HIGH-SPEED DATA HANDLING USING REFLECTIVE MEMORY THREAD FOR TOKAMAK PLASMA CONTROL

S.Y. Park, S.H. Hahn, and W.C. Kim, NFRI 113 Gwahangno,
Yuseong-gu, Daejeon, 305-333, Korea
B.G. Penaflor, R.D. Johnson, D.A. Piglowski, M.L. Walker,
General Atomics, P.O. Box 85608, San Diego, CA 92186-5608, USA

Abstract

The Korea Superconducting Tokamak Advanced Research (KSTAR)[1] plasma control system (PCS) is defined as a system consisting of electronic devices and control software, which identifies and diagnoses various plasma parameters and calculates appropriate control signals to each actuator to keep the plasma sustained in the KSTAR operation regime. The KSTAR PCS consists of a linux system with 8 processors and both analog and digital data acquisition methods are adapted for fast real-time data acquisition up to 20 kHz. The digital interface uses a reflective memory (RFM) technology to share data among various subsystems of KSTAR. RFM technology has been adopted as the real time communication method to enable PCS to interface with the actuators and to do interprocessor communications inside the cluster. To handle the fast control of the RFM data transfer, the communication using the RFM with the actuators and diagnostics system is implemented as a thread which is assigned to a separate process.

INTRODUCTION TO THE PLASMA CONTROL SYSTEM

The plasma control system is composed of real-time computers for feedback calculations, a diagnostic system for plasma information, and a communications interface with actuators. The PCS acquires plasma data from the diagnostic system and performs a feedback control loop to obtain plasma properties. Figure 1 shows the plasma control system structure. The PCS feedback algorithm calculates the difference between target and measured values, and decides how much coil current is needed in order to reduce the error from the target. Next the PCS sends coil voltages to the magnet power supply (MPS) for the desired poloidal magnetic field (PF) and receives coil current measurements from the MPS. In KSTAR this feedback operation is performed over an optical network consisting of reflective memories [2] which are high-speed replicated shared memories with up to 256 nodes featuring very low latency and wide throughput. Although the performance of the initial system [3] with a single process was acceptable, performance demand for a shaped plasma required faster control cycles up to 50 us, as well as increased interprocess communications for sophisticated magnetic controls. In 2011, the amount of I/O data exchanged in a single cycle was about 1 kB, hence the old PCI-based method was not suitable for the requirement because of 3 us time overhead for each access of the RFM space.

A method utilizing the dedicated thread for RFM is introduced for the following purpose: 1) to minimize

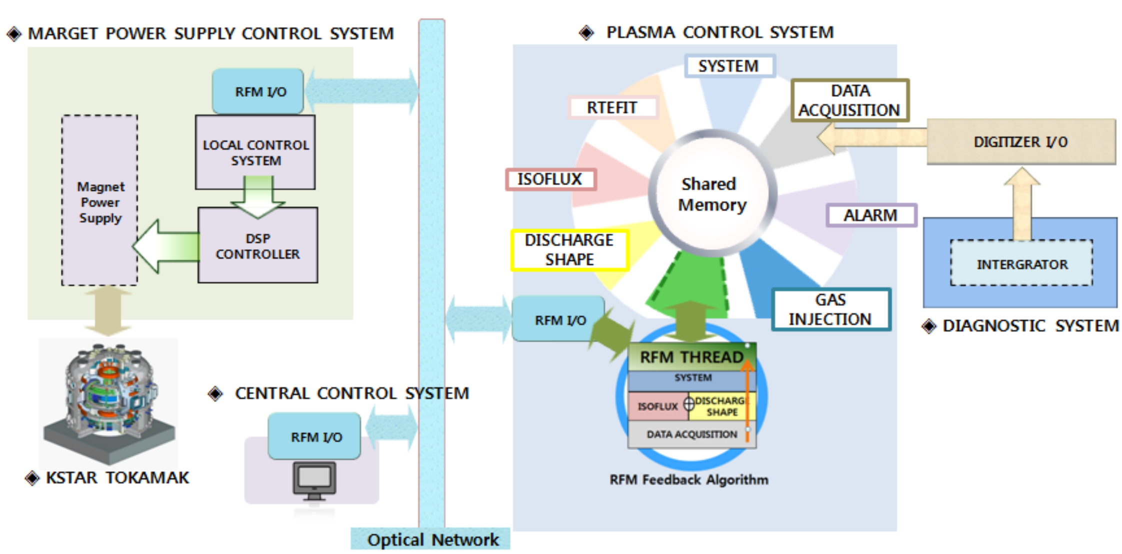

Figure 1: The structure of the plasma control system.

intrinsic time overheads for the RFM access, 2) to transfer all the RFM data within 50 us, and 3) to gather all the RFM data generated in other 3 cpu's running in the PCS via shared spaces. In this paper, we describe the principles of the RFM thread and the feedback algorithm for the PF coil control.

OPERATION OF THE RFM THREAD

The structure of the RFM thread is shown in Fig. 2. The RFM thread is spawned as a child process of one of real time processes and assigned to a separate physical processor at the start of each shot and killed at the end of the shot. The main operation of the RFM thread is to copy the RFM data to a preassigned shared memory area at every cycle to use in the feedback algorithm. In the KSTAR PCS, this shared memory is shared by all the real-time processes and is referred to as the real-time heap memory (rtheap) [4]. The rtheap is a shared memory that contains various constants and pointers to all of the other structures in the memory of the real time process. The RFM thread synchronizes itself with the real time control cycle by accessing the rtheap time counter delivered from the digitizer.

When the RFM thread starts, it calls the initialization function that creates the RFM handle, and allocates the internal memory used for communication with the RFM. If the initialization routine is successful, this routine returns the virtual address of the DMA memory address. After performing the initialization routine the RFM thread executes a continuous loop. At first the RFM thread reads values from a fixed memory area which is assigned for real-time measurement values for PF coil currents and voltages, and flags indicating the software limit defined by the central control system (CCS) or the hardware limit defined by the MPS. These measurement values are moved to an "internal buffer" memory area (see Fig. 2) and are saved in the shared memory in the same format as the acquired DMA data from the digitizers. The feedback control algorithm gets the data from the shared memory after one cycle. After reading the data, the thread waits for the current time to change by monitoring the fixed area of the rtheap. For the time synchronizations of the main and the RFM thread, a counter is used for the "new time". This counter synchronizes the internal CPU clock count of the cpu1 real-time process to the clock counter provided by the external clock of the digitizers [5]. The RFM loop waits for the counter change in order to catch this new time when the next cycle starts to determine the feedback command.

The process of determining feedback command of the next cycle is performed by the PCS feedback algorithm in the spawning process. As shown in Fig. 2, when deciding the PF coil command of the current cycle, the measured coil current of the preceding cycle is used; this coil current is the value saved in the shared memory area before the cycle. The reason for using measured value at the preceding cycle for obtaining the command of the current cycle is to reduce the feedback error occurred at the preceding cycle. The PCS feedback algorithm acquires the error value of the previous cycle by subtracting measured current in the previous cycle from the target current, and calls the PID function for adding compensation voltage at the next cycle. The answer from the PID function is sent to the RFM thread as the next PCS command. The RFM thread gathers all the commands for each MPS and writes the command to the internal buffer in order to do a DMA transfer to the fixed RFM area assigned for the MPS.

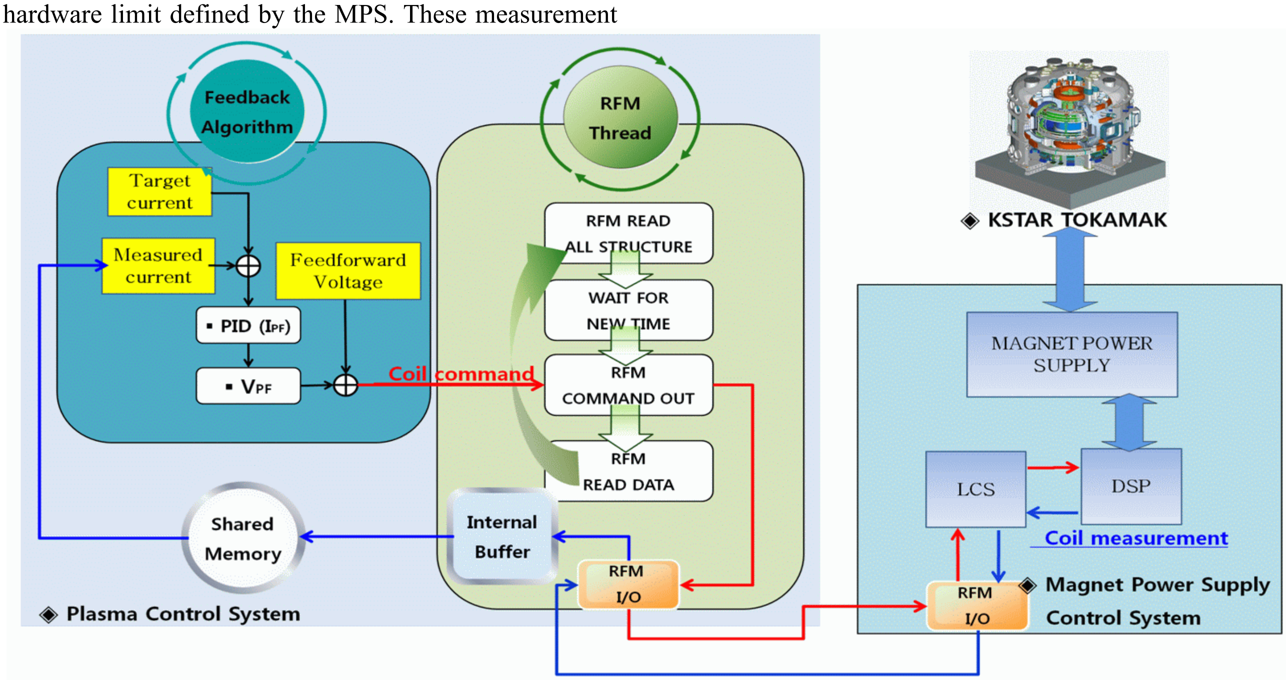

Figure 2: The RFM thread and PCS feedback algorithm.

Timeout Handling of the RFM Thread

The RFM thread has a few event handling algorithms in order to monitor its execution cycle, sync with the main thread and to prevent itself from endangering actuators under control of the PCS. The self-monitoring of the RFM loop is done by a time-out count. If the current time has not changed during the time-out period or the real-time process exits from the real time mode for any reason, the RFM thread fills zeroes in the PCS command structure which it sends to the MPS. Since the MPS (and its DSP controller) accepts PCS commands only during real-time mode, this will cause the MPS to exit out of PCS control and have the DSP controller take over control of the PF power supply. Hence the PF power supply can avoid a dangerous situation when the PCS is out of control by its own internal delay. This time-out period is set to 5 ms, which corresponds to the effective response time of a single MPS voltage command [3]. During this time-out period, the RFM thread also checks whether the main process is not updating the new time counter. If it is not, the RFM thread informs this situation to the central control system by updating a fault code so that it can spread termination signals to the other systems to abort the discharge.

Another example is a “software watchdog”. This is a counter which is increased by 1 on each control cycle. Since the counter is shared as a “timestamp” in the RFM area, this counter can be monitored by any device under the same RFM network as PCS. As a kind of heartbeat, the watchdog counter is monitored by each MPS and the CCS so that it can check communication shutdown on the RFM. If the value exceeds the limit, this value sets to zero again.

When the real time mode is ON by the PCS, the CCS monitors the software watchdog counter and checks the value every 10 ms. The CCS makes a fault code to the other system, if this value is not changed by 10 ms -- which could imply that the PCS has either a communication fault or a serious internal fault such as power down or real-time process hang-up.

Some Issues about the RFM Thread

- The time counter used in the RFM operation is acquired from the digitizer. Hence the clock source of the digitizer should be accurate. If the clock source does not have good accuracy, the RFM thread will be terminated because of a timing mismatch problem.
- Although the data processing speed of the RFM thread is fast, if the MPS interface system does not operate as we expected, there is a possibility that the RFM thread receives the same data during a couple of cycles. This could cause a saturation command issue in the P loop; however, in a practical manner, the design of the algorithm considering the delays by the slower update of the measurement can avoid the saturation issue.
- When writing to the PCS command to the RFM area, the RFM thread assumes that the values in the memory structure are the most recent written by the real time process. The RFM does not keep the previous value before it is updated by the new data. Due to this property, the RFM data written by two different devices is not exactly synchronized in time. Nevertheless, the assumption is true in most cases and the data is acceptable as the most recent one if the read cycle is faster than each writing cycle by the actuators, which fits to our case.
- If the RFM thread doesn’t synchronize with the real time process because of some problem such as accessing wrong memory area or time delay for writing to memory, the thread could miss cycles. Several missing cycles are reported when the RFM initialization function is called. This is due to the time overhead of the initializing hardware handle. In order to avoid timeout errors, the real time mode flag is updated as ENABLED after those missing cycles are gone, so that the PCS can get synchronized data from the MPS and the MPS can receive correct feedbacks.

The Command Structure Sent to the MPS

The entire PCS command structure is read by the local control system (LCS) of each MPS and sent to the corresponding DSP, which actually communicates with the power supply (see Fig. 2). The DSP reads each field of this command structure and decides the way of the coil control. Table 1 shows the PCS command structure sent to the MPS.

- “control method” indicates how the PF command should be interpreted (voltage or current)
- “current direction” indicates the charging direction of the PF coil. (forward or reversed)
- “timestamp” is used as a software watchdog in each control cycle (increased by one on each cycle)
- “real time mode “ tells the MPS when the PCS starts the real-time feedback cycle for them.

The Data Structure of the MPS and CCS

The amount of data that is received from a single power supply is 52 bytes, but the total amount of data is about 1 kB since there are several power supplies that should be controlled by the PCS. Table 2 shows the data structure of the MPS. The RFM thread receives all of the data that is sent by each power supply at once using DMA transfer. This is possible since the memory area corresponding to each power supply is arranged sequentially.

Table 3 shows the structure used for the CCS. There is a field that indicates current time of the PCS. This field is just for reference; the CCS operates in real time mode and has the PCS fault code which is set by PCS when unusual situation is occurred. When the CCS detects the PCS fault, the CCS informs this fault code to the other systems.

Table 1: Command Structure Sent to MPS

Parameter	Description
Id	Magic id
Control method	Indicates command type of the PF coil 0: current command 1: voltage command
Current direction	Initial charging current direction 0: forward 1: reversed
Time stamp	RFM watch dog counter
rt_mode	Indicates the PCS is in real-time mode
PF command	PCS command to send to MPS
Test variable	Dummy field for test
Current trajectory	Current trajectory when voltage is selected

Table 2: Data Measurement Structure from the MPS

Parameter	Description
Id	Magic id
Voltage	total voltage form MPS
Current	total current form MPS
Time stamp	RFM watch dog counter
D-axis current	D converter output current
Y-axis current	Y converter output current
Flag	over current flag
D-axis alpha	D converter alpha degree
Y-axis alpha	Y converter alpha degree
QP voltage	Quench protector voltage
Bris voltage	Bris voltage
D-axis voltage	D converter output voltage
Y-axis voltage	Y converter output voltage

Table 3: Data Structure for the CCS

Parameter	Description
Id	Magic id
PCS current time	PCS current time
PCS fault code	Indicates what kind of fault is occurred in the PCS
Force PCS abort	Central controller uses this value to abort the PCS

CONCLUSION

We were able to transfer all RFM data within 50 us, and control the PF coil efficiently using the RFM thread. We could increase the RFM memory read/write performance by rearranging the RFM memory sequentially and using the DMA transfer. We could identify that we can manage the RFM memory efficiency by using the RFM thread.

There are some sensitive issues. We have a plan to increase the amount of data through the RFM in the future; hence, we need to increase the RFM memory performance and also upgrade the interface devices for improved communication with the PCS in the RFM network and also for better coil control.

Currently we plan to improve performance by upgrading the RFM memory card to the PCI express bus format and to improve the algorithms to safely handle other possible unintended situations.

REFERENCES

[1] G.S. Lee, J.Y. Kim, S. Hwang, C.S. Chang, H.-Y. Chang, M.H. Cho, et al, Nuclear Fusion, 40 (2000) 575.

[2] http://www.ge-ip.com/

[3] S.-hee Hahn, M.L. Walker, K. Kim, H.S. Ahn, B.G. Penaflor, D. a Piglowski, et al., Fusion Engineering and Design, 84 (2009) 867-874.

[4] B.G Penaflor, J.R Ferron, M.L Walker et al, SOFT, 16-20 Sep 1996, Lisbon (Portugal)

[5] S. H. Seo et al, The 6th IAEA Technical Meeting Control, data Acquisition, and Remote Participation or Fusion Research 4-8, June, 2007, Inuyama, Japan