RaceTrack Microtron Control System*

I.V. Gribov†, I.V. Shvedunov† and V.R. Yalijan†,
World Physics Technologies, Blacksburg VA 24060 USA

Abstract

We have developed a tri-level **P**ersonal **C**omputer-based **R**ace**T**rack **M**icrotron control system capable of processing 75 accelerator parameters. Custom smart front-end devices monitor 20 pulse parameters while we control other parameters with commercial multi-channel data acquisition cards. We use a **R**eal-**T**ime Linux operating system and fulfill both hard and soft real-time tasks in the first two levels. We use static fuzzy logic and dynamic direct digital methods in Level-2 to control parameters and assure long-term stability. In Level-3 we support a database of fully described accelerator parameters and operational rules; establish man-machine and world interfaces; provide script handling; and implement RTM startup, tuning, and monitoring models.

1 SYSTEM HARDWARE

Our Ethernet linked control system, seen in Fig. 1, has three network loaded Level-1 and -2 stations and a fully configured Level-3 station. Acquisition cards and digital signal processor modules from the controller area provide data to front-end PCs by a CAN-bus. We have built bipolar direct-current sources that feed magnetic RTM **D**irect **D**igital **C**ontrol monitoring elements, simple stepping motor controllers, and temperature regulators.

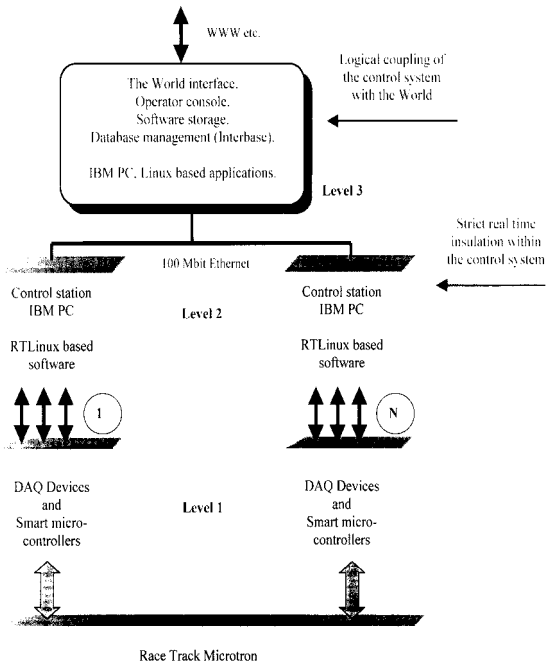

Figure 1: RTM control system.

2 LEVEL-1 AND -2 SOFTWARE

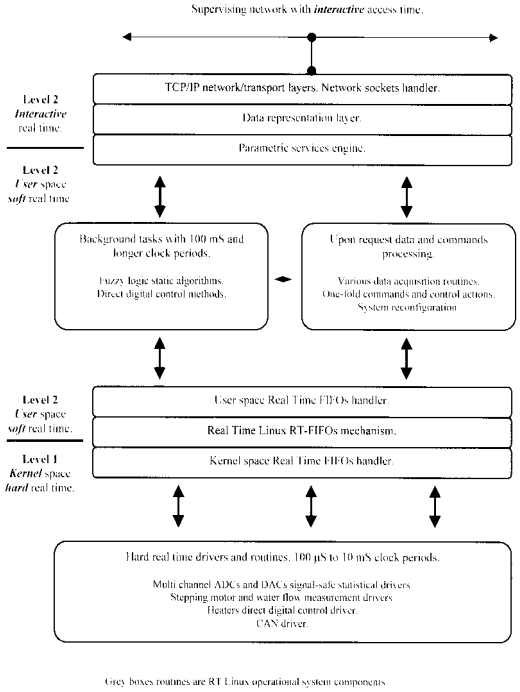

Figure 2: Level-1 and -2 software structure.

Our control system real-time software, seen in Fig. 2, has evolved [1] to include signal-safe device drivers, a signal based hard real-time structure, optimization fuzzy logic static control, proportional-integral DDC feedback, and a parametric accelerator description. Level-1 kernel space tasks are either hardware interrupt driven or activated by user space RT Linux first-in-first-out routines. Level-2 user space tasks are initiated either by a 100 ms timing signal for periodic routines or a network data-receive signal in which the control station acts as a parametric server. All control system operations involving several stations are initiated by the master/client station and use parameter manipulation with uniform command and data exchange. The Level-3 station is a client, while Level-2 stations are servers but can also operate as masters.

*Work supported in part by U.S. Department of Energy 50043-98-II.
†Permanent address: Nuclear Physics Institute, Moscow State University, 119899 Moscow, Russia.

2.1 Level-1 software

Level-1 contains hard RT Linux kernel space modules and drivers. Compact COM port drivers provide basic data exchange while a serial interface gives device access and can be used to debug networking modules. A command driver maintains low-level processor control from user space Linux modules (e.g., CPU disable and enable interrupts.)

A 16-channel DAC signal-safe driver, accessed when all CPU interrupts are inhibited, allows applications to use any device channel at any time. A 32-channel ADC signal-safe reentrant statistical driver disables "no" interrupts, allows applications similar device channel access, obtains 15-bit accuracy from a 16-bit ADC, and discards distorted RTM data. We assume normally distributed data and calculated averages so that data sampling rates are less than those of the ADC. This driver also operates in background mode so that 50 Hz alternating current interference is suppressed in sparse data.

A non-interfering half-period DDC driver cycled at 640 ms controls the 257 power levels of four heaters. Simple hardware synchronizes alternating current TTL pulses used as the driver clock signal that initiate PC parallel port interrupts. By halving these interrupts, on-off solid-state relays are uniformly distributed over the control cycle.

Stepping motor four-phase meander control is provided by DDC drivers clocked with 960 Hz COM port transmitter interrupts that also control the end switch status which stop the motors at their limit.

2.2 Level-2 software

Level-2 modules operate in user space Linux mode and are synchronized with high priority periodic and network data receive signals whose hierarchy guarantees good control routine execution at high network request volume.

Our data representation layer assembles ready-to-send parametric data frames for the transport layer. The parametric receive function checks the transport layer frame consistency, extracts parameters, and initiates the operation code defined services. The multi-data layer performs these same functions but for multimedia data.

The service engine processes parametric requests/commands, checking to see if the parameter exists in the station and if it can be processed by the requested operation. It then calls the corresponding data acquisition or control function and, if needed, returns the requested data. This module also informs the Level-3 station of parameter or station software faults.

Optimization fuzzy logic feedback loop modules control static parameters that are stable when their transition times are less than that of defined jitter. We use a one-dimensional *regula falsi* in optimization and determine the correct algorithm behavior when a parameter goes out of range indicating loss of control or some other irregularity. This method can maintain long-term parameter stability as well as one-fold settlement for measured parameters adjusted to order one.

Proportional-**I**ntegral DDC control modules stabilize the water temperature. Soft heater power switch-on is activated with the power level converted integral. After long operation at extreme power levels, a saturation property allows rapid resumption of linear control.

Master/client stations that initiate operations use the functions of the basic operation and transaction module.

3 PARAMETER MANIPULATION

Our parametric approach provides a full, transparent, and coherent description of the RTM from two types of data. Exchange commands, status, and ordinary formatted data use the standard data representation. Data of any type or size, which is to be interpreted and processed by a final application, use a multi-data representation.

Each data acquisition/control operation is defined by a parameter with properties, a value, and handling rules. This parametric description is supported in the Interbase relational database whose table structure includes four principal and several normalized auxiliary tables. The definition table describes the parameter type, format, and size as well as its operations and their rules. A second table contains ordered parameter values and the status for a reasonable number of RTM states. Rules for external control actions with the parameters are contained in a third table that mainly provides script support when multiple operations use a set of parameters. The fourth table logs all the requested or registered data values and their status during the RTM operation. A normalized name-description table contains text records of the RTM parameters, their values, and the handling rules. Journaling tables log all external control system operations while an administrative table records operator permissions and security information.

Control parameters are manipulated with three data handling transactions capable of simultaneously processing a number of parameters. A read request receives standard or multi-data from a server station. A write transaction writes parameters to a slave station. A set transaction writes parameters and waits for them to be processed by a slave station.

Parameters are manipulated with six basic operations. The read transaction serves status and value requests. The write or set transaction orders a parameter or predefined operation. A read transaction retrieves any orderable parameter. A special routine handles any active status issued by the control stations. These basic operations with script manipulation provide all the necessary tools for our powerful and reliable parametric control.

4 LEVEL-3 SOFTWARE

The Level-3 master/client station performs four main tasks. The control system is accessed by world and man-machine interfaces, two of which are supported. The first gives control system designers database and script capabilities with which to access all parameters and operations. The second, used by RTM operators, has extensive mouse manipulatable graphics. Both X-windows based interfaces are written in C++ and use the QT graphics library.

SQL requests maintain and manage the relational Interbase database which handles basic table and record manipulations using macro operations to ensure a coherent RTM state (parameter values and status) and the corresponding database information. Thus, when separate records are edited, normalized tables are correctly triggered. Some RTM state or external script rules can be used to prototype another state or script. New data are forwarded to Level-2 when an operator/routine changes it in the database.

The control system supports both built-in scripts, which work directly with parameters guided only by inherent rules, and external scripts, which read their rules from the database. Built-in scripts check parameter description consistency in the database and in the Level-2 stations. With scan script any separately controlled parameter can be a linearly scanned argument, during which other values are measured and presented. Save script can retrieve and store the RTM state in the database. A dedicated built-in script that examines the database table, listing external operation control rules, is used to maintain external scripts. The script reads the parameters in a defined sequence - their values, operation codes, and methods - and executes necessary actions.

The Level-3 software also maintains the logical RTM representation so that external tasks can operate with the machine exactly as they do with object simulation functions. The only change is that the task program must replace a function call with a set transaction, thus processing the parameters equivalent to its arguments. This is then followed with a read in which parameters are requested equivalent to the function values. A set call guarantees that all parameters have their new values so the task algorithmic components do not know if they are operating with a real object or its simulation.

5 REFERENCE

[1] A.S. Chepurnov, I.V. Gribov, S.Yu. Morozov, A.V. Shumakov, and S.V. Zinoviev, “Moscow University Race-track Microtron Control System: Ideas and Development,” in Proc. Inter. Conf. Accelerators and Large Experimental Physics Control Systems, C.O. Pak, S. Kurokawa, and T. Katoh, eds. (Tsukuba, Japan, 1991) p. 140.