



International Journal of Engineering Sciences & Management Research

PROTOTYPE SOFTWARE FOR NETWORK BASED DAC OPTIMIZATION

Ramesh Joshi^{*1}, Manoj Singh²

^{*1} PG Student, Information Technology, Shantilal Shah Engineering College, Bhavnagar, India

² Engineer-SE, ICRH-RF Division, Institute for Plasma Research, Gandhinagar, India

*Correspondence Author: arjoshi07@gmail.com

Keywords VME - VersaModule Eurocard, ICRH - Ion Cyclotron Resonance Heating, DAC - Data Acquisition and Control system, EPICS - Experimental Physics and Industrial Control System.

ABSTRACT

EPICS based Ion Cyclotron Resonance Heating (ICRH) Data Acquisition Control system (DAC) client prototype is commissioned for remote operation of heating experiment. ICRH-DAC is physically distributed into two sections at RF Lab and SST-1 hall. RF Generation section at RF Lab and Transmission line, interface and antenna section at the SST-1 hall having its own independent Versa Module Euro card (VME) based DAC server. Linux client software of both DACs have been connected with Ethernet. Master DAC client acts as Experimental Physics and Industrial Control System (EPICS) Input/ Output Controller (IOC) server which is used for Slave DAC client communication for parameter exchange during experiment. One of such clients is Best OPI Yet (BOY), which is part of the Control System Studio (CSS) package. It provides an editor and a runtime for developing operator panel interfaces. This paper describes the prototype software with test results of the user-perceived performance analysis. We give an interpretation of the performance data, and further analyze them with respect to the memory subsystem, CPU speed and the specific implementation of networking protocols. Thorough analysis enables us to identify the performance bottlenecks and approaches to improve the performance. Software has been developed with EPICS Input-Output Controller (IOC) in a way to better accommodate these Channel access layer provided by EPICS as well as the operator panels designed in CSS BOY.

I. INTRODUCTION

Data acquisition and Control System (DAC) has been commissioned for Ion Cyclotron Resonance Heating (ICRH) experiment. ICRH is a promising heating method for a fusion device due to its localized power deposition profile, a direct ion heating at high density, and established technology for high power handling at low cost. 1.5 MW of RF power is to be delivered to the plasma for pulse lengths of up to 1000 second [1]. This operation remotely controlled by Master DAC system and the other part means generated power transmission with matching network and antenna diagnostics would be controlled and monitored by slave DAC system. There is hardware as well as software interlock have been implemented and tested before experiment and maintained periodically [2,3]. The master DAC has been installed at RF Lab to control and monitor the RF generator system, which are provides RF power at 22 - 25 MHz, 45.6 MHz and 91.2 MHz frequencies. All the signals coming from different stages have been connected through front end electronics and signal conditioning which ends at VME terminal. The application allows control of a variety of hardware, ranging from straightforward, continuous data streaming of a few channels to multi-rack based data acquisition instruments. Slave DAC system is responsible to control and monitor RF transmission and diagnostics using two transmission lines with offline and online matching system.

DAC communicated with Ethernet are using TCP/IP socket. As requires by the system parameter will demand for communication data to other network connected system. One system has to establish control connection for exchange data or event from another network connected system. Control connection has been taken place then the data connection would be done using TCP or UDP. Data have been exchanged between connected systems. Finally the desired experimental requirement has been accomplished and connection has been closed or demand for another event has been occurred.

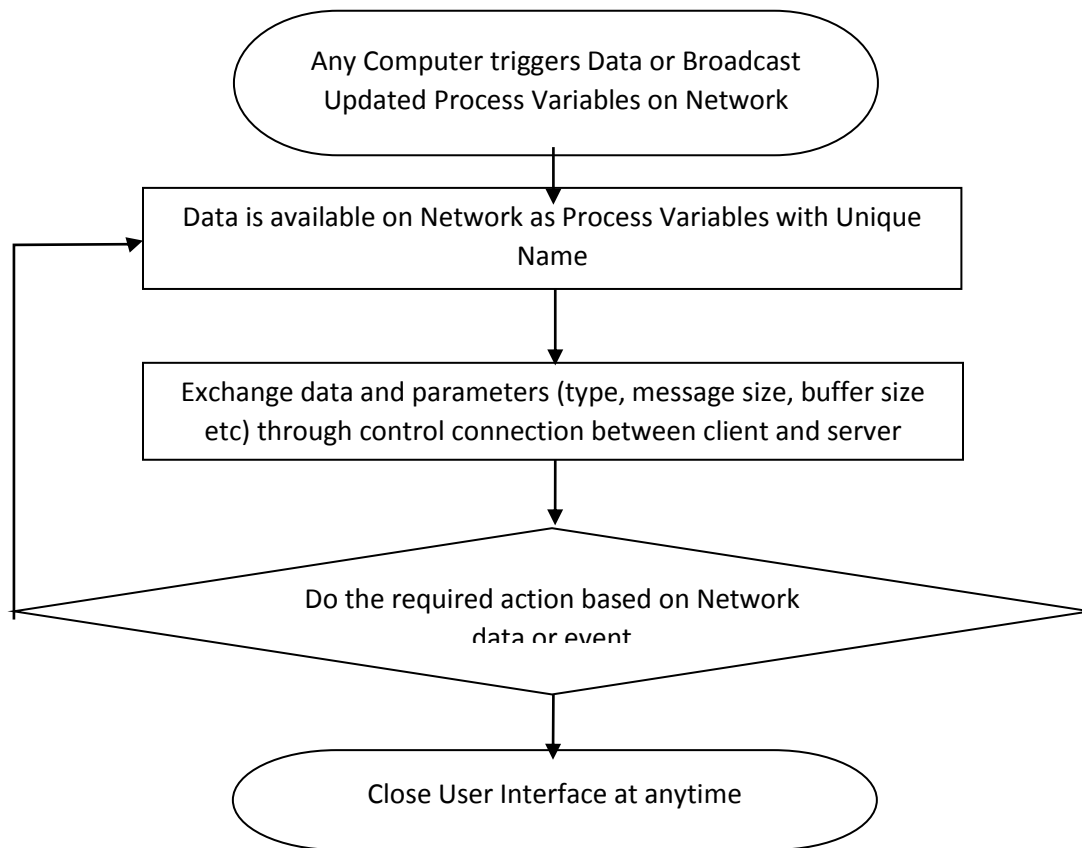


Figure.1: Communication diagram



II. PROPOSED SOLUTION

Several optimizations have been suggested to deal with these problems. The first one is to reduce the number of passes that need to be made over the data for every send and receive socket calls. One suggestion is to replace the cumbersome memory buffers with large contiguous buffers. The memory was an expensive resource that needed to be carefully allocated. Today, with fast CPUs and inexpensive memory, the focus should be shifted to higher performance and simplicity of code. Figure.1 shows the communication diagram for each connection subsystems. The TCP/IP packet overhead and physical communication media bound communication performance. The network-connected systems have been frequently needed for the data and event. These systems require deterministic output with experimental limitations. With the reviews of different papers, it is derived that to use UDP for communication mechanism and TCP for data exchange between different LAN connected systems. EPICS provides channel access layer which provides this paradigm with broadcasting mechanism.

III. EPICS IMPLEMENTATION

The EPICS software [4] was originally designed to be tool based approach to process control and this continues to be its primary application. An infrastructure that encourages proper design of distributed software systems is also important. For example, in multi-threaded distributed systems, toolkit needs communication software interfaces designed to avoid application programmer introduced mutual exclusion deadlocks. The EPICS based software tools includes application software that offers a satisfying solution for measuring, processing tasks and secondary development software which is a powerful technology that allows software integrators and application uses to customize and automate the application software. Several extensions provided with EPICS could also be used for alarm handler, interlocking and alarming mechanism implementation [5]. We could also able to use the CSS Best Opi Yet (BOY) for the user interface development which provides the XML based markup language integration with bundled widget options and integration based on Eclipse platform. Certain aspects of the existing EPICS communication software interfaces appear to be important facilitators for advanced toolkits. Software interfacing with systems capable of independent actions needs interfaces that can generate an asynchronous response synchronized with external events. Figure.2 shows integrated architecture for network connected systems.

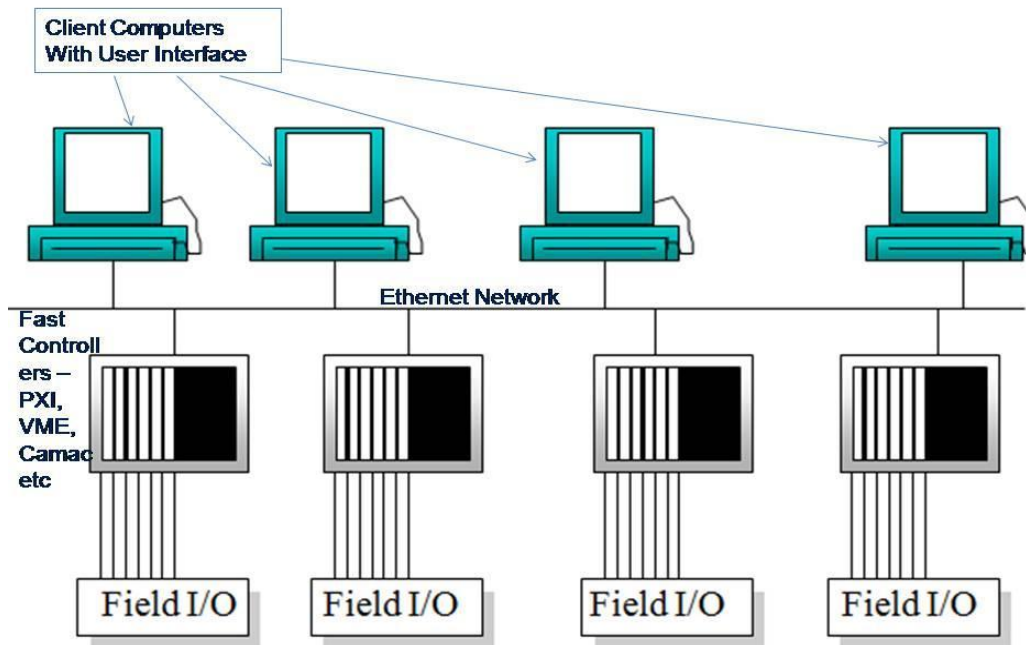


Figure.2: Integrated architecture of Connected DAC systems

IV. PROTOTYPE APPLICATION

CSS BOY is an Operator Interface (OPI) development and runtime environment. An OPI is a general GUI but with extra facilities to connect to your live data directly. CSS BOY allows building your GUI with drag and drop and connecting to your data instantly [6]. It also allows using JavaScript or Jython to manipulate the GUI in a very similar way as using JavaScript in HTML. In BOY, the OPI Editor is a WYSIWYG (What You See Is What You Get) editor which allows you to create your GUI in a similar way of creating PPT. The OPI Runtime works in a similar way as modern web browsers. One can display the OPIs either in tabs, windows or views and navigate OPIs forward or backward. An OPI is a regular XML file that can be edited in OPI editor or text editor and run in OPI Runtime. No compilation is needed. Figure.3 shows the user interface development for DAC software 2 kW and 20 kW stage. Same way in runtime the experimental shot panel has been shown in figure.4. One has to feed the required parameter and give shot in synchronous with other network-connected subsystems. The data communication layer is a separate layer, which allows BOY connecting to various data sources seamlessly. Users can provide their own data source by extending an Eclipse extension point. Figure.5 shows the terminal screen for broadcast of the process variables using EPICS module. EPICIS provides command softIOC that is used for broadcasting.

To make user interface the state notation language (XML) has been used with CSS IDE and assign required field widgets with respective process variables. The signal naming has been specified at the ICRH:<signal_name>. Using softIOC module we have broadcasted the process variables (PVs). XYgraph has been chosen for monitoring the voltage and current signals. Python script has been used for the periodic assignment of the channels process variables using caput command for apply periodic new value to the respective process variable. Separate python script is running periodically using execute command function provided on action button click event. In this script we have used pypicps [7,8] package and import epics as python module and will able to process caget and caput command as per requirements [9,10]. DIII-D has used open source solution for reliable and failsafe solution for the experimental requirement [11].

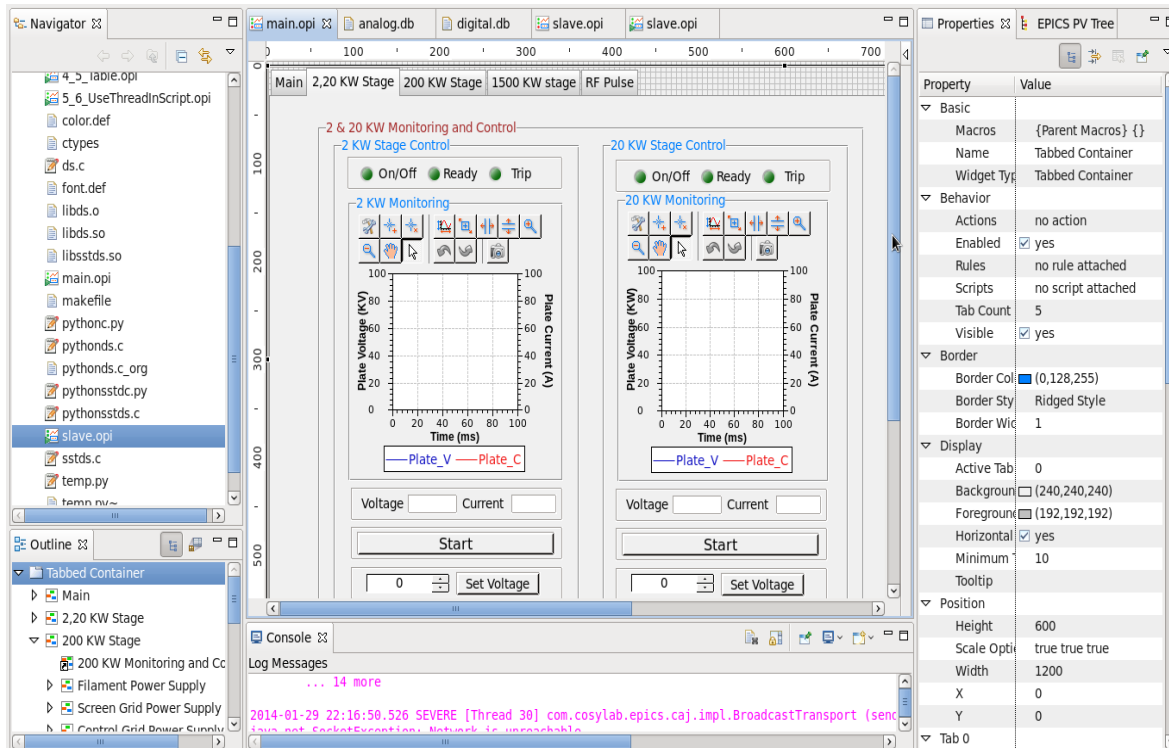


Figure. 3: CSS OPI user interface screen for DAC software

The fast fiber optic trigger network will give trigger pulse to fast controller at Master DAC. This fast controller get triggered that will trigger the fast controller at RF transmission DAC fast controller. As fast controller triggered the digitizer card buffer memory will get filled with the given on-time reference time. This data will be acquired by the Linux terminal user interface program with acquire button by socket command using Ethernet. Master DAC will be synchronized by Network Time Protocol (NTP) from Master GPS timer. Master DAC will communicate with slave DAC system with EPICS process variables. Data acquired at master DAC have been

sent to the slave DAC and that will acquire data accordingly. Instead of using the original EPICS IOC [12, 13], we decided to develop our own EPICS software toolkit, custom implementation, which is capable of building EPICS Channel Access (CA) server and client programs. In J-TEXT, EPICS provides improve productivity with channel access [14]. SPIDER tokamak has also support same kind of implementation with performance [15]. The most results has been matched with ANL Lab, USA [16] and ESS Bilbao, Spain [17] for network based systems.

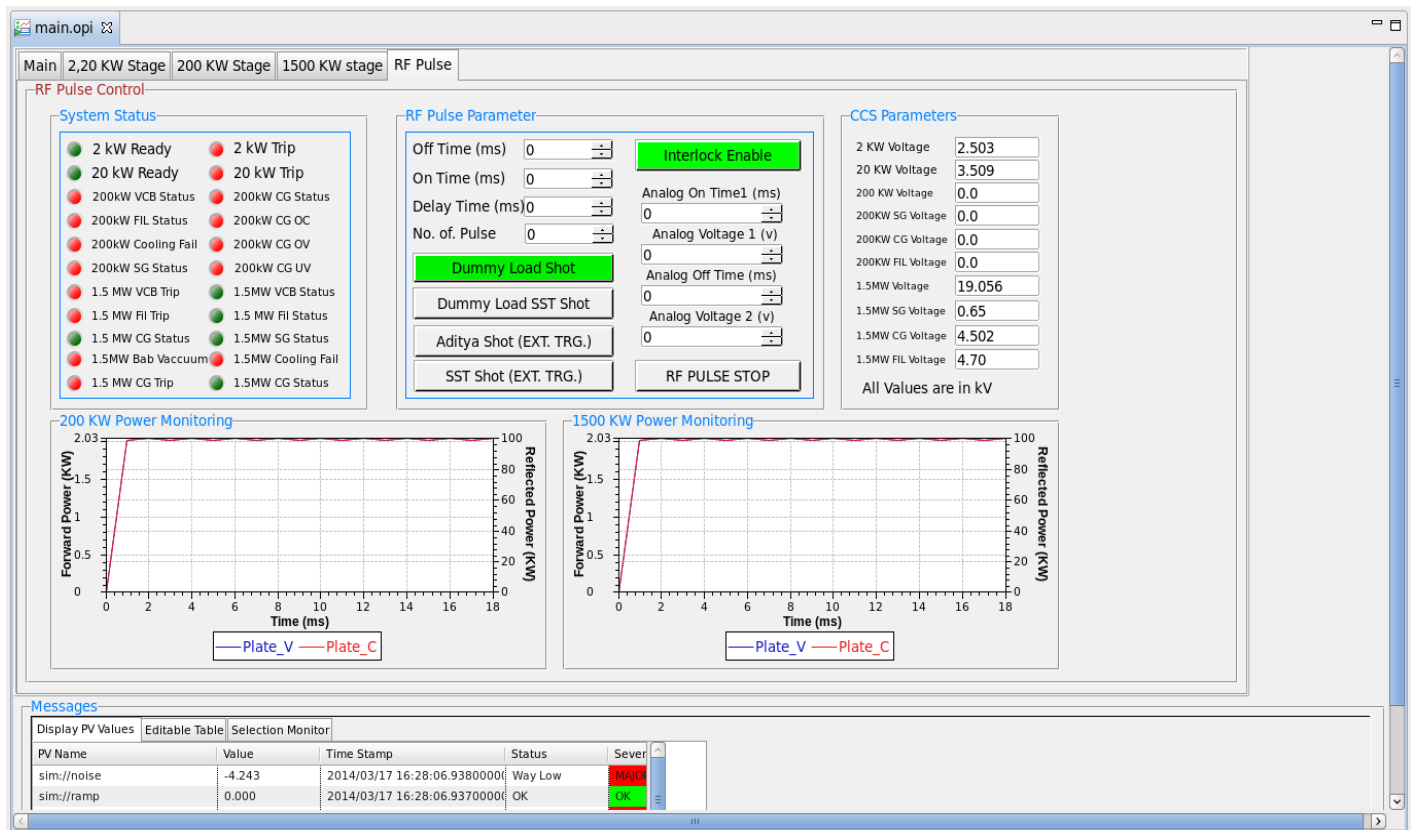
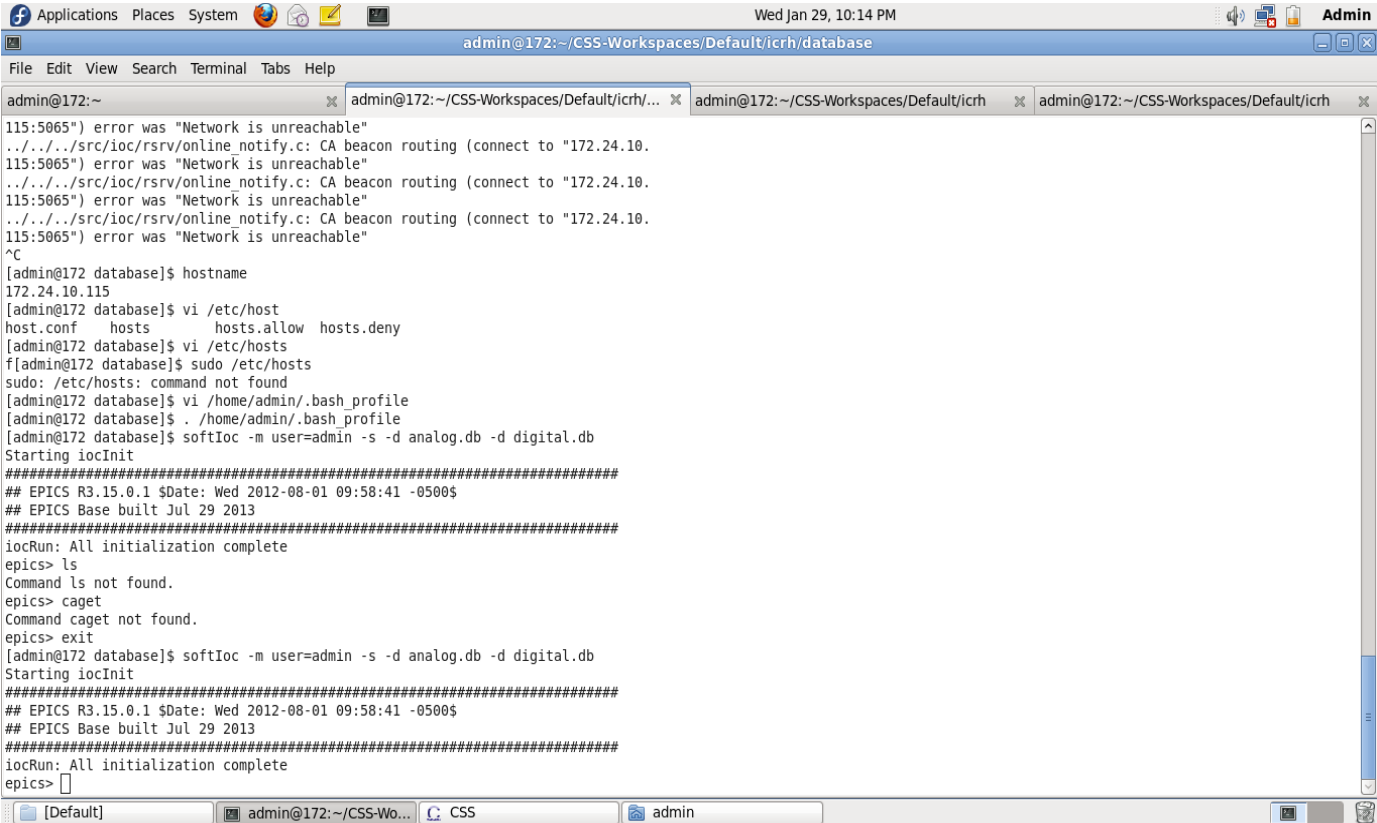


Figure. 4: Experimental shot monitoring and control screen for DAC software

It is important to note that most signals are not monitored by channel access clients and that monitors are only sent on change of state or excursion outside of a dead-band. The database scanning is flexible to provide optimum performance and minimum overhead.



```

admin@172:~
115:5065") error was "Network is unreachable"
../../../../src/ioc/rsrv/online_notify.c: CA beacon routing (connect to "172.24.10.
115:5065") error was "Network is unreachable"
../../../../src/ioc/rsrv/online_notify.c: CA beacon routing (connect to "172.24.10.
115:5065") error was "Network is unreachable"
../../../../src/ioc/rsrv/online_notify.c: CA beacon routing (connect to "172.24.10.
115:5065") error was "Network is unreachable"
^C
[admin@172 database]$ hostname
172.24.10.115
[admin@172 database]$ vi /etc/host
host.conf  hosts      hosts.allow  hosts.deny
[admin@172 database]$ vi /etc/hosts
f[admin@172 database]$ sudo /etc/hosts
sudo: /etc/hosts: command not found
[admin@172 database]$ vi /home/admin/.bash_profile
[admin@172 database]$ ./home/admin/.bash_profile
[admin@172 database]$ softIoc -m user=admin -s -d analog.db -d digital.db
Starting iocInit
#####
## EPICS R3.15.0.1 $Date: Wed 2012-08-01 09:58:41 -0500$
## EPICS Base built Jul 29 2013
#####
iocRun: All initialization complete
epics> ls
Command ls not found.
epics> caget
Command caget not found.
epics> exit
[admin@172 database]$ softIoc -m user=admin -s -d analog.db -d digital.db
Starting iocInit
#####
## EPICS R3.15.0.1 $Date: Wed 2012-08-01 09:58:41 -0500$
## EPICS Base built Jul 29 2013
#####
iocRun: All initialization complete
epics>

```

Figure. 5: Broadcasting of Process Variables using softIOC for DAC communication

V. TEST SETUP AND RESULT ANALYSIS

The prototype system provides an environment for implementing systems that requiring several hundred points on periodic monitor and control with tens of physical connections. The EPICS environment supports system extensions at all levels, enabling the user to integrate other systems or extend the system for their needs. Through the modular software design which supports extensions at all levels, we are able to provide an upgrade path to the future as well as an interface to an installed base. The network overhead can be reduced drastically as well as no requirement of multiple connections each time of data communication needed. Light weight broadcasting mechanism provides easy communication which addresses the experimental setup for collaborative network based environment. Easy and open source solution provides reliable and fail safe operation using EPICS and CSS combination. For future, we are planning to improve further and make it better tool.

The following system setup was used for the EPICS benchmark database measurements:

- EPICS version 3.14.12.2
- CSS Opi for User Interface Development
- Intel Core i5 with Fedora OS 14
- PC with Linux OS
- 100 MBPS Ethernet segment

It should be noted that resource usage depends not only on the total number of records processed/sec but also on the number of channel access (CA) clients. This was measured by running several copies of the benchmark display. Figure 6 show resource usage vs. number of connected CAs. In all cases the 10 Hz scan rate was used. Thus for 100 CA clients 1000 records/sec were processed. Fig. 7 shows the memory, network and CPU utilization with different options of user interface like only broadcasting of PVs, with TK interface, with CSS interface and both interface simultaneously running. CSS provides facility with Java and python programming which gives more reliability and one can develop platform independent programming using Java.

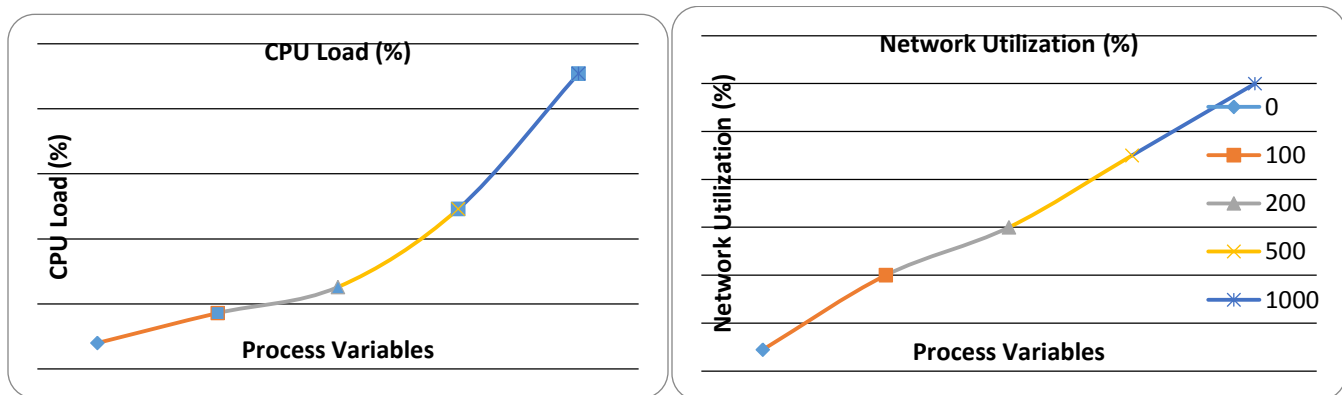


Figure. 6: IOC CPU Load and Network Utilization at 10 Hz Scan Rate

The IOC provides the physical interface to a portion of a machine. The limiting factors in the performance of the IOC are the CPU bandwidth and memory. For analog inputs, scanning on end-of-conversion significantly reduces the latency between gating a signal and processing the record. The database scanning is flexible to provide optimum performance and minimum overhead. For time measurement, we set a monitor PV, and the PV can automatics change its value at regular time intervals. ITER Control Data Access and Communication (CODAC) has identified for the development platform [18]. Instead of using the original EPICS IOC [19, 20], we decided to develop our own EPICS software toolkit, custom implementation, which is capable of building EPICS Channel Access (CA) server and client programs. Communication performance is bounded by the channel access protocol, TCP/IP packet overhead, and the physical communication media. Channel access makes efficient use of the communications overhead, by combining multiple requests

or responses. To avoid collisions and therefore avoid non-determinism, the Ethernet load is kept under 30% [21]. At this level, we can issue 10,000 monitors per second. Use of LAN bandwidth can reduce by 50%-80% by changing the channel access protocol to variable command format and compressing the monitor response data (~ 6-15 bytes per packet). In J-TEXT, EPICS provides improve productivity with channel access [22]. SPIDER tokamak has also support same kind of implementation with performance [23].

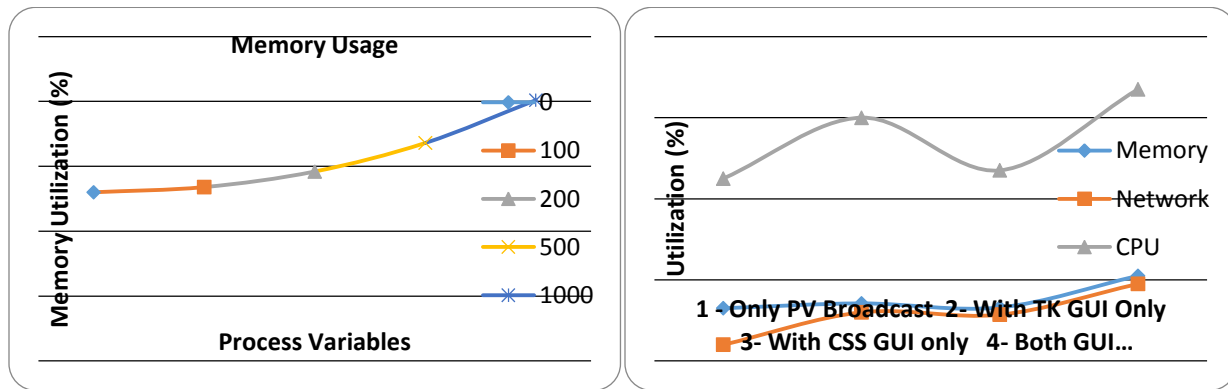


Figure. 7: Memory Utilization at 10 Hz Scan Rate and Parameters with different cases

Channels as process variables have been broadcasted every required time period so when the shot is applied to master DAC, the information has been automatically available to slave DAC on same network. A prototype system has been developed for the same. One EPICS based prototype has been about to complete by which we can produce results and check performance parameters with standard optimization results. We ran the benchmark with the scanning period of 1, 0.5, 0.2, 0.1, and 0.01 second, which corresponds to 100, 200, 500 and 1000 processed records/sec. Tables 1 shows the IOC CPU usage and network utilization respectively. Linux System monitor was used to measure the CPU and memory usage, and the Wireshark network Analyzer tool was used to measure the network segment utilization.

If the record of process variables would be broadcasted than the CPU utilization for every 100 millisecond we can get Table-1 first row results. If the record of process variables would be broadcasted in the network at every 100 millisecond we can get Table-1 second row results. During shot each connected sub systems are working together and client computer acquires data from fast real-time server which increase the CPU load up to 90 percent and network load up to 82 percent as shown in both the table last column. After completion of the experimental shot it come down to the normal state.

TABLE I. Ioc Cpu Load And Network Load

Record Processed (100 ms)	0	100	1000	During Shot
CPU Load (%)	2	4.3	22.7	90
Network Load (%)	0.89	4	12	82

VI. CONCLUSION

The prototype system provides an environment for implementing systems that requiring several hundred points on periodic monitor and control with tens of physical connections. The EPICS provides an environment for implementing systems that range from small test stands requiring several hundred points per second to large distributed systems with tens of thousands of physical connections. The base software is also being extended to support some of the fundamental needs of the projects that are controlling user facilities. In IOC, either CPU or memory may become the bottleneck depending on number of PVs, clients, and scan rate. However the scalability of EPICS allows a redistribution of databases and applications to balance the load. Network overhead can be removed with EPICS light weight broadcasting mechanism with the help of channel access layer.

VII. REFERENCES

1. *Engineering design report (EDR) of ICRH-SST1; SST-2.03-050199-RF Group*
2. *Durodie F., Veriver M.; European physics society (EPS), 1992,p 81*
3. *D. Bora et al, Nucl. Fusion 46 (3) (2006)*
4. *The Plant Control Design Handbook, <http://www.iter.org/doc/www/edit/Lists/WebsiteText/Attachments/94/PCDHv.61.pdf>*
5. *Ramesh Joshi et al. EPICS and MDSplus integration for ICRH data acquisition, International Journal of Computer Science Research and Application, 2013*
6. *Xihui chen et al. BOYA MODERN GRAPHICAL OPRATOR INTERFACE EDITOR AND RUNTIME, Proceedings of 2011 Particle Accelerator Conference, New York, USA*
7. *Experimental Physics and Industrial Control System Home Page, <http://www.aps.anl.gov/epics/>*
8. *Ctypes Python Package, <http://python.net/crwe/theller/ctypes/>.*
9. *EPICS Home Page. [Online] Available <http://www.aps.anl.gov/epics>*
10. *PyEpics Python Package, <http://cars.uchicago.edu/software/python/pyepics3/>*
11. *Benjamin G Penaflor et al. Custom open source solution for DIII-D data acquisition and control system, Fusion Engineering and Desing 87 (2012) 1977-1980*
12. *Ramesh Joshi et al. Conceptual design of EPICS based implementation for ICRH DAC system, Conference preceding Springer, March-2014*
13. *Paulo Carvalho, et al., EPICS IOC module development and imple- mentation for the ISTTOK machine subsystem operation and control, Fusion Engineering and Design 86 (2011) 1085–1090.*
14. *Wei Zheng et al. J-TEXT-EPICS:An EPICS toolkit attempted to improve productivity, Fusion Engineering and Design 88 (2013) 3041-3045*
15. *Architecture of SPIDER control and data acquisition system, A. Luchetta et al., Fusion Engineering and Design 87 (2012) 1933-1939*



16. Shifu Xu et al., EPICS IOCCORE REAL TIME PERFORMANCE MEASUREMENT ON COLDFIRE MODULE, *Proceedings of PCaPAC 2010, Saskatoon, Saskatchewan (2010)*.
17. M. Eguiraun et al. NETWORKED CONTROL SYSTEM OVER AN EPICS BASED ENVIRONMENT, *Proceedings of IPAC'10, Kyoto, Japan (2010)*
18. *The Plant Control Design Handbook*, <http://www.iter.org/doc/www/edit/Lists/WebsiteText/Attachments/94/PCDHv.61.pdf>
19. *Experimental Physics and Industrial Control System Home Page*, <http://www.aps.anl.gov/epics/>
20. Paulo Carvalho, et al., EPICS IOC module development and implementation for the ISTTOK machine subsystem operation and control, *Fusion Engineering and Design* 86,2011,1085–1090.
21. Nemzow, M., "Keeping the Link: Ethernet: Ethernet Installation and Management", *Magraw-Hill Book Co.*, pp. 219-220.
22. Wei Zheng et al. J-TEXT-EPICS:An EPICS toolkit attempted to improve productivity, *Fusion Engineering and Design* 88,2013,3041-3045
23. Architecture of SPIDER control and data acquisition system, A. Luchetta et al., *Fusion Engineering and Design* 87,2012,1933-1939