR. (STORI(IVD1_4+1) .NE. IVD1_2)) THEN
         CALL PSCAD_AGI(423392219,IVD1_2,"BOpen2")
      ENDIF
      IF (FIRSTSTEP .OR. (STORI(IVD1_4+2) .NE. IVD1_3)) THEN
         CALL PSCAD_AGI(423392219,IVD1_3,"BOpen3")
      ENDIF
      STORI(IVD1_4+0) = 2*E_BtoI(OPENBR( (IBRCH(1)+31),SS(1)))
      STORI(IVD1_4+1) = 2*E_BtoI(OPENBR( (IBRCH(1)+32),SS(1)))
      STORI(IVD1_4+2) = 2*E_BtoI(OPENBR( (IBRCH(1)+33),SS(1)))

! 70:[const] Real Constant
      RT_11 = 1.8

! 100:[time-sig] Output of Simulation Time
      RT_2 = TIME

! 110:[compare] Single Input Level Comparator
!
!
      CALL EMTDC_X2COMP(0,0,3.0,RT_2,0.0,0.0,1.0,RVD2_1)
      RT_8 = RVD2_1(1)

! 120:[unity] Type conversion block
```

```
! real -> nearest integer
      DIST = NINT(RT_8)

! 130:[var] Variable Input Slider 'Gradient -k'
      RT_4 = CX(CXMAP(ICX+2))

! 140:[var] Variable Input Slider 'Stdstill torq - b'
      RT_3 = CX(CXMAP(ICX+3))

! 150:[time-sig] Output of Simulation Time
      RT_1 = TIME

! 160:[compare] Single Input Level Comparator
!
!
      CALL EMTDC_X2COMP(0,0,0.0001,RT_1,0.0,0.0,0.0,RVD2_1)
      RT_9 = RVD2_1(1)

! 170:[unity] Type conversion block
! real -> nearest integer
      StoT = NINT(RT_9)

! 250:[const] Real Constant
      RT_72 = 60.0

! 260:[const] Real Constant
      RT_74 = 5.5

! 280:[const] Real Constant
      RT_71 = 32.0

! 290:[signalgen] Signal Generator /w Interpolation
      CALL COMPONENT_ID(ICALL_NO,1628879709)
      CALL E_XSGEN1_EXE(1,60.0,RVD2_1)
      RT_7 = RVD2_1(1)

! 300:[const] Real Constant
      RT_67 = 32.0

! 310:[const] Real Constant
      RT_68 = 60.0

! 320:[const] Real Constant
      RT_70 = 31.5

! 450:[modulator] Amplitude/Frequency/Phase Modulator
! AM/FM/PM MODULATOR
      RT_73 = RT_71 * COS(STORF(NSTORF) + RT_7*PI_BY180)
      STORF(NSTORF) = STORF(NSTORF) + TWO_PI*RT_72*DELT
      IF (STORF(NSTORF) .GT.  TWO_PI)  STORF(NSTORF) = STORF(NSTORF) - T&
     &WO_PI
      IF (STORF(NSTORF) .LT. -TWO_PI)  STORF(NSTORF) = STORF(NSTORF) + T&
     &WO_PI
      NSTORF = NSTORF+1
!

! 460:[sumjct] Summing/Differencing Junctions
      Rcircle = + RT_73 + RT_74

! 520:[modulator] Amplitude/Frequency/Phase Modulator
! AM/FM/PM MODULATOR
      RT_69 = RT_67 * SIN(STORF(NSTORF) + RT_7*PI_BY180)
      STORF(NSTORF) = STORF(NSTORF) + TWO_PI*RT_68*DELT
      IF (STORF(NSTORF) .GT.  TWO_PI)  STORF(NSTORF) = STORF(NSTORF) - T&
     &WO_PI
      IF (STORF(NSTORF) .LT. -TWO_PI)  STORF(NSTORF) = STORF(NSTORF) + T&
     &WO_PI
      NSTORF = NSTORF+1
!

! 530:[sumjct] Summing/Differencing Junctions
```

```
      Xcircle = + RT_69 + RT_70

! 890:[select] Two Input Selector
      IF (DIST .EQ. RTCI(NRTCI)) THEN
         TIN = RT_11
      ELSE
         TIN = Tload
      ENDIF
      NRTCI = NRTCI + 1
!

! 900:[wound_rotor] Wound Rotor Induction Machine 'WR'
      IVD1_1 = NEXC
      CALL COMPONENT_ID(ICALL_NO,708257837)
      CALL INDUCMC1_EXE(SS(1), (IBRCH(1)+1), (IBRCH(1)+2), (IBRCH(1)+3),&
     & (IBRCH(1)+4), (IBRCH(1)+5), (IBRCH(1)+6), (IBRCH(1)+7), (IBRCH(1)&
     &+8), (IBRCH(1)+9), (IBRCH(1)+10), (IBRCH(1)+11), (IBRCH(1)+12),WIN&
     &,IT_1,TIN)
      W = STOR(IVD1_1+71)
      Te = STOR(IVD1_1+61)

! 910:[square] Square
      RT_12 = W * W

! 920:[mult] Multiplier
      RT_10 = RT_12 * RT_4

! 930:[sumjct] Summing/Differencing Junctions
      Tload = + RT_10 + RT_3

! 1150:[pgb] Output Channel 'Xcircle'

      PGB(IPGB+22) = Xcircle

! 1210:[pgb] Output Channel 'Rcircle'

      PGB(IPGB+27) = Rcircle

! 1280:[logic_mult] Multiple Input Logic Gate
!
! Multi input OR gate
!
      IF ( (IT_3 .NE. 0) .OR. (IT_8 .NE. 0) .OR. (IT_2 .NE. 0) ) THEN
        IT_7 = 1
      ELSE
        IT_7 = 0
      ENDIF

! 1290:[pgb] Output Channel 'Tload'

      PGB(IPGB+31) = Tload

! 1300:[logic_mult] Multiple Input Logic Gate
!
! Multi input OR gate
!
      IF ( (IT_6 .NE. 0) .OR. (IT_7 .NE. 0) ) THEN
        C = 1
      ELSE
        C = 0
      ENDIF

! 1310:[pgb] Output Channel 'Speed'

      PGB(IPGB+32) = W

! 1320:[pgb] Output Channel 'Electric torque'

      PGB(IPGB+33) = Te

! 1330:[logic_mult] Multiple Input Logic Gate
```

```
!
! Multi input OR gate
!
      IF ( (IT_5 .NE. 0) .OR. (IT_7 .NE. 0) ) THEN
        B = 1
      ELSE
        B = 0
      ENDIF

! 1340:[logic_mult] Multiple Input Logic Gate
!
! Multi input OR gate
!
      IF ( (IT_4 .NE. 0) .OR. (IT_7 .NE. 0) ) THEN
        A = 1
      ELSE
        A = 0
      ENDIF

! 1350:[pgb] Output Channel 'Torque Input'

      PGB(IPGB+34) = TIN

! 1360:[varrlc] Variable R, L or C
      CALL E_VARRLC1_EXE(0 ,SS(1) ,  (IBRCH(1)+14), 0, Rrotor, 0.0)

! 1370:[varrlc] Variable R, L or C
      CALL E_VARRLC1_EXE(0 ,SS(1) ,  (IBRCH(1)+13), 0, Rrotor, 0.0)

! 1380:[varrlc] Variable R, L or C
      CALL E_VARRLC1_EXE(0 ,SS(1) ,  (IBRCH(1)+15), 0, Rrotor, 0.0)

! 1:[xfmr-3p2w] 3 Phase 2 Winding Transformer
!  TRANSFORMER SATURATION SUBROUTINE
      CALL TSAT1_EXE( (IBRCH(1)+40), (IBRCH(1)+41), (IBRCH(1)+42),SS(1),&
     &1.0,1)

! 1:[xfmr-3p2w] 3 Phase 2 Winding Transformer
!  TRANSFORMER SATURATION SUBROUTINE
      CALL TSAT1_EXE( (IBRCH(1)+25), (IBRCH(1)+26), (IBRCH(1)+27),SS(1),&
     &1.0,1)

! 1:[xfmr-3p2w] 3 Phase 2 Winding Transformer
!  TRANSFORMER SATURATION SUBROUTINE
      CALL TSAT1_EXE( (IBRCH(1)+49), (IBRCH(1)+50), (IBRCH(1)+51),SS(1),&
     &1.0,1)

! 1:[source3] Three Phase Voltage Source Model 1 'Source'
!  3-Phase source: Source
      RVD1_1 = RTCF(NRTCF+12)
      RVD1_2 = RTCF(NRTCF+14)
      RVD1_3 = RTCF(NRTCF+13)
      CALL ESYS651_EXE(SS(1), (IBRCH(1)+28), (IBRCH(1)+29), (IBRCH(1)+30&
     &),0,0,0, SS(1), NT_11(1),NT_11(2),NT_11(3), 0, RVD1_2, RVD1_1, 0.0&
     &, 1.0, 1.0, 1.0,RVD1_3, 1.0, 0.02, 0.05, 1.0, 0.02, 0.05, RVD1_4, &
     &RVD1_5, RVD1_6, RVD1_7)

!---------------------------------------
! Feedbacks and transfers to storage
!---------------------------------------

      STOF(ISTOF + 3) = Tload
      STOF(ISTOF + 17) = Vrms
      STOF(ISTOF + 24) = RT_5
      STOF(ISTOF + 25) = RT_6
      STOI(ISTOI + 4) = IT_2
      STOI(ISTOI + 5) = IT_3
      STOI(ISTOI + 6) = IT_4
      STOI(ISTOI + 7) = IT_5
      STOI(ISTOI + 8) = IT_6
      STOF(ISTOF + 38) = Xa
```

```
STOF(ISTOF + 39) = Xb
STOF(ISTOF + 40) = Xc
STOF(ISTOF + 41) = Ra
STOF(ISTOF + 42) = Rb
STOF(ISTOF + 43) = Rc
STOF(ISTOF + 44) = Xab
STOF(ISTOF + 45) = Xbc
STOF(ISTOF + 46) = Xca
STOF(ISTOF + 47) = Rab
STOF(ISTOF + 48) = Rbc
STOF(ISTOF + 49) = Rca
STOF(ISTOF + 60) = vam
STOF(ISTOF + 61) = vbm
STOF(ISTOF + 62) = vcm
STOF(ISTOF + 63) = vap
STOF(ISTOF + 64) = vbp
STOF(ISTOF + 65) = vcp
STOF(ISTOF + 66) = vpm
STOF(ISTOF + 67) = vnm
STOF(ISTOF + 68) = vzm
STOF(ISTOF + 69) = vpp
STOF(ISTOF + 70) = vnp
STOF(ISTOF + 71) = vzp
STOF(ISTOF + 72) = iam
STOF(ISTOF + 73) = ibm
STOF(ISTOF + 74) = icm
STOF(ISTOF + 75) = iap
STOF(ISTOF + 76) = ibp
STOF(ISTOF + 77) = icp
STOF(ISTOF + 78) = ipm
STOF(ISTOF + 79) = inm
STOF(ISTOF + 80) = izm
STOF(ISTOF + 81) = ipp
STOF(ISTOF + 82) = inp
STOF(ISTOF + 83) = izp
STOF(ISTOF + 118) = RT_16
STOF(ISTOF + 119) = RT_17
STOF(ISTOF + 120) = RT_18
STOF(ISTOF + 163) = RT_25
STOF(ISTOF + 164) = RT_26
STOF(ISTOF + 165) = RT_27
STOF(ISTOF + 187) = RT_31
STOF(ISTOF + 188) = RT_32
STOF(ISTOF + 189) = RT_33
STOF(ISTOF + 190) = RT_34
STOF(ISTOF + 191) = RT_35
STOF(ISTOF + 192) = RT_36
STOF(ISTOF + 193) = RT_37
STOF(ISTOF + 194) = RT_38
STOF(ISTOF + 195) = RT_39
STOF(ISTOF + 196) = RT_40
STOF(ISTOF + 197) = RT_41
STOF(ISTOF + 198) = RT_42
STOI(ISTOI + 14) = IT_8
STOF(ISTOF + 199) = RT_43
STOF(ISTOF + 200) = RT_44
STOF(ISTOF + 201) = RT_45
STOF(ISTOF + 202) = RT_46
STOF(ISTOF + 203) = RT_47
STOF(ISTOF + 204) = RT_48
STOF(ISTOF + 205) = RT_49
STOF(ISTOF + 206) = RT_50
STOF(ISTOF + 207) = RT_51
STOF(ISTOF + 208) = RT_52
STOF(ISTOF + 209) = RT_53
STOF(ISTOF + 210) = RT_54
STOF(ISTOF + 211) = RT_55
STOF(ISTOF + 212) = RT_56
STOF(ISTOF + 213) = RT_57
STOF(ISTOF + 214) = RT_58
STOF(ISTOF + 215) = RT_59
```

```
      STOF(ISTOF + 216) = RT_60
      STOF(ISTOF + 217) = RT_61
      STOF(ISTOF + 218) = RT_62
      STOF(ISTOF + 219) = RT_63
      STOF(ISTOF + 220) = RT_64
      STOF(ISTOF + 221) = RT_65
      STOF(ISTOF + 222) = RT_66
      STOF(ISTOF + 231) = Ib
      STOF(ISTOF + 232) = Ic

! Array (1:3) quantities...
      DO IT_0 = 1,3
         STOF(ISTOF + 3 + IT_0) = Ix(IT_0)
         STOF(ISTOF + 10 + IT_0) = Vload(IT_0)
         STOF(ISTOF + 13 + IT_0) = Iload(IT_0)
         STOF(ISTOF + 17 + IT_0) = V_ad(IT_0)
         STOF(ISTOF + 20 + IT_0) = I_ad(IT_0)
         STOF(ISTOF + 25 + IT_0) = vp(IT_0)
         STOF(ISTOF + 28 + IT_0) = vm(IT_0)
         STOF(ISTOF + 31 + IT_0) = ip(IT_0)
         STOF(ISTOF + 34 + IT_0) = im(IT_0)
         STOF(ISTOF + 50 + IT_0) = iseqm(IT_0)
         STOF(ISTOF + 53 + IT_0) = iseqp(IT_0)
         STOF(ISTOF + 83 + IT_0) = vseqm(IT_0)
         STOF(ISTOF + 86 + IT_0) = vseqp(IT_0)
      END DO

! Array (1:7) quantities...
      DO IT_0 = 1,7
         STOF(ISTOF + 96 + IT_0) = RT_13(IT_0)
         STOF(ISTOF + 103 + IT_0) = RT_14(IT_0)
         STOF(ISTOF + 110 + IT_0) = RT_15(IT_0)
         STOF(ISTOF + 120 + IT_0) = RT_19(IT_0)
         STOF(ISTOF + 127 + IT_0) = RT_20(IT_0)
         STOF(ISTOF + 134 + IT_0) = RT_21(IT_0)
         STOF(ISTOF + 141 + IT_0) = RT_22(IT_0)
         STOF(ISTOF + 148 + IT_0) = RT_23(IT_0)
         STOF(ISTOF + 155 + IT_0) = RT_24(IT_0)
         STOF(ISTOF + 165 + IT_0) = RT_28(IT_0)
         STOF(ISTOF + 172 + IT_0) = RT_29(IT_0)
         STOF(ISTOF + 179 + IT_0) = RT_30(IT_0)
      END DO

!---------------------------------------
! Transfer to Exports
!---------------------------------------

!---------------------------------------
! Close Model Data read
!---------------------------------------

      IF ( TIMEZERO ) CALL EMTDC_CLOSEFILE
      RETURN
      END

!=======================================================================

      SUBROUTINE MainOut()

!---------------------------------------
! Standard includes
!---------------------------------------

      INCLUDE 'nd.h'
      INCLUDE 'emtconst.h'
      INCLUDE 'emtstor.h'
      INCLUDE 's0.h'
      INCLUDE 's1.h'
      INCLUDE 's2.h'
      INCLUDE 's4.h'
      INCLUDE 'branches.h'
```

```
      INCLUDE 'pscadv3.h'
      INCLUDE 'fnames.h'
      INCLUDE 'radiolinks.h'
      INCLUDE 'matlab.h'
      INCLUDE 'rtconfig.h'

!---------------------------------------
! Function/Subroutine Declarations
!---------------------------------------

      REAL    EMTDC_VVDC    !
      REAL    VM3PH2        ! '3 Phase RMS Voltage Measurement'
      REAL    VBRANCH       !
!     SUBR    FTN180        ! FFT Calculation
!     SUBR    SEQFILTER     ! Converts A,B,C to +,-,0

!---------------------------------------
! Variable Declarations
!---------------------------------------


! Electrical Node Indices
      INTEGER  NT_5(3), NT_7(3)

! Control Signals
      INTEGER  IT_2, IT_3, IT_4, IT_5, IT_6, IT_8
      REAL     Ix(3), Vload(3), Iload(3), Vrms
      REAL     V_ad(3), I_ad(3), RT_5, RT_6
      REAL     vp(3), vm(3), ip(3), im(3), Xa, Xb
      REAL     Xc, Ra, Rb, Rc, Xab, Xbc, Xca, Rab
      REAL     Rbc, Rca, iseqm(3), iseqp(3), vam
      REAL     vbm, vcm, vap, vbp, vcp, vpm, vnm
      REAL     vzm, vpp, vnp, vzp, iam, ibm, icm
      REAL     iap, ibp, icp, ipm, inm, izm, ipp
      REAL     inp, izp, vseqm(3), vseqp(3)
      REAL     RT_13(7), RT_14(7), RT_15(7)
      REAL     RT_16, RT_17, RT_18, RT_19(7)
      REAL     RT_20(7), RT_21(7), RT_22(7)
      REAL     RT_23(7), RT_24(7), RT_25, RT_26
      REAL     RT_27, RT_28(7), RT_29(7)
      REAL     RT_30(7), RT_31, RT_32, RT_33
      REAL     RT_34, RT_35, RT_36, RT_37, RT_38
      REAL     RT_39, RT_40, RT_41, RT_42, RT_43
      REAL     RT_44, RT_45, RT_46, RT_47, RT_48
      REAL     RT_49, RT_50, RT_51, RT_52, RT_53
      REAL     RT_54, RT_55, RT_56, RT_57, RT_58
      REAL     RT_59, RT_60, RT_61, RT_62, RT_63
      REAL     RT_64, RT_65, RT_66, Ib, Ic

! Internal Variables
      INTEGER  IVD1_1
      REAL     RVD1_1, RVD1_2

! Indexing variables
      INTEGER ICALL_NO                          ! Module call num
      INTEGER ISTOL, ISTOI, ISTOF, ISTOC, IT_0  ! Storage Indices
      INTEGER IPGB                              ! Control/Monitoring
      INTEGER ISUBS, SS(1), IBRCH(1), INODE     ! SS/Node/Branch/Xfmr
      INTEGER IXFMR


!---------------------------------------
! Local Indices
!---------------------------------------

! Dsdyn <-> Dsout transfer index storage

      NTXFR = NTXFR + 1

      ISTOL = TXFR(NTXFR,1)
      ISTOI = TXFR(NTXFR,2)
```

```
      ISTOF = TXFR(NTXFR,3)
      ISTOC = TXFR(NTXFR,4)

! Increment and assign runtime configuration call indices

      ICALL_NO  = NCALL_NO
      NCALL_NO  = NCALL_NO + 1

! Increment global storage indices

      IPGB      = NPGB
      NPGB      = NPGB + 34
      NCX       = NCX + 0
      INODE     = NNODE + 2
      NNODE     = NNODE + 30
      IXFMR     = NXFMR
      NXFMR     = NXFMR + 9

! Initialize Subsystem Mapping

      ISUBS = NSUBS + 0
      NSUBS = NSUBS + 1

      DO IT_0 = 1,1
         SS(IT_0) = SUBS(ISUBS + IT_0)
      END DO

! Initialize Branch Mapping.

      IBRCH(1)     = NBRCH(SS(1))
      NBRCH(SS(1)) = NBRCH(SS(1)) + 63
!---------------------------------------
! Transfers from storage arrays
!---------------------------------------

      Vrms     = STOF(ISTOF + 17)
      RT_5     = STOF(ISTOF + 24)
      RT_6     = STOF(ISTOF + 25)
      IT_2     = STOI(ISTOI + 4)
      IT_3     = STOI(ISTOI + 5)
      IT_4     = STOI(ISTOI + 6)
      IT_5     = STOI(ISTOI + 7)
      IT_6     = STOI(ISTOI + 8)
      Xa       = STOF(ISTOF + 38)
      Xb       = STOF(ISTOF + 39)
      Xc       = STOF(ISTOF + 40)
      Ra       = STOF(ISTOF + 41)
      Rb       = STOF(ISTOF + 42)
      Rc       = STOF(ISTOF + 43)
      Xab      = STOF(ISTOF + 44)
      Xbc      = STOF(ISTOF + 45)
      Xca      = STOF(ISTOF + 46)
      Rab      = STOF(ISTOF + 47)
      Rbc      = STOF(ISTOF + 48)
      Rca      = STOF(ISTOF + 49)
      vam      = STOF(ISTOF + 60)
      vbm      = STOF(ISTOF + 61)
      vcm      = STOF(ISTOF + 62)
      vap      = STOF(ISTOF + 63)
      vbp      = STOF(ISTOF + 64)
      vcp      = STOF(ISTOF + 65)
      vpm      = STOF(ISTOF + 66)
      vnm      = STOF(ISTOF + 67)
      vzm      = STOF(ISTOF + 68)
      vpp      = STOF(ISTOF + 69)
      vnp      = STOF(ISTOF + 70)
      vzp      = STOF(ISTOF + 71)
      iam      = STOF(ISTOF + 72)
      ibm      = STOF(ISTOF + 73)
      icm      = STOF(ISTOF + 74)
      iap      = STOF(ISTOF + 75)
```

```
      ibp      = STOF(ISTOF + 76)
      icp      = STOF(ISTOF + 77)
      ipm      = STOF(ISTOF + 78)
      inm      = STOF(ISTOF + 79)
      izm      = STOF(ISTOF + 80)
      ipp      = STOF(ISTOF + 81)
      inp      = STOF(ISTOF + 82)
      izp      = STOF(ISTOF + 83)
      RT_16    = STOF(ISTOF + 118)
      RT_17    = STOF(ISTOF + 119)
      RT_18    = STOF(ISTOF + 120)
      RT_25    = STOF(ISTOF + 163)
      RT_26    = STOF(ISTOF + 164)
      RT_27    = STOF(ISTOF + 165)
      RT_31    = STOF(ISTOF + 187)
      RT_32    = STOF(ISTOF + 188)
      RT_33    = STOF(ISTOF + 189)
      RT_34    = STOF(ISTOF + 190)
      RT_35    = STOF(ISTOF + 191)
      RT_36    = STOF(ISTOF + 192)
      RT_37    = STOF(ISTOF + 193)
      RT_38    = STOF(ISTOF + 194)
      RT_39    = STOF(ISTOF + 195)
      RT_40    = STOF(ISTOF + 196)
      RT_41    = STOF(ISTOF + 197)
      RT_42    = STOF(ISTOF + 198)
      IT_8     = STOI(ISTOI + 14)
      RT_43    = STOF(ISTOF + 199)
      RT_44    = STOF(ISTOF + 200)
      RT_45    = STOF(ISTOF + 201)
      RT_46    = STOF(ISTOF + 202)
      RT_47    = STOF(ISTOF + 203)
      RT_48    = STOF(ISTOF + 204)
      RT_49    = STOF(ISTOF + 205)
      RT_50    = STOF(ISTOF + 206)
      RT_51    = STOF(ISTOF + 207)
      RT_52    = STOF(ISTOF + 208)
      RT_53    = STOF(ISTOF + 209)
      RT_54    = STOF(ISTOF + 210)
      RT_55    = STOF(ISTOF + 211)
      RT_56    = STOF(ISTOF + 212)
      RT_57    = STOF(ISTOF + 213)
      RT_58    = STOF(ISTOF + 214)
      RT_59    = STOF(ISTOF + 215)
      RT_60    = STOF(ISTOF + 216)
      RT_61    = STOF(ISTOF + 217)
      RT_62    = STOF(ISTOF + 218)
      RT_63    = STOF(ISTOF + 219)
      RT_64    = STOF(ISTOF + 220)
      RT_65    = STOF(ISTOF + 221)
      RT_66    = STOF(ISTOF + 222)
      Ib       = STOF(ISTOF + 231)
      Ic       = STOF(ISTOF + 232)

! Array (1:3) quantities...
      DO IT_0 = 1,3
         Ix(IT_0) = STOF(ISTOF + 3 + IT_0)
         Vload(IT_0) = STOF(ISTOF + 10 + IT_0)
         Iload(IT_0) = STOF(ISTOF + 13 + IT_0)
         V_ad(IT_0) = STOF(ISTOF + 17 + IT_0)
         I_ad(IT_0) = STOF(ISTOF + 20 + IT_0)
         vp(IT_0) = STOF(ISTOF + 25 + IT_0)
         vm(IT_0) = STOF(ISTOF + 28 + IT_0)
         ip(IT_0) = STOF(ISTOF + 31 + IT_0)
         im(IT_0) = STOF(ISTOF + 34 + IT_0)
         iseqm(IT_0) = STOF(ISTOF + 50 + IT_0)
         iseqp(IT_0) = STOF(ISTOF + 53 + IT_0)
         vseqm(IT_0) = STOF(ISTOF + 83 + IT_0)
         vseqp(IT_0) = STOF(ISTOF + 86 + IT_0)
      END DO
```

```
! Array (1:7) quantities...
      DO IT_0 = 1,7
         RT_13(IT_0) = STOF(ISTOF + 96 + IT_0)
         RT_14(IT_0) = STOF(ISTOF + 103 + IT_0)
         RT_15(IT_0) = STOF(ISTOF + 110 + IT_0)
         RT_19(IT_0) = STOF(ISTOF + 120 + IT_0)
         RT_20(IT_0) = STOF(ISTOF + 127 + IT_0)
         RT_21(IT_0) = STOF(ISTOF + 134 + IT_0)
         RT_22(IT_0) = STOF(ISTOF + 141 + IT_0)
         RT_23(IT_0) = STOF(ISTOF + 148 + IT_0)
         RT_24(IT_0) = STOF(ISTOF + 155 + IT_0)
         RT_28(IT_0) = STOF(ISTOF + 165 + IT_0)
         RT_29(IT_0) = STOF(ISTOF + 172 + IT_0)
         RT_30(IT_0) = STOF(ISTOF + 179 + IT_0)
      END DO

!---------------------------------------
! Electrical Node Lookup
!---------------------------------------


! Array (1:3) quantities...
      DO IT_0 = 1,3
         NT_5(IT_0) = NODE(INODE + 1 + IT_0)
         NT_7(IT_0) = NODE(INODE + 7 + IT_0)
      END DO

!---------------------------------------
! Configuration of Models
!---------------------------------------

      IF ( TIMEZERO ) THEN
         FILENAME = 'Main.dta'
         CALL EMTDC_OPENFILE
         SECTION = 'DATADSO:'
         CALL EMTDC_GOTOSECTION
      ENDIF
!---------------------------------------
! Generated code from module definition
!---------------------------------------


! 50:[multimeter] Multimeter
      IVD1_1 = NRTCF
      NRTCF  = NRTCF + 4
      I_ad(1) = ( CBR((IBRCH(1)+52), SS(1)))
      I_ad(2) = ( CBR((IBRCH(1)+53), SS(1)))
      I_ad(3) = ( CBR((IBRCH(1)+54), SS(1)))
      V_ad(1) = EMTDC_VVDC(SS(1), NT_7(1), 0)
      V_ad(2) = EMTDC_VVDC(SS(1), NT_7(2), 0)
      V_ad(3) = EMTDC_VVDC(SS(1), NT_7(3), 0)
      RVD1_1 = RTCF(IVD1_1+1) * VM3PH2(SS(1), NT_7(1), NT_7(2), NT_7(3),&
     & RTCF(IVD1_1+2))
      Vrms = RVD1_1

! 60:[breaker3] 3 Phase Breaker 'BRK'
! Three Phase Breaker Currents
      Ib = ( CBR((IBRCH(1)+32), SS(1)))
      Ic = ( CBR((IBRCH(1)+33), SS(1)))
      CALL BRK_POWER(SS(1), (IBRCH(1)+31), (IBRCH(1)+32), (IBRCH(1)+33),&
     &0,0,0,IVD1_1,0.02,RVD1_1,RVD1_2)

! 80:[ammeter] Current Meter 'Ix'
      Ix(1) = ( CBR((IBRCH(1)+16), SS(1)))
      Ix(2) = ( CBR((IBRCH(1)+17), SS(1)))
      Ix(3) = ( CBR((IBRCH(1)+18), SS(1)))

! 90:[multimeter] Multimeter
      IVD1_1 = NRTCF
      NRTCF  = NRTCF + 4
      Iload(1) = ( CBR((IBRCH(1)+61), SS(1)))
```

```
      Iload(2) = ( CBR((IBRCH(1)+62), SS(1)))
      Iload(3) = ( CBR((IBRCH(1)+63), SS(1)))
      Vload(1) = EMTDC_VVDC(SS(1), NT_5(1), 0)
      Vload(2) = EMTDC_VVDC(SS(1), NT_5(2), 0)
      Vload(3) = EMTDC_VVDC(SS(1), NT_5(3), 0)

! 180:[pgb] Output Channel 'TERMINAL VOLTAGE'

      PGB(IPGB+1) = Vrms

! 190:[pgb] Output Channel 'Current'

      DO IVD1_1 = 1, 3
         PGB(IPGB+2+IVD1_1-1) = Ix(IVD1_1)
      ENDDO

! 200:[pgb] Output Channel 'Vload'

      DO IVD1_1 = 1, 3
         PGB(IPGB+5+IVD1_1-1) = Vload(IVD1_1)
      ENDDO

! 210:[pgb] Output Channel 'Iload'

      DO IVD1_1 = 1, 3
         PGB(IPGB+8+IVD1_1-1) = Iload(IVD1_1)
      ENDDO

! 220:[pgb] Output Channel 'Vs'

      DO IVD1_1 = 1, 3
         PGB(IPGB+11+IVD1_1-1) = V_ad(IVD1_1)
      ENDDO

! 230:[fft] On-Line Frequency Scanner
      IVD1_1=0
      CALL FTN180(0,0,7,1,50.0,50.0,V_ad(1),IVD1_1,RT_20,RT_14,RT_16)
      CALL FTN180(0,0,7,1,50.0,50.0,V_ad(2),IVD1_1,RT_19,RT_13,RT_18)
      CALL FTN180(0,0,7,1,50.0,50.0,V_ad(3),IVD1_1,RT_21,RT_15,RT_17)
!

! 240:[datatap] Scalar/Array Tap
       vbp = RT_13(1)

! 270:[datatap] Scalar/Array Tap
       vcp = RT_15(1)

! 330:[pgb] Output Channel 'Is'

      DO IVD1_1 = 1, 3
         PGB(IPGB+14+IVD1_1-1) = I_ad(IVD1_1)
      ENDDO

! 340:[fft] On-Line Frequency Scanner
      IVD1_1=0
      CALL FTN180(0,0,7,1,50.0,50.0,I_ad(1),IVD1_1,RT_29,RT_23,RT_25)
      CALL FTN180(0,0,7,1,50.0,50.0,I_ad(2),IVD1_1,RT_28,RT_22,RT_27)
      CALL FTN180(0,0,7,1,50.0,50.0,I_ad(3),IVD1_1,RT_30,RT_24,RT_26)
!

! 350:[datatap] Scalar/Array Tap
       ibp = RT_22(1)

! 360:[datatap] Scalar/Array Tap
       icp = RT_24(1)

! 370:[datatap] Scalar/Array Tap
       vam = RT_20(1)

! 380:[datatap] Scalar/Array Tap
       vbm = RT_19(1)
```

```
! 390:[datatap] Scalar/Array Tap
       vcm = RT_21(1)

! 400:[datatap] Scalar/Array Tap
       vap = RT_14(1)

! 410:[datamerge] Merges data signals into an array

      vm(1) = vam
      vm(2) = vbm
      vm(3) = vcm

! 420:[datatap] Scalar/Array Tap
       RT_59 = vm(3)

! 430:[seq_filter] Sequence Filter
! Sequence filter
      CALL SEQFILTER(vam,vbm,vcm,vap,vbp,vcp,vpm,vnm,vzm,vpp,vnp,vzp,0,0&
     &)
!

! 440:[datamerge] Merges data signals into an array

      vp(1) = vap
      vp(2) = vbp
      vp(3) = vcp

! 470:[datatap] Scalar/Array Tap
       RT_61 = vm(1)

! 480:[datatap] Scalar/Array Tap
       RT_62 = vp(1)

! 490:[datatap] Scalar/Array Tap
       iam = RT_29(1)

! 500:[datatap] Scalar/Array Tap
       ibm = RT_28(1)

! 510:[datatap] Scalar/Array Tap
       icm = RT_30(1)

! 540:[datatap] Scalar/Array Tap
       iap = RT_23(1)

! 550:[seq_filter] Sequence Filter
! Sequence filter
      CALL SEQFILTER(iam,ibm,icm,iap,ibp,icp,ipm,inm,izm,ipp,inp,izp,0,0&
     &)
!

! 560:[datamerge] Merges data signals into an array

      im(1) = iam
      im(2) = ibm
      im(3) = icm

! 570:[datamerge] Merges data signals into an array

      iseqm(1) = ipm
      iseqm(2) = inm
      iseqm(3) = izm

! 580:[datamerge] Merges data signals into an array

      ip(1) = iap
      ip(2) = ibp
      ip(3) = icp

! 590:[datamerge] Merges data signals into an array
```

```
      iseqp(1) = ipp
      iseqp(2) = inp
      iseqp(3) = izp

! 600:[datatap] Scalar/Array Tap
       RT_5 = iseqm(1)

! 610:[datatap] Scalar/Array Tap
       RT_6 = iseqp(1)

! 620:[datatap] Scalar/Array Tap
       RT_32 = vm(1)

! 630:[datatap] Scalar/Array Tap
       RT_31 = vp(1)

! 640:[datatap] Scalar/Array Tap
       RT_33 = im(1)

! 650:[datatap] Scalar/Array Tap
       RT_34 = ip(1)

! 660:[datatap] Scalar/Array Tap
       RT_44 = vm(2)

! 670:[datatap] Scalar/Array Tap
       RT_43 = vp(2)

! 680:[datatap] Scalar/Array Tap
       RT_45 = im(2)

! 690:[datatap] Scalar/Array Tap
       RT_46 = ip(2)

! 700:[datatap] Scalar/Array Tap
       RT_48 = vm(3)

! 710:[datatap] Scalar/Array Tap
       RT_47 = vp(3)

! 720:[datatap] Scalar/Array Tap
       RT_49 = im(3)

! 730:[datatap] Scalar/Array Tap
       RT_50 = ip(3)

! 740:[datatap] Scalar/Array Tap
       RT_35 = vm(1)

! 750:[datatap] Scalar/Array Tap
       RT_36 = vp(1)

! 760:[datatap] Scalar/Array Tap
       RT_39 = im(1)

! 770:[datatap] Scalar/Array Tap
       RT_40 = ip(1)

! 780:[datatap] Scalar/Array Tap
       RT_37 = vm(2)

! 790:[datatap] Scalar/Array Tap
       RT_38 = vp(2)

! 800:[datatap] Scalar/Array Tap
       RT_41 = im(2)

! 810:[datatap] Scalar/Array Tap
       RT_42 = ip(2)

! 820:[datatap] Scalar/Array Tap
```

```
       RT_51 = vm(2)

! 830:[datatap] Scalar/Array Tap
       RT_52 = vp(2)

! 840:[datatap] Scalar/Array Tap
       RT_55 = im(2)

! 850:[datatap] Scalar/Array Tap
       RT_56 = ip(2)

! 860:[datatap] Scalar/Array Tap
       RT_60 = vp(3)

! 870:[datatap] Scalar/Array Tap
       RT_63 = im(3)

! 880:[datatap] Scalar/Array Tap
       RT_64 = ip(3)

! 940:[l2l_imp] Line to Line impedance
      CALL E_L2LIMP1_EXE(RT_35,RT_36,RT_39,RT_40,RT_37,RT_38,RT_41,RT_42&
     &,Rab,Xab)

! 950:[l2g_imp] Line to Ground Impedance
      CALL E_L2GIMP1_EXE(RT_48,RT_47,RT_49,RT_50,RT_5,RT_6,Rc,Xc)
!

! 960:[l2g_imp] Line to Ground Impedance
      CALL E_L2GIMP1_EXE(RT_44,RT_43,RT_45,RT_46,RT_5,RT_6,Rb,Xb)
!

! 970:[l2g_imp] Line to Ground Impedance
      CALL E_L2GIMP1_EXE(RT_32,RT_31,RT_33,RT_34,RT_5,RT_6,Ra,Xa)
!

! 980:[datatap] Scalar/Array Tap
       RT_66 = ip(1)

! 990:[datatap] Scalar/Array Tap
       RT_65 = im(1)

! 1000:[l2l_imp] Line to Line impedance
      CALL E_L2LIMP1_EXE(RT_59,RT_60,RT_63,RT_64,RT_61,RT_62,RT_65,RT_66&
     &,Rca,Xca)

! 1010:[datatap] Scalar/Array Tap
       RT_58 = ip(3)

! 1020:[datatap] Scalar/Array Tap
       RT_57 = im(3)

! 1030:[datatap] Scalar/Array Tap
       RT_54 = vp(3)

! 1040:[datatap] Scalar/Array Tap
       RT_53 = vm(3)

! 1050:[l2l_imp] Line to Line impedance
      CALL E_L2LIMP1_EXE(RT_51,RT_52,RT_55,RT_56,RT_53,RT_54,RT_57,RT_58&
     &,Rbc,Xbc)

! 1060:[mho_circle] Mho Circle
      CALL TRPCCL1_EXE(Rab,Xab,IT_3)

! 1070:[mho_circle] Mho Circle
      CALL TRPCCL1_EXE(Rc,Xc,IT_6)

! 1080:[mho_circle] Mho Circle
      CALL TRPCCL1_EXE(Rb,Xb,IT_5)
```

```
! 1090:[mho_circle] Mho Circle
      CALL TRPCCL1_EXE(Ra,Xa,IT_4)
! 1100:[pgb] Output Channel 'Xca'
      PGB(IPGB+17) = Xca
! 1110:[pgb] Output Channel 'Xc'
      PGB(IPGB+18) = Xc
! 1120:[pgb] Output Channel 'Rca'
      PGB(IPGB+19) = Rca
! 1130:[pgb] Output Channel 'Rc'
      PGB(IPGB+20) = Rc
! 1140:[pgb] Output Channel 'Xbc'
      PGB(IPGB+21) = Xbc
! 1160:[pgb] Output Channel 'Xb'
      PGB(IPGB+23) = Xb
! 1170:[pgb] Output Channel 'Rbc'
      PGB(IPGB+24) = Rbc
! 1180:[pgb] Output Channel 'Rb'
      PGB(IPGB+25) = Rb
! 1190:[mho_circle] Mho Circle
      CALL TRPCCL1_EXE(Rca,Xca,IT_2)
! 1200:[pgb] Output Channel 'Xab'
      PGB(IPGB+26) = Xab
! 1220:[datamerge] Merges data signals into an array
      vseqp(1) = vpp
      vseqp(2) = vnp
      vseqp(3) = vzp
! 1230:[pgb] Output Channel 'Xa'
      PGB(IPGB+28) = Xa
! 1240:[pgb] Output Channel 'Rab'
      PGB(IPGB+29) = Rab
! 1250:[pgb] Output Channel 'Ra'
      PGB(IPGB+30) = Ra
! 1260:[datamerge] Merges data signals into an array
      vseqm(1) = vpm
      vseqm(2) = vnm
      vseqm(3) = vzm
! 1270:[mho_circle] Mho Circle
      CALL TRPCCL1_EXE(Rbc,Xbc,IT_8)
!---------------------------------------
! Feedbacks and transfers to storage
!---------------------------------------
      STOF(ISTOF + 17) = Vrms
      STOF(ISTOF + 24) = RT_5
      STOF(ISTOF + 25) = RT_6
      STOI(ISTOI + 4) = IT_2
      STOI(ISTOI + 5) = IT_3
      STOI(ISTOI + 6) = IT_4
      STOI(ISTOI + 7) = IT_5
      STOI(ISTOI + 8) = IT_6
      STOF(ISTOF + 38) = Xa
      STOF(ISTOF + 39) = Xb
      STOF(ISTOF + 40) = Xc
      STOF(ISTOF + 41) = Ra
      STOF(ISTOF + 42) = Rb
      STOF(ISTOF + 43) = Rc
      STOF(ISTOF + 44) = Xab
      STOF(ISTOF + 45) = Xbc
      STOF(ISTOF + 46) = Xca
      STOF(ISTOF + 47) = Rab
      STOF(ISTOF + 48) = Rbc
      STOF(ISTOF + 49) = Rca
      STOF(ISTOF + 60) = vam
      STOF(ISTOF + 61) = vbm
      STOF(ISTOF + 62) = vcm
      STOF(ISTOF + 63) = vap
      STOF(ISTOF + 64) = vbp
      STOF(ISTOF + 65) = vcp
      STOF(ISTOF + 66) = vpm
      STOF(ISTOF + 67) = vnm
      STOF(ISTOF + 68) = vzm
      STOF(ISTOF + 69) = vpp
```

```
      STOF(ISTOF + 70) = vnp
      STOF(ISTOF + 71) = vzp
      STOF(ISTOF + 72) = iam
      STOF(ISTOF + 73) = ibm
      STOF(ISTOF + 74) = icm
      STOF(ISTOF + 75) = iap
      STOF(ISTOF + 76) = ibp
      STOF(ISTOF + 77) = icp
      STOF(ISTOF + 78) = ipm
      STOF(ISTOF + 79) = inm
      STOF(ISTOF + 80) = izm
      STOF(ISTOF + 81) = ipp
      STOF(ISTOF + 82) = inp
      STOF(ISTOF + 83) = izp
      STOF(ISTOF + 118) = RT_16
      STOF(ISTOF + 119) = RT_17
      STOF(ISTOF + 120) = RT_18
      STOF(ISTOF + 163) = RT_25
      STOF(ISTOF + 164) = RT_26
      STOF(ISTOF + 165) = RT_27
      STOF(ISTOF + 187) = RT_31
      STOF(ISTOF + 188) = RT_32
      STOF(ISTOF + 189) = RT_33
      STOF(ISTOF + 190) = RT_34
      STOF(ISTOF + 191) = RT_35
      STOF(ISTOF + 192) = RT_36
      STOF(ISTOF + 193) = RT_37
      STOF(ISTOF + 194) = RT_38
      STOF(ISTOF + 195) = RT_39
      STOF(ISTOF + 196) = RT_40
      STOF(ISTOF + 197) = RT_41
      STOF(ISTOF + 198) = RT_42
      STOI(ISTOI + 14) = IT_8
      STOF(ISTOF + 199) = RT_43
      STOF(ISTOF + 200) = RT_44
      STOF(ISTOF + 201) = RT_45
      STOF(ISTOF + 202) = RT_46
      STOF(ISTOF + 203) = RT_47
      STOF(ISTOF + 204) = RT_48
      STOF(ISTOF + 205) = RT_49
      STOF(ISTOF + 206) = RT_50
      STOF(ISTOF + 207) = RT_51
      STOF(ISTOF + 208) = RT_52
      STOF(ISTOF + 209) = RT_53
      STOF(ISTOF + 210) = RT_54
      STOF(ISTOF + 211) = RT_55
      STOF(ISTOF + 212) = RT_56
      STOF(ISTOF + 213) = RT_57
      STOF(ISTOF + 214) = RT_58
      STOF(ISTOF + 215) = RT_59
      STOF(ISTOF + 216) = RT_60
      STOF(ISTOF + 217) = RT_61
      STOF(ISTOF + 218) = RT_62
      STOF(ISTOF + 219) = RT_63
      STOF(ISTOF + 220) = RT_64
      STOF(ISTOF + 221) = RT_65
      STOF(ISTOF + 222) = RT_66
      STOF(ISTOF + 231) = Ib
      STOF(ISTOF + 232) = Ic

! Array (1:3) quantities...
      DO IT_0 = 1,3
         STOF(ISTOF + 3 + IT_0) = Ix(IT_0)
         STOF(ISTOF + 10 + IT_0) = Vload(IT_0)
         STOF(ISTOF + 13 + IT_0) = Iload(IT_0)
         STOF(ISTOF + 17 + IT_0) = V_ad(IT_0)
         STOF(ISTOF + 20 + IT_0) = I_ad(IT_0)
         STOF(ISTOF + 25 + IT_0) = vp(IT_0)
         STOF(ISTOF + 28 + IT_0) = vm(IT_0)
         STOF(ISTOF + 31 + IT_0) = ip(IT_0)
         STOF(ISTOF + 34 + IT_0) = im(IT_0)
```

```
         STOF(ISTOF + 50 + IT_0) = iseqm(IT_0)
         STOF(ISTOF + 53 + IT_0) = iseqp(IT_0)
         STOF(ISTOF + 83 + IT_0) = vseqm(IT_0)
         STOF(ISTOF + 86 + IT_0) = vseqp(IT_0)
      END DO

! Array (1:7) quantities...
      DO IT_0 = 1,7
         STOF(ISTOF + 96 + IT_0) = RT_13(IT_0)
         STOF(ISTOF + 103 + IT_0) = RT_14(IT_0)
         STOF(ISTOF + 110 + IT_0) = RT_15(IT_0)
         STOF(ISTOF + 120 + IT_0) = RT_19(IT_0)
         STOF(ISTOF + 127 + IT_0) = RT_20(IT_0)
         STOF(ISTOF + 134 + IT_0) = RT_21(IT_0)
         STOF(ISTOF + 141 + IT_0) = RT_22(IT_0)
         STOF(ISTOF + 148 + IT_0) = RT_23(IT_0)
         STOF(ISTOF + 155 + IT_0) = RT_24(IT_0)
         STOF(ISTOF + 165 + IT_0) = RT_28(IT_0)
         STOF(ISTOF + 172 + IT_0) = RT_29(IT_0)
         STOF(ISTOF + 179 + IT_0) = RT_30(IT_0)
      END DO

!---------------------------------------
! Close Model Data read
!---------------------------------------

      IF ( TIMEZERO ) CALL EMTDC_CLOSEFILE
      RETURN
      END

!=======================================================================

      SUBROUTINE MainDyn_Begin()

!---------------------------------------
! Standard includes
!---------------------------------------

      INCLUDE 'nd.h'
      INCLUDE 'emtconst.h'
      INCLUDE 's0.h'
      INCLUDE 's1.h'
      INCLUDE 's4.h'
      INCLUDE 'branches.h'
      INCLUDE 'pscadv3.h'
      INCLUDE 'rtconfig.h'

!---------------------------------------
! Function/Subroutine Declarations
!---------------------------------------


!---------------------------------------
! Variable Declarations
!---------------------------------------


! Subroutine Arguments

! Electrical Node Indices
      INTEGER  NT_11(3)

! Control Signals
      INTEGER  IT_1
      REAL     WIN, RT_11, RT_67, RT_68, RT_70
      REAL     RT_71, RT_72, RT_74

! Internal Variables
      INTEGER  IVD1_1
      REAL     RVD1_1, RVD1_2, RVD5_1(5)
      REAL     RVD12_1(12), RVD10_1(10)
```

```
      REAL     RVD10_2(10), RVD10_3(10)
      REAL     RVD10_4(10), RVD1_3, RVD1_4
      REAL     RVD1_5, RVD1_6

! Indexing variables
      INTEGER ICALL_NO                          ! Module call num
      INTEGER IT_0                              ! Storage Indices
      INTEGER ISUBS, SS(1), IBRCH(1), INODE     ! SS/Node/Branch/Xfmr
      INTEGER IXFMR


!---------------------------------------
! Local Indices
!---------------------------------------


! Increment and assign runtime configuration call indices

      ICALL_NO  = NCALL_NO
      NCALL_NO  = NCALL_NO + 1

! Increment global storage indices

      NCX       = NCX + 0
      INODE     = NNODE + 2
      NNODE     = NNODE + 30
      IXFMR     = NXFMR
      NXFMR     = NXFMR + 9

! Initialize Subsystem Mapping

      ISUBS = NSUBS + 0
      NSUBS = NSUBS + 1

      DO IT_0 = 1,1
         SS(IT_0) = SUBS(ISUBS + IT_0)
      END DO

! Initialize Branch Mapping.

      IBRCH(1)     = NBRCH(SS(1))
      NBRCH(SS(1)) = NBRCH(SS(1)) + 63
!---------------------------------------
! Electrical Node Lookup
!---------------------------------------


! Array (1:3) quantities...
      DO IT_0 = 1,3
         NT_11(IT_0) = NODE(INODE + 19 + IT_0)
      END DO

!---------------------------------------
! Generated code from module definition
!---------------------------------------


! 10:[var] Variable Input Slider 'Rrotor'

! 30:[const] Real Constant

! 40:[consti] Integer Constant

! 60:[breaker3] 3 Phase Breaker 'BRK'
      CALL COMPONENT_ID(ICALL_NO,423392219)
      RTCF(NRTCF) = ABS(0.0)
      NRTCF = NRTCF + 1

! 70:[const] Real Constant

! 100:[time-sig] Output of Simulation Time
```

```
! 110:[compare] Single Input Level Comparator
! 120:[unity] Type conversion block
! 130:[var] Variable Input Slider 'Gradient -k'
! 140:[var] Variable Input Slider 'Stdstill torq - b'
! 150:[time-sig] Output of Simulation Time
! 160:[compare] Single Input Level Comparator
! 170:[unity] Type conversion block
! 250:[const] Real Constant
! 260:[const] Real Constant
! 280:[const] Real Constant
! 290:[signalgen] Signal Generator /w Interpolation
      CALL COMPONENT_ID(ICALL_NO,1628879709)
      CALL E_XSGEN1_CFG(1,0.0,100.0,360.0,0.0)
! 300:[const] Real Constant
! 310:[const] Real Constant
! 320:[const] Real Constant
! 450:[modulator] Amplitude/Frequency/Phase Modulator
! 460:[sumjct] Summing/Differencing Junctions
! 520:[modulator] Amplitude/Frequency/Phase Modulator
! 530:[sumjct] Summing/Differencing Junctions
! 890:[select] Two Input Selector
      RTCI(NRTCI) = 1
      NRTCI = NRTCI + 1
! 900:[wound_rotor] Wound Rotor Induction Machine 'WR'
      RVD5_1(1)   = 0.0034
      RVD5_1(2)   = 0.00607
      RVD5_1(3:5) = 0.0
      RVD12_1(1)  = 0.9
      RVD12_1(2)  = 0.0102
      RVD12_1(3)  = 0.011
      RVD12_1(4:12) = 0.0
      RVD10_1(1)  = 0.0
      RVD10_2(1)  = 0.0
      RVD10_1(2)  = 1.0
      RVD10_2(2)  = 1.0
      RVD10_1(3)  = -1.0
      RVD10_2(3)  = -1.0
      RVD10_1(4)  = 2.861
      RVD10_2(4)  = 0.564
      RVD10_1(5)  = 5.882
      RVD10_2(5)  = 0.981
      RVD10_1(6)  = -1.0
      RVD10_2(6)  = -1.0
      RVD10_1(7)  = -1.0
      RVD10_2(7)  = -1.0
      RVD10_1(8)  = -1.0
      RVD10_2(8)  = -1.0
      RVD10_1(9)  = -1.0
      RVD10_2(9)  = -1.0
      RVD10_1(10) = -1.0
      RVD10_2(10) = -1.0
      RVD10_3(1)  = 0.0
      RVD10_4(1)  = 0.0
      RVD10_3(2)  = 1.0
      RVD10_4(2)  = 1.0
      RVD10_3(3)  = -1.0
      RVD10_4(3)  = -1.0
      RVD10_3(4)  = 1.0
      RVD10_4(4)  = 0.947
      RVD10_3(5)  = 1.6
      RVD10_4(5)  = 1.076
      RVD10_3(6)  = 1.7
      RVD10_4(6)  = 1.2
      RVD10_3(7)  = 2.1
      RVD10_4(7)  = 1.26
      RVD10_3(8)  = 2.4
      RVD10_4(8)  = 1.32
      RVD10_3(9)  = 3.2
      RVD10_4(9)  = 1.42
      RVD10_3(10) = 4.2
      RVD10_4(10) = 1.5
```

```
      CALL COMPONENT_ID(ICALL_NO,708257837)
      CALL INDUCMC1_CFG(0,1.2,13.8,376.991118431,3.7267,0.01,2.637687,20&
     &.0,10.0,RVD5_1,RVD12_1,RVD10_1,RVD10_2,RVD10_3,RVD10_4)
! 910:[square] Square
! 920:[mult] Multiplier
! 930:[sumjct] Summing/Differencing Junctions
! 1150:[pgb] Output Channel 'Xcircle'
! 1210:[pgb] Output Channel 'Rcircle'
! 1290:[pgb] Output Channel 'Tload'
! 1310:[pgb] Output Channel 'Speed'
! 1320:[pgb] Output Channel 'Electric torque'
! 1350:[pgb] Output Channel 'Torque Input'
! 1360:[varrlc] Variable R, L or C
      CALL E_VARRLC1_CFG(0 ,SS(1) ,  (IBRCH(1)+14), 0)
! 1370:[varrlc] Variable R, L or C
      CALL E_VARRLC1_CFG(0 ,SS(1) ,  (IBRCH(1)+13), 0)
! 1380:[varrlc] Variable R, L or C
      CALL E_VARRLC1_CFG(0 ,SS(1) ,  (IBRCH(1)+15), 0)
! 1:[xfmr-3p2w] 3 Phase 2 Winding Transformer
      CALL COMPONENT_ID(ICALL_NO,1680367034)
      RVD1_1 = ONE_3RD*1.0
      RVD1_2 = 12.0*SQRT_1BY3
      RVD1_3 = 13.8*SQRT_1BY3
      CALL E_TF2W_CFG((IXFMR + 1),0,RVD1_1,60.0,0.02,0.0,RVD1_2,RVD1_3,1&
     &.0)
      CALL E_TF2W_CFG((IXFMR + 2),0,RVD1_1,60.0,0.02,0.0,RVD1_2,RVD1_3,1&
     &.0)
      CALL E_TF2W_CFG((IXFMR + 3),0,RVD1_1,60.0,0.02,0.0,RVD1_2,RVD1_3,1&
     &.0)
      IF (0.0 .LT. 0.001) THEN
        RVD1_5 = 0.0
        RVD1_6 = 0.0
        IVD1_1 = 0
      ELSE
        RVD1_6 = 0.0
        RVD1_4 = 6.0/(1.0*RVD1_6)
        RVD1_5 = RVD1_4*RVD1_2*RVD1_2
        RVD1_6 = RVD1_4*RVD1_3*RVD1_3
        IVD1_1 = 1
      ENDIF
      CALL E_BRANCH_CFG( (IBRCH(1)+34),SS(1),IVD1_1,0,0,RVD1_5,0.0,0.0)
      CALL E_BRANCH_CFG( (IBRCH(1)+35),SS(1),IVD1_1,0,0,RVD1_5,0.0,0.0)
      CALL E_BRANCH_CFG( (IBRCH(1)+36),SS(1),IVD1_1,0,0,RVD1_5,0.0,0.0)
      CALL E_BRANCH_CFG( (IBRCH(1)+37),SS(1),IVD1_1,0,0,RVD1_6,0.0,0.0)
      CALL E_BRANCH_CFG( (IBRCH(1)+38),SS(1),IVD1_1,0,0,RVD1_6,0.0,0.0)
      CALL E_BRANCH_CFG( (IBRCH(1)+39),SS(1),IVD1_1,0,0,RVD1_6,0.0,0.0)
      CALL TSAT1_CFG( (IBRCH(1)+40), (IBRCH(1)+41), (IBRCH(1)+42),SS(1),&
     &RVD1_1,RVD1_2,0.2,1.25,60.0,1.0,1.0,0.1)
! 1:[xfmr-3p2w] 3 Phase 2 Winding Transformer
      CALL COMPONENT_ID(ICALL_NO,1644239544)
      RVD1_1 = ONE_3RD*100.0
      RVD1_2 = 13.8
      RVD1_3 = 1.732*SQRT_1BY3
      CALL E_TF2W_CFG((IXFMR + 4),0,RVD1_1,50.0,0.1,0.0,RVD1_2,RVD1_3,1.&
     &0)
      CALL E_TF2W_CFG((IXFMR + 5),0,RVD1_1,50.0,0.1,0.0,RVD1_2,RVD1_3,1.&
     &0)
      CALL E_TF2W_CFG((IXFMR + 6),0,RVD1_1,50.0,0.1,0.0,RVD1_2,RVD1_3,1.&
     &0)
      IF (0.0 .LT. 0.001) THEN
        RVD1_5 = 0.0
        RVD1_6 = 0.0
        IVD1_1 = 0
      ELSE
        RVD1_6 = 0.0
        RVD1_4 = 6.0/(100.0*RVD1_6)
        RVD1_5 = RVD1_4*RVD1_2*RVD1_2
        RVD1_6 = RVD1_4*RVD1_3*RVD1_3
        IVD1_1 = 1
      ENDIF
      CALL E_BRANCH_CFG( (IBRCH(1)+19),SS(1),IVD1_1,0,0,RVD1_5,0.0,0.0)
```

```
      CALL E_BRANCH_CFG( (IBRCH(1)+20),SS(1),IVD1_1,0,0,RVD1_5,0.0,0.0)
      CALL E_BRANCH_CFG( (IBRCH(1)+21),SS(1),IVD1_1,0,0,RVD1_5,0.0,0.0)
      CALL E_BRANCH_CFG( (IBRCH(1)+22),SS(1),IVD1_1,0,0,RVD1_6,0.0,0.0)
      CALL E_BRANCH_CFG( (IBRCH(1)+23),SS(1),IVD1_1,0,0,RVD1_6,0.0,0.0)
      CALL E_BRANCH_CFG( (IBRCH(1)+24),SS(1),IVD1_1,0,0,RVD1_6,0.0,0.0)
      CALL TSAT1_CFG( (IBRCH(1)+25), (IBRCH(1)+26), (IBRCH(1)+27),SS(1),&
     &RVD1_1,RVD1_2,0.2,1.25,50.0,1.0,1.0,0.1)
! 1:[resistive_load] Three phase resistive load
      CALL RESLOAD_CFG(SS(1), (IBRCH(1)+58), (IBRCH(1)+59), (IBRCH(1)+60&
     &),0,0.6,1.732)
! 1:[reactive_load] Three phase inductive load
      CALL INDLOAD_CFG(SS(1), (IBRCH(1)+55), (IBRCH(1)+56), (IBRCH(1)+57&
     &),0,0.24,1.732,50.0)
! 1:[xfmr-3p2w] 3 Phase 2 Winding Transformer
      CALL COMPONENT_ID(ICALL_NO,923147127)
      RVD1_1 = ONE_3RD*1.0
      RVD1_2 = 132.0*SQRT_1BY3
      RVD1_3 = 13.8*SQRT_1BY3
      CALL E_TF2W_CFG((IXFMR + 7),0,RVD1_1,50.0,0.02,0.0,RVD1_2,RVD1_3,1&
     &.0)
      CALL E_TF2W_CFG((IXFMR + 8),0,RVD1_1,50.0,0.02,0.0,RVD1_2,RVD1_3,1&
     &.0)
      CALL E_TF2W_CFG((IXFMR + 9),0,RVD1_1,50.0,0.02,0.0,RVD1_2,RVD1_3,1&
     &.0)
      IF (0.0 .LT. 0.001) THEN
        RVD1_5 = 0.0
        RVD1_6 = 0.0
        IVD1_1 = 0
      ELSE
        RVD1_6 = 0.0
        RVD1_4 = 6.0/(1.0*RVD1_6)
        RVD1_5 = RVD1_4*RVD1_2*RVD1_2
        RVD1_6 = RVD1_4*RVD1_3*RVD1_3
        IVD1_1 = 1
      ENDIF
      CALL E_BRANCH_CFG( (IBRCH(1)+43),SS(1),IVD1_1,0,0,RVD1_5,0.0,0.0)
      CALL E_BRANCH_CFG( (IBRCH(1)+44),SS(1),IVD1_1,0,0,RVD1_5,0.0,0.0)
      CALL E_BRANCH_CFG( (IBRCH(1)+45),SS(1),IVD1_1,0,0,RVD1_5,0.0,0.0)
      CALL E_BRANCH_CFG( (IBRCH(1)+46),SS(1),IVD1_1,0,0,RVD1_6,0.0,0.0)
      CALL E_BRANCH_CFG( (IBRCH(1)+47),SS(1),IVD1_1,0,0,RVD1_6,0.0,0.0)
      CALL E_BRANCH_CFG( (IBRCH(1)+48),SS(1),IVD1_1,0,0,RVD1_6,0.0,0.0)
      CALL TSAT1_CFG( (IBRCH(1)+49), (IBRCH(1)+50), (IBRCH(1)+51),SS(1),&
     &RVD1_1,RVD1_2,0.2,1.25,50.0,1.0,1.0,0.1)

! 1:[source3] Three Phase Voltage Source Model 1 'Source'
      CALL COMPONENT_ID(ICALL_NO,1406920211)
      CALL ESYS651_CFG(3,2,0,0,0,SS(1), (IBRCH(1)+28), (IBRCH(1)+29), (I&
     &BRCH(1)+30),0,0,0, 60.0,60.0,90.0,136.0,0.0,0.0,1.5,132.0,230.0, 1&
     &.0,80.0,2.0,0.5,1.0,0.1, 1.0,80.0,1.0,0.1)

      RETURN
      END

!=======================================================================

      SUBROUTINE MainOut_Begin()

!---------------------------------------
! Standard includes
!---------------------------------------

      INCLUDE 'nd.h'
      INCLUDE 'emtconst.h'
      INCLUDE 's0.h'
      INCLUDE 's1.h'
      INCLUDE 's4.h'
      INCLUDE 'branches.h'
      INCLUDE 'pscadv3.h'
      INCLUDE 'rtconfig.h'

!---------------------------------------
```

```
! Function/Subroutine Declarations
!---------------------------------------
!---------------------------------------
! Variable Declarations
!---------------------------------------
! Subroutine Arguments
! Electrical Node Indices
      INTEGER  NT_5(3), NT_7(3)
! Control Signals
! Internal Variables
      INTEGER  IVD1_1
! Indexing variables
      INTEGER ICALL_NO                          ! Module call num
      INTEGER IT_0                              ! Storage Indices
      INTEGER ISUBS, SS(1), IBRCH(1), INODE     ! SS/Node/Branch/Xfmr
      INTEGER IXFMR
!---------------------------------------
! Local Indices
!---------------------------------------
! Increment and assign runtime configuration call indices
      ICALL_NO  = NCALL_NO
      NCALL_NO  = NCALL_NO + 1
! Increment global storage indices
      NCX       = NCX + 0
      INODE     = NNODE + 2
      NNODE     = NNODE + 30
      IXFMR     = NXFMR
      NXFMR     = NXFMR + 9
! Initialize Subsystem Mapping
      ISUBS = NSUBS + 0
      NSUBS = NSUBS + 1
      DO IT_0 = 1,1
         SS(IT_0) = SUBS(ISUBS + IT_0)
      END DO
! Initialize Branch Mapping.
      IBRCH(1)     = NBRCH(SS(1))
      NBRCH(SS(1)) = NBRCH(SS(1)) + 63
!---------------------------------------
! Electrical Node Lookup
!---------------------------------------
! Array (1:3) quantities...
      DO IT_0 = 1,3
         NT_5(IT_0) = NODE(INODE + 1 + IT_0)
         NT_7(IT_0) = NODE(INODE + 7 + IT_0)
      END DO

!---------------------------------------
! Generated code from module definition
!---------------------------------------
! 50:[multimeter] Multimeter
      IVD1_1 = NRTCF
      NRTCF  = NRTCF + 4
      IF (ABS(1.0) .GT. 1.0E-20) THEN
        RTCF(IVD1_1+1) = 1.0/ABS(1.0)
      ELSE
        RTCF(IVD1_1+1) = 1.0
      ENDIF
      RTCF(IVD1_1+2) = 0.02
! 90:[multimeter] Multimeter
      IVD1_1 = NRTCF
      NRTCF  = NRTCF + 4
! 180:[pgb] Output Channel 'TERMINAL VOLTAGE'
! 190:[pgb] Output Channel 'Current'
! 200:[pgb] Output Channel 'Vload'
! 210:[pgb] Output Channel 'Iload'
! 220:[pgb] Output Channel 'Vs'
! 230:[fft] On-Line Frequency Scanner
! 240:[datatap] Scalar/Array Tap
! 270:[datatap] Scalar/Array Tap
! 330:[pgb] Output Channel 'Is'
! 340:[fft] On-Line Frequency Scanner
```

```
! 350:[datatap] Scalar/Array Tap
! 360:[datatap] Scalar/Array Tap
! 370:[datatap] Scalar/Array Tap
! 380:[datatap] Scalar/Array Tap
! 390:[datatap] Scalar/Array Tap
! 400:[datatap] Scalar/Array Tap
! 410:[datamerge] Merges data signals into an array
! 420:[datatap] Scalar/Array Tap
! 430:[seq_filter] Sequence Filter
! 440:[datamerge] Merges data signals into an array
! 470:[datatap] Scalar/Array Tap
! 480:[datatap] Scalar/Array Tap
! 490:[datatap] Scalar/Array Tap
! 500:[datatap] Scalar/Array Tap
! 510:[datatap] Scalar/Array Tap
! 540:[datatap] Scalar/Array Tap
! 550:[seq_filter] Sequence Filter
! 560:[datamerge] Merges data signals into an array
! 570:[datamerge] Merges data signals into an array
! 580:[datamerge] Merges data signals into an array
! 590:[datamerge] Merges data signals into an array
! 600:[datatap] Scalar/Array Tap
! 610:[datatap] Scalar/Array Tap
! 620:[datatap] Scalar/Array Tap
! 630:[datatap] Scalar/Array Tap
! 640:[datatap] Scalar/Array Tap
! 650:[datatap] Scalar/Array Tap
! 660:[datatap] Scalar/Array Tap
! 670:[datatap] Scalar/Array Tap
! 680:[datatap] Scalar/Array Tap
! 690:[datatap] Scalar/Array Tap
! 700:[datatap] Scalar/Array Tap
! 710:[datatap] Scalar/Array Tap
! 720:[datatap] Scalar/Array Tap
! 730:[datatap] Scalar/Array Tap
! 740:[datatap] Scalar/Array Tap
! 750:[datatap] Scalar/Array Tap
! 760:[datatap] Scalar/Array Tap
! 770:[datatap] Scalar/Array Tap
! 780:[datatap] Scalar/Array Tap
! 790:[datatap] Scalar/Array Tap
! 800:[datatap] Scalar/Array Tap
! 810:[datatap] Scalar/Array Tap
! 820:[datatap] Scalar/Array Tap
! 830:[datatap] Scalar/Array Tap
! 840:[datatap] Scalar/Array Tap
! 850:[datatap] Scalar/Array Tap
! 860:[datatap] Scalar/Array Tap
! 870:[datatap] Scalar/Array Tap
! 880:[datatap] Scalar/Array Tap
! 940:[l2l_imp] Line to Line impedance
      CALL E_L2LIMP1_CFG(0,0.1,-458.8,56.7)
! 950:[l2g_imp] Line to Ground Impedance
      CALL E_L2GIMP1_CFG(1.6,0.0,0.1,-458.8,56.7,0)
! 960:[l2g_imp] Line to Ground Impedance
      CALL E_L2GIMP1_CFG(1.6,0.0,0.1,-458.8,56.7,0)
! 970:[l2g_imp] Line to Ground Impedance
      CALL E_L2GIMP1_CFG(1.6,0.0,0.1,-458.8,56.7,0)
! 980:[datatap] Scalar/Array Tap
! 990:[datatap] Scalar/Array Tap
! 1000:[l2l_imp] Line to Line impedance
      CALL E_L2LIMP1_CFG(0,0.1,-458.8,56.7)
! 1010:[datatap] Scalar/Array Tap
! 1020:[datatap] Scalar/Array Tap
! 1030:[datatap] Scalar/Array Tap
! 1040:[datatap] Scalar/Array Tap
! 1050:[l2l_imp] Line to Line impedance
      CALL E_L2LIMP1_CFG(0,0.1,-458.8,56.7)
! 1060:[mho_circle] Mho Circle
      CALL TRPCCL1_CFG(0,32.0,5.5,31.5)
```

```
! 1070:[mho_circle] Mho Circle
      CALL TRPCCL1_CFG(0,32.0,5.5,31.5)
! 1080:[mho_circle] Mho Circle
      CALL TRPCCL1_CFG(0,32.0,5.5,31.5)
! 1090:[mho_circle] Mho Circle
      CALL TRPCCL1_CFG(0,32.0,5.5,31.5)
! 1100:[pgb] Output Channel 'Xca'
! 1110:[pgb] Output Channel 'Xc'
! 1120:[pgb] Output Channel 'Rca'
! 1130:[pgb] Output Channel 'Rc'
! 1140:[pgb] Output Channel 'Xbc'
! 1160:[pgb] Output Channel 'Xb'
! 1170:[pgb] Output Channel 'Rbc'
! 1180:[pgb] Output Channel 'Rb'
! 1190:[mho_circle] Mho Circle
      CALL TRPCCL1_CFG(0,32.0,5.5,31.5)
! 1200:[pgb] Output Channel 'Xab'
! 1220:[datamerge] Merges data signals into an array
! 1230:[pgb] Output Channel 'Xa'
! 1240:[pgb] Output Channel 'Rab'
! 1250:[pgb] Output Channel 'Ra'
! 1260:[datamerge] Merges data signals into an array
! 1270:[mho_circle] Mho Circle
      CALL TRPCCL1_CFG(0,32.0,5.5,31.5)
      RETURN
      END
```