http://dx.doi.org/10.5516/NET.04.2014.012

KTM TOKAMAK OPERATION SCENARIOS SOFTWARE INFRASTRUCTURE

V. PAVLOV[1], K. BAYSTRUKOV[1], YU. GOLOBOKOV[1*], A. OVCHINNIKOV[1], A. MEZENTSEV[1], S. MERKULOV[1], A. LEE[1], I. TAZHIBAYEVA[2], and G. SHAPOVALOV[2]

[1]National Research Tomsk Polytechnic University (TPU) 634050, Russian Federation, Tomsk, Lenin Avenue, 30

[2]National Nuclear Center (NNC), 071100, Kazakhstan, Kurchatov, st.Krasnoarmeyskaya, 2 buil.54 B

[*]Corresponding author. E-mail : golobokov@tpu.ru

Received February 10, 2014
Accepted for Publication May 25, 2014

One of the largest problems for tokamak devices such as Kazakhstan Tokamak for Material Testing (KTM) is the operation scenarios' development and execution. Operation scenarios may be varied often, so a convenient hardware and software solution is required for scenario management and execution. Dozens of diagnostic and control subsystems with numerous configuration settings may be used in an experiment, so it is required to automate the subsystem configuration process to coordinate changes of the related settings and to prevent errors. Most of the diagnostic and control subsystems software at KTM was unified using an extra software layer, describing the hardware abstraction interface. The experiment sequence was described using a command language.

The whole infrastructure was brought together by a universal communication protocol supporting various media, including Ethernet and serial links. The operation sequence execution infrastructure was used at KTM to carry out plasma experiments.

KEYWORDS : Tokamak, KTM, Automatic Control, Embedded Software, Experiment Scenario Description Language

1. INTRODUCTION

One of the largest problems for research fusion devices, as well as for ITER, is the operation scenarios' development, combined with accurate and reliable execution of the scenarios. The latter is mostly provided by the software infrastructure, including utilities for scenario parameters configuration, real-time scenario execution, network communication and event logging support.

A typical situation for the operation scenarios of research devices is that the scenarios may be varied very often, including changes in the diagnostic and control subsystems operating sequence, duration of the discharge, and variations of the list of subsystems used in the experiments. Thus, a convenient hardware and software solution is required to manage and execute the operation scenarios.

2. THE EXPERIMENT AUTOMATION SYSTEM STRUCTURE

The Kazakhstan Tokamak for Material Testing (KTM) fusion device [1] automation system is a hierarchical distributed system functionally divided into units [2] shown in Fig. 1.

2.1 Plant Control System

The plant control system is used to control such slow continuous or cyclic processes as vessel heating, vacuuming, cleaning boronization and water cooling.

2.2 Discharge Control System

The discharge control system is used for the following:

- Overall experiment flow control by sending command messages over the network to diagnostic and control subsystems used during the discharge;
- Distribution of synchronization and timing signals through the fiber optic lines to provide event time tagging and synchronous operation of all subsystems during the discharge;
- Reception and processing of asynchronous events including alarm signals for equipment and instrumentation protection;
- Plasma control at discharge mode for required plasma parameters achievement and plasma confinement.

The discharge control system is based on a multiprocessor VME system for parallel computations and solving control tasks.

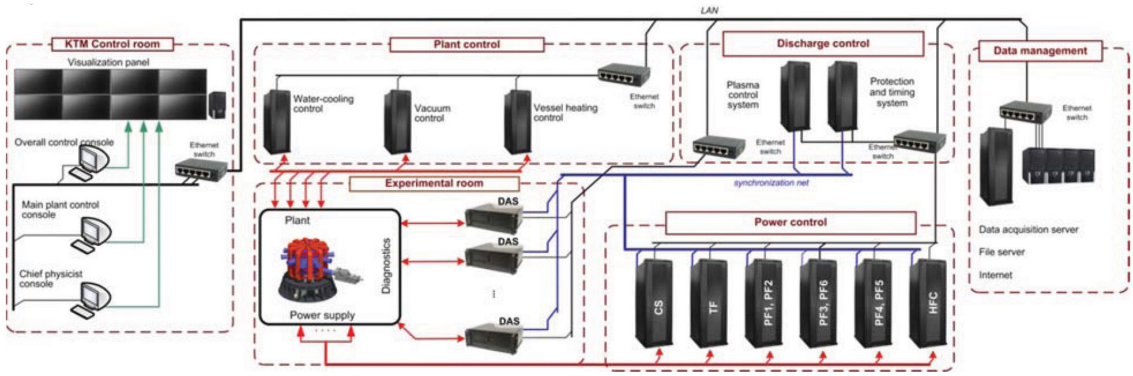

Fig. 1. The KTM Tokamak Experiment Automation System Structure.

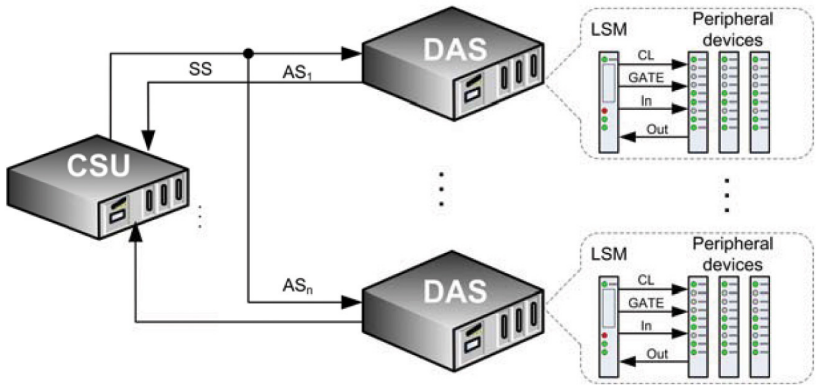

Fig. 2. Synchronization and Timing System Structure.

2.3 Magnetic Coils Power Supply Control System

The magnetic coils power supply control system is used for power supply thyristors control using a phase control method, data acquisition, data logging, alarm signal generation, coil current and voltage monitoring.

2.4 Data Acquisition System

The data acquisition system (DAS) provides functions that are vital for a research device: data acquisition, logging, analysis, and experiment results visualization. DAS incorporates diagnostic data acquisition subsystems, data logging servers, data post-processing servers and software to provide access to the experimental data for researchers. A technical description of the KTM tokamak DASs can be found in [3-7].

3. DISCHARGE CONTROL MODE

As a result of the experiment analysis it was found that there are two groups of tasks of synchronization during the discharge at the KTM tokamak: timing and event synchronization. The task of timing is to synchronize data sampling cycles with the relative experiment time scale with the accuracy of up to 100 ns. The task of event synchronization is to synchronize all the distributed control subsystems operations with event time markers with the accuracy of up to 100 μs.

The synchronization functions are implemented by the discharge control system. The hardware used for this system is organized as a two-level system with a central synchronization unit (CSU) based on a VME crate on the upper level and local synchronization modules (LSM) on the lower level. Local synchronization modules are installed in every plant control subsystem and diagnostics data acquisition subsystem (Fig. 2). These modules are unified and have a typical set of synchronization inputs and outputs. The central synchronization unit and local synchronization modules are interconnected using a dedicated optical network that provides real-time signals and data transmission. This structure provides a number of synchronization levels with different precision and flexibility: the centralized timing provides a 10MHz reference timing signal to clock the DAQ systems, the hardware event processing at CSU and LSM provides up to a 10 μs accuracy of event synchronization tied to the LSM and CSU digital I/O pins used as DAQ Gate signals, interlocking and safety signals, fast hardware control signals. Also, an additional level of synchronization includes transferring synchronization events to a Linux/Xenomai OS domain

using hardware interrupts with a latency of up to 30 μs for the transferring of events to the user level. The control system also provides event synchronization using the Ethernet network during the total time of the experiment, including the preparation phase and final data transfer to the data server. It is the slowest, but most flexible synchronization method, best suited for the preparation phase as its latency may be up to a few milliseconds. The main event for the automation system is the transfer from one state to another, according to the experimental states sequence diagram (Fig. 3).

The sequence of operations during the experiment is developed by the leading physicist of tokamak and approved as an organization document. Table 1 contains an example of one of the test purpose experiments at KTM tokamak.

4. THE MAIN CONTROL CONSOLE SYSTEM FUNCTIONS AND SOFTWARE

The main control console system (MCCS) of the KTM tokamak is widely used in setup and equipment test modes and for short discharges control.

In all modes, the MCCS provides workplaces for the following tokamak operators (Fig. 4): experiment leader, leading physicist, leading technologist and diagnostics operators. Remote participation in the experiments is also provided by MCCS software.

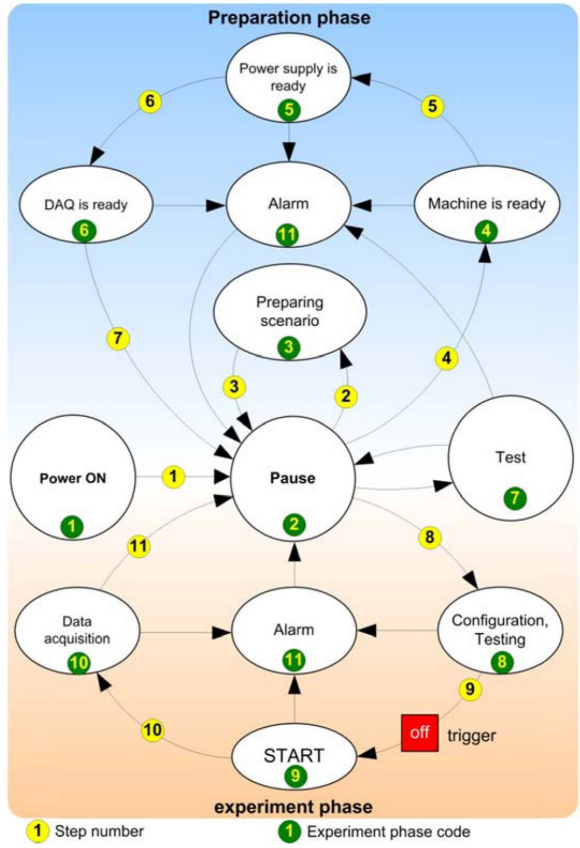

Fig. 3. Experiment Sequence Diagram.

Table 1. Operation Sequence

Relative event begin time, ms	Relative event end time, ms	Event
-	-	Control console power up
-	-	DAS units power up
-	-	Setting the shot number
-	-	Editing the shot configuration parameters
-	-	Editing the DAQ subsystems parameters
-	-	Sending setup parameters to the control subsystems
-	-	Initialization and self-test of control subsystems and DAQ subsystems
0	1100	Rising TF coil current
0	1100	Starting magnetic diagnostics data acquisition
995	1100	Starting data acquisition for all subsystems
1000	1100	CS condenser battery discharge to the CS coil
1005	1100	Starting pre-ionization
1010	1100	Rising PF4 coil current
-	-	Saving acquired data to local DAQ subsystems storage
-	-	Transmission of acquired data to data acquisition server

The MCCS software consists of a set of MS Windows applications. One of these applications is a "Pult_Main" application developed using the SCADA/HMI system "Adastra Trace Mode 6" for machine and diagnostics control. The "Pult_Main" application provides a graphical interface for technological and experiment parameters visualization, sending commands to subsystems, and creation and editing of the experiment sequence diagram.

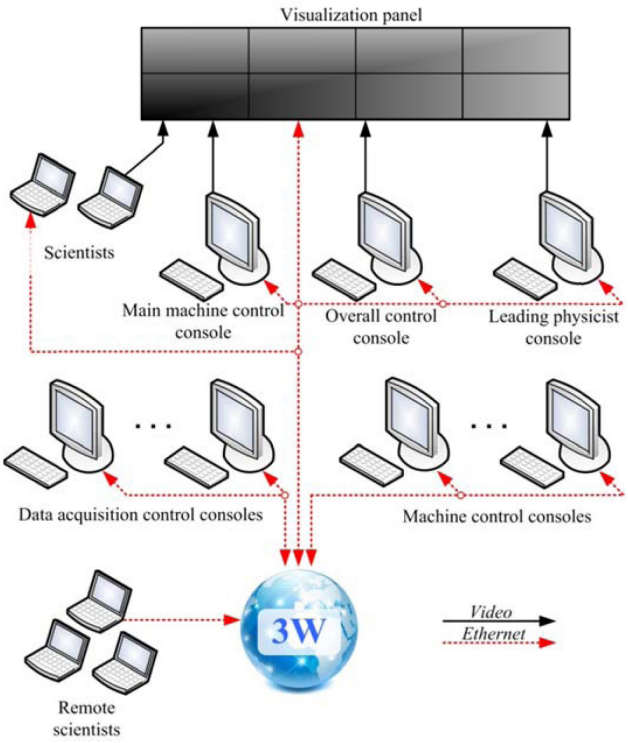

Fig. 4. KTM Tokamak Control Consoles.

The main control console system is used in the following KTM modes:

- Experiment preparation;
- Discharge preparation;
- Inter-discharge pause;
- Discharge mode;

In the experiment preparation mode, the MCCS provides functions for switching on/off the power of technological and diagnostics equipment, parameters and readiness monitoring of the equipment required for the current experiment, and remote setup and testing of the diagnostic system elements.

In the discharge preparation and the inter-discharge modes, the MCCS provides the discharge scenario preparation, pre-analysis of experiment results, transfer of the new experiment parameters to the subsystems, monitoring of parameters and readiness of the equipment, and testing of synchronization and interlocking subsystems.

In the discharge mode, according to the experiment diagram, the MCCS may issue commands to start machine control subsystems, power supply subsystems, and plasma diagnostic subsystems. It is also used for visualization of the reference and real experiment parameters, together with the results of plasma shape reconstruction and video from cameras (Fig. 5).

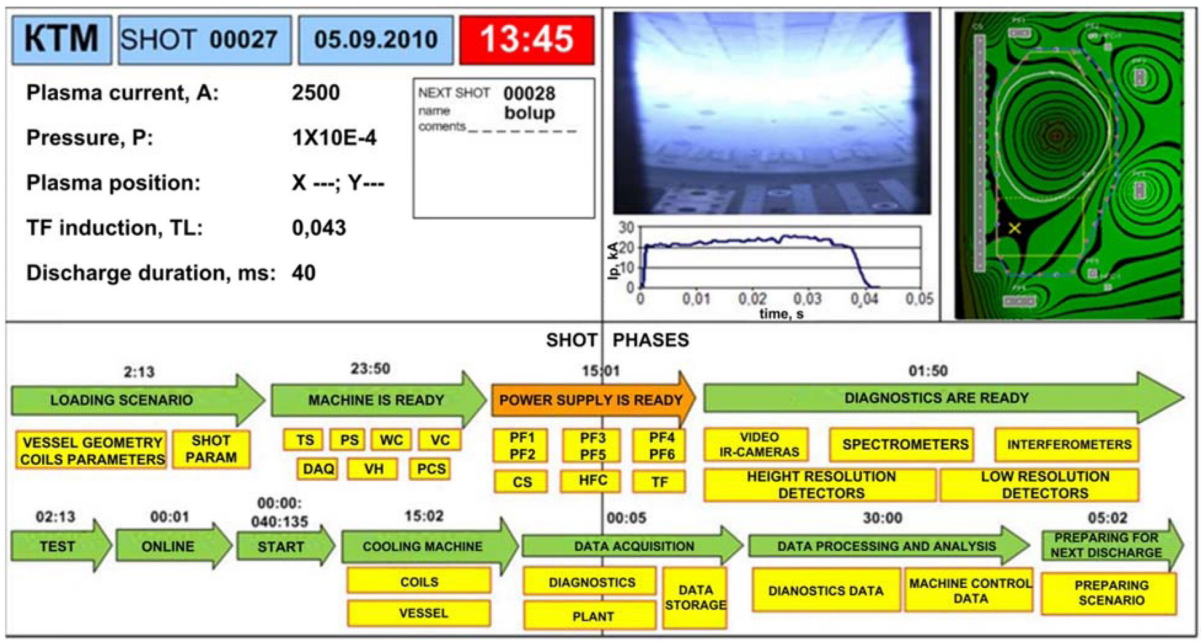

Fig. 5. The Parameters Screen at the Discharge Mode.

5. THE EXPERIMENT CONTROL LANGUAGE DESCRIPTION

Although the main control console system software provides the ability of manual control over the control and data acquisition subsystems, this mode is generally used for testing purposes. The automated mode with the experiment diagram description using a special command language and further on-line interpretation of this diagram is most preferred in the discharge mode [8].

The language is based on the group of commands for transferring the KTM subsystems to one of the allowed states, as in Fig. 6. The changes of states are cyclical as a rule. The process of transfer from one state to other leads to the execution of operations associated with this transfer.

The main logical states for devices, subsystems, and language constructions are shown in Table 2.

The experiment sequence description may have a priority tag for every subsystem. Subsystems may have such priorities as:

— CRITICAL — a subsystem that must be functioning during the experiment (it is the default tag);
— VALUABLE — a subsystem that is desired to be functioning during the experiment;
— OPTIONAL — a subsystem that may not function during the experiment, but the shot sequence will not be interrupted without this.

Some subsystems may be bound together into one group with a unique name to make the configuration process easier.

The experiment sequence definition must have two subroutines with the reserved names:

— DEFINE STATE MAIN — the initial state of the sequence. Commands defined at this state are executed at the beginning of the experiment sequence;
— DEFINE STATE TERMINATE — the final state of the sequence. This state is defined before the execution of a diagram.

One of the advantages of this language is an ability to globally process the responses from all subsystems. It may be used to automatically make decisions during the execution of an experimental sequence. After the command execution, the subsystem sends a response with an error code or a success message. This code may be processed during the sequence using the value of the RC variable. According to the return code, a new change of state may be executed, e.g. emergency stop. The language may bind the execution of commands to the experiment relative time with an accuracy of 1 ms.

In these constructions: The BEGINTIME and ENDTIME variables may also be used in the logical expression. The values of these variables are the time of the beginning of the execution of the last operation and the end of the execution, respectively. The time is counted in milliseconds from the beginning of the experiment. Also, simple arithmetical expressions and comparison operators may be used in the time expressions.

To protect against possible sequence description errors, the interpreter software analyzes a supplied experiment sequence description file prior to its execution. It provides consistency check algorithms and a validation check that compares to the current plant operation status and operation limit data table. If an error occurs, it sends a message with the line number and error-type information to the operator.

Communication with all of the control and acquisition subsystems is made using in-house developed T-ICS protocol over the UDP/IP Ethernet or serial link in the preparation phase, and using a dedicated optical event and timing network during the discharge.

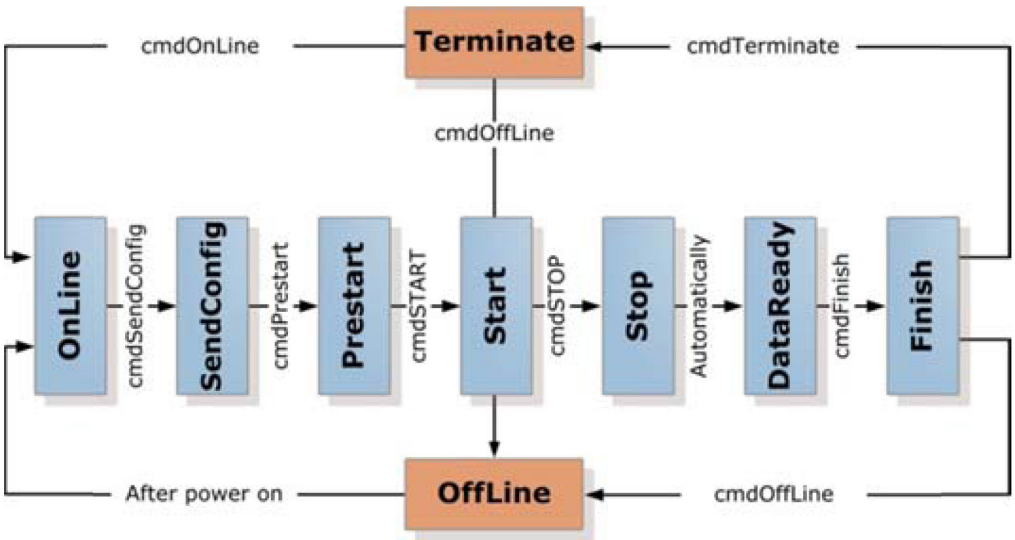

Fig. 6. Control and Data Acquisition System States.

Table 2. Main Logical States and Language Constructions

Main logical states	
ONLINE	Subsystem after power-up and successful initialization
SENDCONFIG	Subsystem after successful reception of the configuration parameters for the ongoing experiment. It designates that parameters were successfully received and passed validity check
INIT	Subsystem after successful initialization
PRESTART	The pre-start state. It designates that the subsystem is ready for discharge
START	The state during the discharge
STOP	The state of system right after the end of discharge. Data acquisition systems generally stop acquisition at this state
DATAREADY	The state means that all experimental data were saved to the local subsystem storage
FINISH	The state means that all experimental data were successfully transferred to the data acquisition server
Language constructions	
DEFINE STATE <STATE_NAME> {<ACTION_1> <ACTION_N>}	Command group (subroutine). A set of commands may be organized into one functional unit (like a subroutine). Every subroutine may be assigned a unique name and it may be defined using special DEFINE STATE construction where <STATE_NAME> is a name assigned to the group of commands (subroutine) and <ACTION_1>, …, <ACTION_N> are commands that are executed in the subroutine.
CHSTATE <STATE>	Starting the defined subroutine for transferring to a new system state, one may use the CHSTATE keyword, where <STATE> is a name of the state to transfer to. After the execution of all the subroutine commands, a return to calling state is made.
EXECUTE COMMAND <SYSTEM_NAME> <COMMAND>,	To send a command to the subsystem, the EXECUTE COMMAND construction is used where <SYSTEM_NAME> is the name of subsystem to send a command to; <COMMAND> is a command type.
IF RC = <RC_CODE> CHSTATE <STATE>,	Conditional operator using variable. This operation may be implemented using IF keyword where <RC_CODE> is one of the RC values to compare the actual value to; <STATE> is a name of state to go to if the comparison will produce “truth” result. The <RC_CODE> may not only be compared to using the “equal to” operator, but also may be used for “not equal to”.
IF TIME = <TIME> CHSTATE <STATE>	Conditional operator using time. TIME = <TIME> is a logical expression; <STATE> is a state change subroutine, that is required to execute if the result of the logical expression is true.
IF TIME = <TIME> WAIT	Conditional operator using time. The execution waits while the logical expression is true.

After the preparation of the sequence description file at the main control console, using special software or an ordinary text editor, the file is placed in a special “model” subdirectory at the data storage server by the main control console software. After that the preparation to execute the programmed sequence may be started. It includes the transfer of a file to the discharge control system by the data storage server software, after the command issued by the main control console, and the sequence correctness is checked by the discharge control system. If all the steps are passed successfully, the operator may start the experiment sequence execution by the discharge control system.

6. THE CONTROL AND DATA ACQUISITION SUBSYSTEMS SOFTWARE

The control and data acquisition devices at KTM are mainly based on the Industrial PC architecture. As a basic operating system, GNU/Linux was chosen, except for some special diagnostics that have proprietary device drivers only for MS Windows.

The majority of control and data acquisition devices with low data flow rates (up to 5 Mbyte/s) use non-real-time Linux with a read-only root file system. On the contrary, high performance diagnostics use special, fast, data acquisition devices, the drivers of which were adopted for a real-time

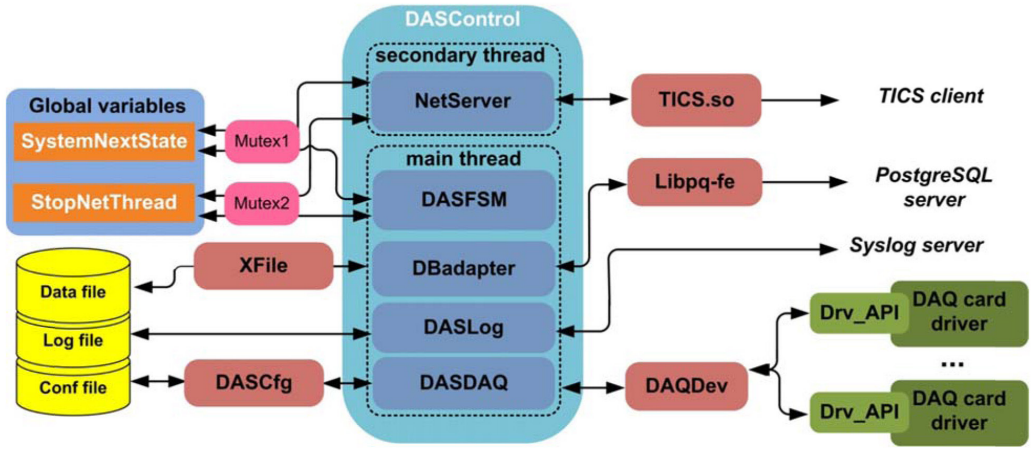

Fig. 7. The Software Structure of the Data Acquisition and Control Subsystems.

Linux configuration under an open Xenomai[1] framework.

The control of the data acquisition subsystem is made using DASControl software [7]. Its general structure is shown in Fig. 7.

The DASDAQ software component provides the unified API for communication with the I/O device drivers for equipment control, setup etc. To unify the DASDAQ, the hardware abstraction implementation details were placed in the DAQDev library. Upon the appearance of new I/O modules, the DAQDev library is improved and extended.

The DBadapter component is used for the unification of the software interface for data storage. According to the selected mode, the DBadapter may send a flow of experimental data to a binary data file in a local hard drive, or to the network database server. At the present time, the XFile binary data format and the open PostgreSQL[2] DBMS are supported.

The DASlog component provides API for the event logging facility. The text messages generated using DASControl are stored in RAM, a local file, or sent to the remote server using the standard syslog protocol.

The DASFSM software component provides the general control over the subsystem using a configurable state machine. The transfer from one logical state to another is made according to commands from the upper hierarchical levels of the experiment automation system. The main logical states of the DASControl are shown in Fig. 6.

The text format configuration file with the experiment parameters is copied to the local storage of the subsystem before the beginning of the experiment. The format was developed with a simple intuitive structure in mind. No special software tools, except the text editor, are required to process it, but this format has powerful features for defining fixed and real number parameters, text strings, and tables. To simplify access to the configuration parameters from the control software, the DASCfg library API is used.

After the start of the DASControl application, it uses two threads of execution. The parent thread is provided for the DASFSM, DASLog, DBadapter and DASDAQ components, and the child thread is used for NetServer to provide communication with the upper level subsystems using T-ICS protocol. This separation was made to prevent software freezes when communicating with slow subsystems, e.g. subsystems with slow vendor-supplied hardware initialization functions. If the child thread gets a command to change the logical state of the subsystem, the NetServer component makes a request for a state change to the parent thread using shared variables. To prevent data access race conditions, mutexes are used. The usage of mutexes was considered a good technique because of their portability.

The unification of DASControl API makes possible an easy extension of the data acquisition and control system functionality by the integration of new devices into the data acquisition system of the KTM.

7. THE TICS PROTOCOL AND NETWORK COMMUNICATION

The TICS protocol was developed in-house to provide the unified communication API for different device classes of the KTM, even in real-time mode. It may use serial line or Ethernet as a transmission media. When using a serial line it uses an RS-232 or RS-485 interface and provides a single-master binary protocol to communicate with data acquisition and control devices. Over the Ethernet it provides communication over UDP/IP with a special retransmission algorithm. It uses the following types of frames with different structures: message, command, response, and request to

[1] Xenomai: Real-Time Framework for Linux. www.xenomai.org
[2] www.postgresql.org

write or to read the variable. The protocol provides access to up to 64-bit address data space for extended data exchange during the communication. The frame structure of the protocol was optimized for fast transmission of vital data such as commands and responses.

8. CONCLUSION

In summary, the infrastructure described fits the requirements of timing and event synchronization, accurate and reliable execution of the scenarios, and flexibility. The solution proposed allows preparation for and control of the KTM discharges. The specialized sequence description language may be used to describe the experiment sequence in a fast and clear way.

The developed network communication protocol allows the centralized monitoring and control of all the experiment automation subsystems at the main control console. Also, remote participation was made possible for the remote researchers equipped with the appropriate software.

Nowadays, the described control software for the experiment control is used at KTM tokamak in Kazakhstan.

REFERENCES

[1] Azizov E.A. et al. "Kazakhstan tokamak for material testing." *Plasma Dev. Oper*., vol. 11, no. 1, pp. 39-55 (2003).

[2] Baystrukov K.I., Pavlov V.M., Sharnin A.V., Obhodskij A.V., Merkulov S.V., Golobokov Y.N., Mezentsev A.A., Ovchinnikov A.V., Tazhibaeva I.L., "Control and data acquisition system of tokamak KTM". *Proc. of the 17th IAEA Technical Meeting on Research Using Small Fusion Devices 22-24 October 2007 Lisbon, Portugal*: American Institute of Physics, Melville, New York, 2008, vol 996, pp. 297-306.

[3] Ovchinnikov A.V., Pavlov V.M., Erusaev A.P., Sharnin A.V. and Golobokov Y.N.. "Software architecture of tokamak KTM data acquisition subsystems", *19th IAEA Technical Meeting on Research Using Small Fusion Devices 28-30 September 2009 Kurchatov, Kazakhstan*.

[4] Sharnin A.V., Baystrukov K.I., Rogozniy D.V.. "Design of data acquisition and control subsystem of the KTM tokamak multifrequency pulse radar reflectometer". *Plasma Dev. Oper*., vol. 11., no. 4., pp. 237-241 (2003).

[5] Obkhodskii A. V., Baistrukov K. I., Pavlov V. M., Merkulov S. V. and Golobokov Yu. N.. "A system for measuring the electromagnetic characteristics of the KTM tokamak". *Instruments and Experimental Techniques*, vol. 51, no. 6, pp. 799 – 804 (2008).

[6] Obkhodskii A.V., Dmitrienko A.A., Merkulov S.V. and Golobokov Y.N.. "Software structure of the KTM tokamak magnetic diagnostics data acquisition system". *19th IAEA Technical Meeting on Research Using Small Fusion Devices 28-30 September 2009 Kurchatov, Kazakhstan*.

[7] Ovchinnikov A.V., Erusaev A.P., Pavlov V.M., Golobokov Y.N., Mezentsev A.A., Tazhibaeva I.L. and Shapovalov G.V.. "Software structure of KTM tokamak data acquisition subsystems". [in Russian], *Devices and Systems: Control, Monitoring and Diagnostics*, no. 1, pp. 33-36 (2010).

[8] V.M. Pavlov, Y. N. Golobokov, A.V. Lysionok. "The Software for Runtime Control of the Experimental Physical Device". *Problems of Atomic Science and Technology, Ser. Thermonuclear Fusion*, 2011, issue 3, pp. 81-87.