MDSPLUS REAL-TIME DATA ACCESS IN RTAI

A. Barbalace [1], G. Manduchi [1], A. Luchetta [1], C. Taliercio [1], T. Fredian [2], J. Stillerman [2]

1) Consorzio RFX, Euratom-ENEA Association, corso Stati Uniti 4, 35127 Padova, Italy
2) Massachusetts Institute of technology, 175 Albany Street, Cambridge, MA 02139

Abstract - **The MDSplus package is widely used in nuclear Fusion research for data acquisition and management. Recent extensions of the system provide useful features for real-time applications, such as the possibility of locking selected data items in memory and real-time notification. The real-time extensions of MDSplus have been implemented as a set of C++ classes and can be easily ported to any target architecture by developing a few adapter classes. The real-time data access layer of MDSplus is currently available for Windows, Linux, VxWorks and RTAI. In particular, the RTAI platform is very promising in this context because it allows the co-existence of non-real-time and real-time tasks. It is hence possible to devise an architecture where real-time functionality is handled by a few selected tasks using the real-time data access layer of MDSplus, whereas background, non-real-time activity is carried out by "traditional" Linux tasks. This organization may be of interest for the next generation of fusion devices with long-duration discharges, during which the system has to provide feedback control in real time and to sustain continuous data acquisition and storage.**

INTRODUCTION

MDSplus :

✓ has been successfully adopted in many fusion experiments for data acquisition, storage and access [1];
✓ was not fit for the new generation of fusion experiments in which the plasma discharge may last minutes or hours;
✓ the most recent release allows data samples to be appended to stored signals in the database [2];
✓cannot guarantee the real-time responsiveness required in active plasma control.

For this reason, work is in progress for introducing real-time functionality in MDSplus [3]. Feedback control is in fact becoming a standard procedure in fusion experiments.

ACHIEVING MULTIPLATFORM SUPPORT

✓ The real-time layer of MDSplus is written in C++, and in this case multiplatform support has been achieved by encapsulating system–dependent code into a subset of C++ classes.
✓ Porting the system to a new platform requires re-implementing only those classes.
✓ Real-time communication, required to maintain consistency in distributed data caches, relies on the implementation of a set of generic classes for communication.
✓ UDP-based implementation over Ethernet is available, but other communication media can be integrated, such as ATM and reflective memories.
✓ The real-time layer of MDSplus has been ported to Linux, Windows and VxWorks.
  ✓Among these, only VxWorks provides real-time responsiveness
  ✓ However, quasi real-time performance can be obtained using the Linux kernel 2.6.

For this reason, the performance of the real-time MDSplus layer can be satisfactory for those systems for which an occasional delay in the response time is tolerable.

SYSTEM ARCHITECTURE

The real-time layer of MDSplus provides :
✓ data caching in memory;
✓ real-time notification of data update.
They represent the basic ingredients required to build feedback control systems.

Data caching

✓ The data cache component does not replace any existing component of MDSplus.
✓ It is built on top of the MDSplus data access layer.
✓ It allows selecting a subset of data items for which a copy is maintained by the system in memory in order to provide deterministic access time.

Real-time notification

✓ Data update notification is useful in feedback control systems where the system has to react to the occurrence of a new event, such as the availability of a new input sample from the sensors.
✓ It is achieved using a publish-subscribe pattern centered on data.
  ✓An actor can express its interest in being notified when a given data item is updated, by passing the address of a callback routine.
  ✓Afterwards, the system will call such routine in a separate thread each time that data item is updated.

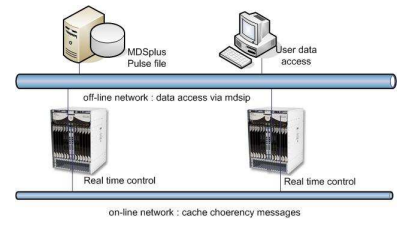
Distributed systems

✓ Both memory caching and notification are supported also in a distributed environment.
✓ Distributed data caching requires exchanging information over the network in order to maintain data consistency.
✓ MDSplus defines two different approaches for handling cache coherence: ***push mode*** and ***pull mode***.
  ✓ In ***pull mode***, when a data item is updated in one cache, all the other data caches holding the same data item are notified that this cache has become the current owner of that data item.
  ✓ In ***push mode***, the current owner sends an updated version of the data item to all the caches sharing it, which hold therefore an up-to-date version of the data item.
  ✓ The push mechanism is used also to achieve remote notification, i.e. activating a callback routine in response to updating that data item in a different machine.
  ✓ Depending on the way data are accessed, either push or pull mechanisms may minimize the number of exchanged messages.

INTEGRATION IN RTAI

✓ In order to port MDSPlus to a free, open source real-time operating system, we have considered RTAI, a real-time extension of the Linux kernel [4].
✓ There is a substantial difference in software organization between VxWorks and RTAI and, more in general, the real-time extensions of Linux.
  ✓ In VxWorks all code runs natively in privileged mode, thus letting user program have full control of the hardware resources
  ✓ In this case the porting of the MDSplus, and in particular of the real-time data access layer, is straightforward, requiring only slight changes in the system-specific classes.
  ✓ On Linux, instead, task can run in either user or kernel mode
✓ Moving to kernel space, however, would have required a deep re-arrangement of the code to provide the required interface to the GLIBC support.
✓ This can be avoided in RTAI because user processes can become real-time processes.
  ✓ In this case the only changes required to take advantage of the real-time capability of RTAI, with respect to the Linux version, is the usage of a RTAI-specific real-time semaphore in the data update notification.
✓ When considering distributed systems involving real-time communication to achieve data cache coherence, a promising approach is the usage of RTNet [5],
  ✓RTNet is an open source hard real-time network protocol stack for RTAI which makes use of standard Ethernet hardware.
  ✓ The integration of RTNet into the framework requires an implementation of the system specific classes for communication, and is currently under development.

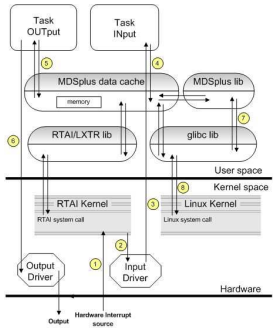
The availability of real-time user processes represents a good tradeoff between performance and software maintenance, and the real-time extension of MDSplus over RTAI represents a valid candidate for the development of real-time control systems in the next generation of long lasting fusion experiments.



The figure shows a typical configuration in distributed control. Two networks are involved: an off-line network (upper network in figure) is used by the underlying MDSplus system to read data from the pulse files, hosted in a separate data server.
During real-time operation, most of the data access is performed on the local cache. To maintain the consistency of data items shared by two or more computers, the communication is achieved via messages exchanged along the online network (lower network in figure), usually an isolated Ethernet segment. The off-line network may also be concurrently used by low priority, non real-time tasks for data logging.

REFERENCES

[1] T. Fredian , J. Stillerman "MDSplus Current developments and future directions, Fusion Engineering and Design", vol 60, 2002, pp 229-233
[2] T. Fredian, J. Stillerman, G. Manduchi, "MDSplus extensions for long pulse experiments", to appear in Fusion Engineering and Design.
[3] G.Manduchi, A. Luchetta, C. Taliercio, T. Fredian, J. Stillerman "Real Time Data Access Layer for MDSplus", to appear in Fusion Engineering and Design.
[4] RTAI Home page, [Online]. http://www.rtai.org
[5] RTNet Home Page [Online] http://www.rts.uni-hannover.de/rtnet



Control flow in an application involving two RTAI real time tasks
1) An interrupt is generated by the ADC when a new sample is ready;
2) The RTAI interrupt dispatcher calls the input driver ISR;
3) The ISR copies the sample into a buffer shared with the input task and awakes it via a RTAI semaphore;
4) The INput task is a real time user process : it performs the required computation on the input sample an writes it into the MDSplus data cache;
5) The real time MDSplus layer notifies the OUTput task, which previously registered as a listener for that data item. Notification is done via a RTAI semaphore;
6) The output task is a real time user process: it reads the data item from the data cache possibly performing further computation and sends the output sample to the output driver, e.g. via a mmap operation;
7,8) In parallel, the data items is written to disk by a MDSplus non real time task using standard Linux I/O.

The sequence from 1 to 6 corresponds to a control cycle and occurs without the intervention of the Linux scheduler, being it entirely handled by the RTAI interrupt dispatcher and the RTAI/LXRT Scheduler.

2007 International Conference on Accelerator and Large Experimental Physics Control Systems October 15 – 19 2007
Oak Ridge National Laboratory P.O. Box 2008 Oak Ridge, TN, 37831-6471 USA