УДК 004.415.2

ПРОГРАММА УПРАВЛЕНИЯ ЭКСПЕРИМЕНТАЛЬНОЙ ФИЗИЧЕСКОЙ УСТАНОВКОЙ В ПУСКОВОМ РЕЖИМЕ РАБОТЫ

В.М. Павлов, Ю.Н. Голобоков, А.В. Лысенко (Томский политехнический университет, Томск, Россия)

Программа экспериментов на любой исследовательской установке может меняться очень динамично по составу задействованных диагностических и управляющих подсистем, длительности и последовательности выполняемых операций. По этой причине персонал установки должен иметь удобный программный инструмент для описания и последующего исполнения циклограмм эксперимента. Одним из возможных путей решения данной проблемы является использование специального языка описания эксперимента и программной среды для интерпретации подготовленных сценариев. Эти функции реализуются в предлагаемом авторами программном обеспечении, которое разработано для казахстанского материаловедческого токамака КТМ. Использование разработанного языка даёт возможность оперативному персоналу установки описать сценарий проведения эксперимента без участия разработчиков программного обеспечения. Программное обеспечение в полной своей функциональности использовалось при проведении пусков и получении первой плазмы на казахстанском токамаке КТМ.

Ключевые слова: экспериментальная физическая установка, система автоматизации, программа управления, язык описания сценария эксперимента.

THE CONTROL SOFTWARE OF EXPERIMENTAL PHYSICAL DEVICE IN OPERATION MODE. V.M. PAVLOV, Yu.N. GOLOBOKOV, A.V. LYSENOK. The experiment schedule for any scientific device may vary very fast including changes in the set of used diagnostic and control subsystems. The duration of experiment and sequence of operations may also vary from one experiment to another. This requires flexible and convenient software to describe and run experiment scenario. One of the solutions for this problem is to develop a special language for experiment scenario description and software to execute such scenario programs. These functions are implemented in the software, developed by the authors of the article for Kazakhstan KTM tokamak. The software allows scientists to make and run flexible scenarios without control software modification. The software was tested and used in the series of experiments to prepare the KTM to the first plasma shot and when obtaining the first plasma.

Key words: experimental physical device, automation system, control software, experiment scenario description language.

Эффективность проведения экспериментальных исследований на термоядерных установках типа токамак во многом определяется наличием комплексной системы автоматизации и её функциональными возможностями. Токамак КТМ и комплекс его технологических систем как объект автоматизации обладает рядом особенностей, накладывающих дополнительные требования к системе автоматизации. К таким особенностям относятся, с одной стороны, многофункциональность технологических и диагностических систем при их достаточно большом разнесении в пространстве, а с другой — наличие различных режимов работы, каждому из которых соответствуют определённая динамика управляемых процессов, порядок включения оборудования, состав и диапазоны контролируемых физических параметров.

Программа каждого эксперимента может меняться по составу, длительности и последовательности выполняемых в ходе эксперимента операций, при этом необходимо организовать синхронную работу всех подсистем, участвующих в эксперименте с его фазами. Задача осложняется тем, что в зависимости от целей планируемого эксперимента некоторые подсистемы могут быть отключены или включены, кроме этого, локальные нарушения работы на уровне подсистемы могут потребовать принятия централизованных решений о дальнейшем ходе эксперимента, в том числе и о необходимости противоаварийного останова всего комплекса.

Учитывая то, что продолжительность всего пуска составляет порядка нескольких секунд, очевидно, что с такой скоростью производить управление вручную с пульта невозможно и требуется организация автоматического режима управления.

Из анализа процессов проведения экспериментов на токамаке КТМ выделены следующие две группы задач синхронизации оборудования в ходе пусков: задачи временной и событийной синхронизации. Временная синхронизация заключается в привязке циклов измерения и управления к экспериментальному времени. Событийная синхронизация, рассматриваемая в дальнейшем в статье, предназначена для привязки операций, выполняемых всеми узлами распределённой системы автоматизации, к меткам событий.

Основным событием в системе автоматизации является время перехода из одной фазы (стадии) эксперимента в другую в соответствии с циклограммой эксперимента, представленной на экране пусковых параметров (рис. 1).

Рис. 1. Экран пусковых параметров

Циклограмма эксперимента формируется ведущим физиком токамака и утверждается в виде организационного документа «Диаграмма пусковой последовательности». Пример диаграммы пусковой последовательности для одного из тестовых пусков токамака КТМ приведён в таблице.

Диаграмма пусковой последовательности

Время начала	Время окончания	Команда
—	—	Включение пульта управления
—	—	Включение питания блоков управления и сбора данных
—	—	Задание номера разряда
—	—	Редактирование параметров разряда
—	—	Редактирование параметров настройки подсистем управления
—	—	Передача параметров настройки в подсистемы управления
—	—	Инициализация и тестирование подсистем управления и подсистем сбора и регистрации данных
0 мс	1100 мс	Ввод тока в обмотку TF
0 мс	1100 мс	Запуск подсистемы электромагнитной диагностики
995 мс	1100 мс	Запуск подсистем сбора и регистрации данных
1000 мс	1100 мс	Старт процесса разряда конденсаторной батареи на обмотку CS
1005 мс	1100 мс	Включение системы предионизации
1010 мс	1100 мс	Ввод тока в обмотку PF4
—	—	Сохранение измеренных данных в локальной памяти подсистем управления и сбора данных
—	—	Передача данных на сервер регистрации ИИС

В связи с тем, что циклограмма эксперимента на исследовательской установке, как правило, меняется очень динамично и предусмотреть все допустимые варианты заранее нельзя, так как они зачастую складываются постепенно в ходе эксплуатации, в программном обеспечении системы автоматизации необходимо предусмотреть возможность его гибкого изменения и конфигурирования оператором токамака без участия программиста-разработчика. Наиболее удобным инструментом для реализации такой функции является адаптированный к конкретной области применения язык управления экспериментом интерпретирующего типа, который обеспечивает запись командных выражений в простой форме [1]. Используя этот язык, оператор может активизировать выполнение последовательностей команд управления установкой или вызывать на исполнение заранее подготовленные файлы с описанием программы эксперимента. Для уменьшения количества ошибок оперативного персонала в этом случае могут быть предусмотрены средства синтаксического и семантического анализа языка управления экспериментом.

Исходя из этого, для реализации задачи управления в процессе проведения разряда на КТМ было принято решение разработать специализированное программное приложение Control_Kernel, выполняющее функции событийной синхронизации в пусковом режиме работы установки. Основное назначе-

ние данной программы — интерпретация циклограммы эксперимента, задаваемой оператором, и управление устройствами распределённой системы автоматизации.

Поясним работу программы Control_Kernel при помощи схемы, изображённой на рис. 2. На рисунке показаны основные элементы системы и диаграммы изменения их состояний в процессе взаимодействия в пусковом режиме работы установки. Желаемая циклограмма эксперимента готовится оператором токамака и размещается в файле states.diag (имя по умолчанию) на сервере регистрации данных ИИС. По команде оператора о начале разряда программа Control_Kernel загружает файл циклограммы и начинает автоматическое управление подсистемами в соответствии с заранее определённым сценарием. Управление подсистемами производится с помощью команд, которые передаются с помощью сети Ethernet по специализированному протоколу T-ICS [2]. Подсистема, принявшая команду, осуществляет смену своего состояния в соответствии со схемой переходов, изображенной справа на рис. 2. Смена состояний происходит циклически. Переход подсистемы в новое состояние приводит к выполнению заданных для этого состояния операций.

Рис. 2. Диаграммы изменения состояний элементов системы управления

Диаграмма работы программы Control_Kernel также имеет циклический вид (слева на рис. 2). Рассмотрим её более подробно. При запуске программа Control_Kernel автоматически выполняет переход в состояние OnLine. В данном состоянии программа ожидает сетевого соединения, которое может инициироваться пультом общего управления (ПОУ) либо другой разрешённой системой. После установки соединения программа ожидает дальнейших команд по установленному сетевому соединению.

Затем при получении от оператора пульта команды cmdSendConfig программа переходит в состояние SendConfig. В данном состоянии программа считывает и обрабатывает файл настроек и файл диаграммы состояний.

Из состояния SendConfig по команде cmdPrestart от ПОУ программа переходит в состояние Prestart. В данном состоянии программа устанавливает связь с подчинёнными подсистемами, участвующими в текущем эксперименте, и ожидает подачи команды cmdStart, означающей начало разряда. Перечень активных подсистем текущего эксперимента задаётся в файле настроек Control_Kernel.

После получения команды Start от ПОУ программа переходит в состояние Start. В данном состоянии программа выполняет заданную циклограмму эксперимента путём выдачи команд подчинённым подсистемам в соответствии с файлом настроек и диаграммой смены состояний. После выполнения программы эксперимента ControlKernel автоматически переходит в состояние Stop.

Из состояния Stop программа может перейти в состояние OffLine при приходе команды cmdOffLine от ПОУ. При этом программа закрывает все сетевые соединения, освобождает выделенную память, сохраняет журнал, закрывает все открытые файлы и перезапускается.

Если при выполнении каких-либо операций при выполнении циклограммы возникают ошибки, требующие прерывания эксперимента, программа переходит сама и переводит все подсистемы в состояние Terminate, в котором выполняются операции останова экспериментального комплекса. Код возникшей ошибки при этом сохраняется и доступен оператору для анализа. Из этого состояния программа переводится в состояние OffLine только командой с пульта.

В режимах тестирования и настройки системы автоматизации остаётся возможность ручного управления подсистемами с помощью прямой выдачи команд оператором с пульта общего управления без использования программы Control_Kernel.

В режиме пуска циклограмма эксперимента формируется программой Control_Kernel путём интерпретации сценария выдачи команд управления, заранее сохранённого оператором в специальном файле, структура которого представляет собой набор текстовых конструкций языка описания эксперимента.

Ключевой особенностью языка является сетевая рассылка управляющих команд подсистемам, которые участвуют в эксперименте, а также механизмы, позволяющие чётко указать время формирования этих управляющих команд. В сценарии каждой подсистеме присваивается тип, указывающий степень её значимости. Существуют следующие типы подсистем:

- CRITICAL — система, функционирование без которой невозможно (этот тип имеют все подсистемы по умолчанию);
- VALUABLE — система, функционирование без которой возможно, но нежелательно;
- OPTIONAL — система не является критической, разряд будет продолжаться и без связи с данной системой.

Для удобства некоторые подсистемы, схожие по назначению в сценарии, можно объединить в одну группу и присвоить ей уникальное имя.

При написании сценария можно использовать модульный подход, т.е. объединять несколько команд в один функциональный блок (подпрограмму). Каждой подпрограмме присваивается уникальное имя. Обычно переходу установки и системы управления в каждое новое состояние соответствует вызов соответствующей подпрограммы. Описать подпрограмму в сценарии можно с помощью конструкции:

```
DEFINE STATE <STATE_NAME>
{<ACTION_1>
<ACTION_N>},
```

где <STATE_NAME> — имя, присваиваемое группе команд (подпрограмме); <ACTION_1>, ..., <ACTION_N> — команды, выполняемые в подпрограмме.

Например, для инициализации систем управления источниками импульсного электропитания и проверки правильности инициализации можно определить следующее состояние (подпрограмму):

```
DEFINE STATE INIT_POWER_SUPPLY
{EXECUTE COMMAND POWER_SUPPLY INIT
IF RC = FAIL CHSTATE POWER_FAIL}
```

В программе обязательно должны присутствовать две подпрограммы с зарезервированными именами:

- DEFINE STATE MAIN — начальное состояние программы. Команды этой подпрограммы выполняются при запуске циклограммы эксперимента;
- DEFINE STATE TERMINATE — конечное состояние программы. Это состояние выполняется перед завершением выполнения диаграммы.

Для того чтобы запустить описанную программу перехода в новое состояние системы, можно использовать конструкцию CHSTATE <STATE>, где <STATE> — имя запускаемого состояния. После вы-

полнения всех команд подпрограммы выполнение возвращается в подпрограмму, из которой произошёл вызов.

Управление подсистемами осуществляется путём формирования управляющих команд. Каждой команде соответствует действие, которое должна выполнить подсистема. В настоящее время в сценарии можно использовать следующие типы команд:

- Init — первичная инициализация и тестирование подсистемы;
- Prestart — ввод подсистемы в состояние предпусковой готовности;
- Start — перевод подсистемы в пусковой режим;
- Stop — остановка подсистемы;
- Terminate — полная остановка подсистемы;
- Sendconfig — команда, указывающая подсистеме, что необходимо загрузить конфигурационные файлы с сервера.

Отправка команды подсистеме описывается в сценарии конструкцией:

EXECUTE COMMAND <SYSTEM_NAME> <COMMAND>,

где <SYSTEM_NAME> — имя подсистемы, которой необходимо отправить команду; <COMMAND> — тип команды, которую необходимо отправить подсистеме.

Важной особенностью языка является возможность централизованной обработки ответов подсистем, на основании которых могут приниматься глобальные решения о дальнейшем ходе эксперимента. После выполнения команды подсистема посылает ответ, в котором указывает код успешного выполнения или код возникшей ошибки. Этот код записывается в переменную RC и может быть обработан специальной конструкцией в сценарии. При исполнении команды для группы подсистем переменная RC принимает значение самой большой произошедшей ошибки в подсистеме с максимальным приоритетом. Текущее значение переменной RC в сценарии можно сравнивать на равенство или неравенство одному из принимаемых значений, и в зависимости от результата сравнения выполняется запуск перехода системы в новое состояние, например, аварийный останов. Эту операцию выполняет конструкция условного запуска состояния в зависимости от переменной RC:

IF RC = <RC_CODE> CHSTATE <STATE>,

где <RC_CODE> — одно из принимаемых переменной RC значений, с которым будет производиться сравнение текущего значения; <STATE> — имя состояния, которое необходимо запустить, если результат сравнения будет «ИСТИНА». Допускается проверка <RC_CODE> как на равенство, так и на неравенство заданному значению.

В отличие от других языков описания сценариев в разработанном языке поддерживаются операции со счётчиком времени. Временные метки можно задавать с точностью до 1 мс. При написании сценария команды, жёстко привязанные к заданным моментам времени, вызываются при помощи следующих конструкций языка:

— IF TIME = <TIME> CHSTATE <STATE> — вызов подпрограммы для изменения состояния системы в зависимости от времени, прошедшего от начала эксперимента;

— IF TIME = <TIME> WAIT — задерживает выполнение диаграммы пока истинно логическое выражение.

В этих конструкциях: TIME = <TIME> — логическое выражение; <STATE> — имя подпрограммы, которую необходимо запустить, если результат логического выражения будет «ИСТИНА». В логическом выражении, кроме переменной TIME, можно использовать переменные BEGINTIME и ENDTIME, значение которых равно времени начала и окончания выполнения предыдущей операции диаграммы от начала эксперимента в миллисекундах. Возможно также применение простых арифметических выражений и логических операторов отношения, проверки на равенство и неравенство.

Пример описания сценария эксперимента, начиная от подготовительных этапов и заканчивая сохранением и передачей результатов на сервере регистрации данных, для одного из тестовых разрядов токамака КТМ показан на рис. 3. Приведённая программа соответствует диаграмме пусковой последовательности из таблицы.

```

DEFINE GROUP DIAG HADA SPOBZ SYNC
DEFINE GROUP SUBSYS TF EMD CS ION PF4

DEFINE STATE S_DIAG
{
EXECUTE COMMAND TF Start
EXECUTE COMMAND EMD Start
IF TIME<995 Wait
EXECUTE COMMAND DIAG Start
IF TIME<BEGINTIME+5 Wait
IF RC=FAIL CHSTATE TERMINATE
EXECUTE COMMAND CS Start
IF TIME<BEGINTIME+5 Wait
EXECUTE COMMAND ION Start
IF TIME<BEGINTIME+5 Wait
EXECUTE COMMAND PF4 Start
}

DEFINE STATE MAIN
{
CHSTATE S_DIAG
CHSTATE TERMINATE
}

STATE TERMINATE
{
IF TIME<1100 Wait
EXECUTE COMMAND SUBSYS Stop
EXECUTE COMMAND DIAG Stop
}

```

Группировка подсистем для одновременной отправки команд

Описание состояния — функциональных блоков диаграммы разряда

Выполнение команды подсистемой

Ожидание заданного момента времени

Выполнение команды группой подсистем

В случае ошибки переход в завершающее состояние

Ожидание заданного момента времени с момента отправки предыдущей команды

Основное состояние программы, которым начинается её выполнение

Основное состояние программы, которым заканчивается её выполнение

Язык Control Kernel

- * Поддержка подпрограмм
- * Поддержка условных операторов и переходов
- * Поддержка абсолютной и относительной временной синхронизации событий
- * Поддержка группирования подсистем
- * Поддержка обработки кодов ошибок подсистем

Рис. 3. Программа описания сценария эксперимента

Для уменьшения вероятности ошибок оператора программа Control_Kernel автоматически выполняет проверку подготовленного файла с описанием сценария эксперимента на наличие синтаксических ошибок. При обнаружении ошибки оператору на дисплей выдаётся список с указанием имён файлов и номеров строк, в которых обнаружены ошибки.

Как уже отмечалось, для рассылки команд управления подсистемами управления и диагностики при исполнении сценария используется протокол T-ICS. Протокол используется на всех уровнях системы автоматизации токамака КТМ: при взаимодействии интеллектуальных модулей УСО через системную шину ЭВМ, шину магистрально-модульной системы; при подключении через последовательные интерфейсы (рис. 4). Протокол предназначен для унификации операций настройки, тестирования и обмена пакетами информации с модулями УСО.

Рис. 4. Характеристики протокола взаимодействия устройств САЭ (T-ICS)

Протокол также используется для организации сетевого взаимодействия с подсистемами автоматизации. В этом случае T-ICS реализован на базе транспортного протокола UDP/IP. Ведущий и подчинённые узлы обмениваются данными в режиме запрос—ответ. В пакете можно передавать с высоким уровнем надёжности до 63 байт данных.

Основное назначение протокола — передача команд управления. Для передачи больших объёмов измеренной в ходе эксперимента информации параллельно с протоколом T-ICS используются встроенные в операционную систему протоколы прикладного уровня FTP и SMB.

ЗАКЛЮЧЕНИЕ

Подводя итоги, следует отметить, что в целом описанная программа удовлетворяет заявленным требованиям. Предложенное решение задачи позволяет выполнять управление разрядом на КТМ. Разработанный специализированный язык позволяет быстро описать сценарий проведения эксперимента.

Благодаря реализованному протоколу сетевого взаимодействия всех подсистем САЭ оператор может централизованно наблюдать за ходом проведения эксперимента на пульте общего управления и своевременно реагировать на аварийные ситуации. Допускается удалённое участие в подготовке и проведении эксперимента исследователей, имеющих соответствующее программное обеспечение, поддерживающее протокол T-ICS.

На сегодняшний день разработка программы общего управления разрядом Control_Kernel завершена. Программа в полной своей функциональности использовалась при проведении пусков и получении первой плазмы на казахстанском токамаке КТМ.

REFERENCES

1. **Timonin V.V., Fyakhretdinov A.N.** Organization of the computing process in the systems of the automation of experiments on physical installations: Preprint IAE -3777/15, 1983. 15 p. (in Russian).
2. **Lysenok A.V., Golobokov Yu.N., Pavlov V.M.** Data exchange library by protocol T-ICS. Certificate of computer program state registration № 2010612945, 30.04.2010 (in Russian).

AUTHORS

Pavlov V.M. Tomsk Polytechnical University, pr.Lenina 30, 834050 Tomsk, Russia; pavlov@tpu.ru.

Golobokov Yu.N. Tomsk Polytechnical University, pr.Lenina 30, 834050 Tomsk, Russia; golobokov@tpu.ru.

Lysenok A.V. Tomsk Polytechnical University, pr.Lenina 30, 834050 Tomsk, Russia; lisand@myrambler.ru.

Received 9 June 2011

Problems of Atomic Science and Technology
Ser. Thermonuclear Fusion, 2011, issue 3, pp. 81—87