THE IUAC TANDEM-LINAC CONTROL SYSTEM

Ajith Kumar, E.T. Subrahmaniam, K. Singh, B. Sahu, D. Munda
Inter-University Accelerator Centre, Aruna Asaf Ali Road, New Delhi, India

Abstract

The 16MV Tandem Van de Graaff accelerator at IUAC is one of the earliest machines to go for a PC based control system. The PDP11, supplied with it, was replaced by an IBM PC running DOS before the accelerator was commissioned in 1989. The present system, commissioned in 1997 and supports LINAC, runs on a network of PCs under the GNU/Linux operating system. Every parameter to be controlled/monitored is assigned a unique identifier consisting of a Label, Function and Unit. The signals are grouped according to location and they are connected to server PCs using interfaces like CAMAC, VME or custom hardware. The features like a user interface, alarm systems, data logging and partial automation are handled by several client programs, communicating to multiple servers to access the hardware over a TCP/IP connection.

INTRODUCTION

Inter-University Accelerator Centre has a 16 MV Tandem Van de Graaff accelerator and a super-conducting heavy ion LINAC. The Tandem was originally supplied with CAMAC Serial Highway Interface and a DEC PDP11 based control system. The PDP11 was replaced by a PC based system running DOS before the accelerator was commissioned [1]. Later on, the addition of the LINAC demanded features like multiple operator consoles and the ability to run special purpose programs to condition the resonator cavity, automatically setting the amplitude and phase of RF etc. To meet the new requirements, a new control system was developed with the following basic guidelines; low implementation and maintenance costs, reliability, flexibility and scalability. PCs connected over Ethernet, running GNU/Linux operating system, forms the basic computer network required for the system. CAMAC, VME and custom hardware are supported.

DESIGN

The main task of the accelerator control system is to monitor/control signals from various devices like power supplies, vacuum systems, beam monitors an so on. Each device may have one or more analog/digital signals to be controlled or monitored, resulting in a large pool of signals. We have assigned a unique identifier for each signal. Three character strings, Name, Function and Unit, uniquely identify each parameter in the whole system. For example “CPS031”, “VC”, “KV” identifies the Charging Chain power supply voltage control signal. This nomenclature is followed mainly because our accelerator people were already familiar with it. The signals are grouped according to location and each group is connected to server PC, closer

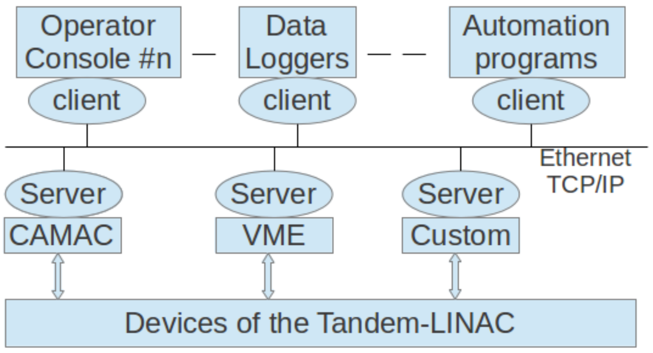


Figure 1: Schematic of Control System Hardware.

to the devices, using interfaces like CAMAC, VME or custom hardware.

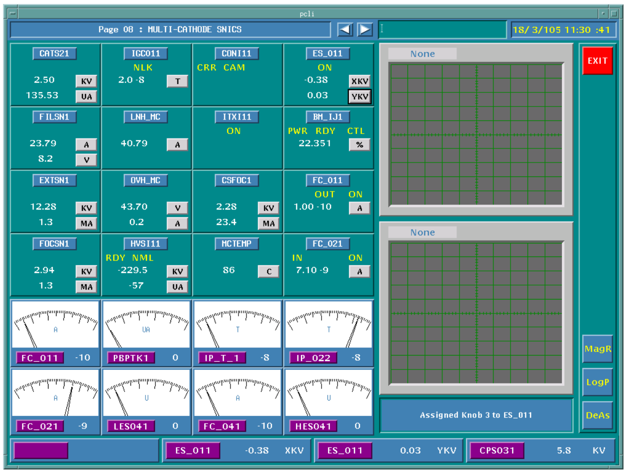
The computers connected to the hardware runs a server program supporting the control/monitor of the signals over a TCP/IP connection. Only the unique signal identifier is known outside the server, the hardware details and the complexities of all the signals are hidden from the outside world. A simple message passing protocol, over a TCP connection, supporting functions like getState, setState, getValue and setValue are used for accessing the signals from remote computers running client programs.

HARDWARE

In the accelerator we have devices requiring analog and digital type signals for controlling/monitoring. Signals from devices located nearby are grouped and connected to one computer, using CAMAC, VME or some custom hardware interface. Several such groups are required to take care of all the signals. Each signal is characterized by a unique name and interfacing details like hardware address and datasize. Most of them are handled using standard modules like ADC, DACs, Output Registers and Input Gates. Special modules were made for interfacing the Beam Profile Monitor and the LINAC resonator controllers. The CAMAC Crate Controller used in the system are equipped with a built-in single board computer having PC104 interface and Ethernet port. On power-up the CAMAC controllers boot from a central server, using the Linux Terminal Server software. The VME Crate controllers also have embedded PCs inside them with Ethernet interface. All of them are connected to the same Ethernet network.

Computers providing the user interface are equipped with four shaft encoder knobs, acting as incremental input devices, for setting the value of analog signals. They are

ISBN 978-3-95450-124-3



CATS21 2.50 KV 135.53 UA
IGC011 NLK 2.0 -8 T
CONI11 CRR CAM
ES_011 ON -0.38 XKV 0.03 YKV
FILSN1 23.79 A 8.2 V
LNH_MC 40.79 A
ITXI11 ON
BM_IJ1 PWR RDY CTL 22.351 %
EXTSN1 12.28 KV 1.3 MA
OVH_MC 43.70 V 0.2 A
CSFOC1 2.28 KV 23.4 MA
FC_011 OUT ON 1.00 -10 A
FOCSN1 2.94 KV 1.3 MA
HVSI11 RDY NML -229.5 KV -57 UA
MCTEMP 86 C
FC_021 IN ON 7.10 -9 A
FC_011 -10
PBPTK1 0
IP_T_1 -8
IP_022 -8
FC_021 -9
LES041 0
FC_041 -10
HES041 0
ES_011 -0.38 XKV
ES_011 0.03 YKV
CPS031 5.8 KV

Figure 2: Screen-shot of operator console program.

connected to the USB port of the PC using custom designed hardware. A schematic of the control system hardware is shown in Figure 1.

SOFTWARE

We have followed a distributed approach and each server has a database storing the details of signals connected to it, there is no centralized database. The databases contain static and dynamic fields of information about the connected signals. The static fields are loaded from a file during startup and the dynamic fields are periodically updated from the hardware. All servers listen over the network, using TCP protocol, for client requests and establishes client connections, after authentication. The only way to access any signal is through a TCP/IP socket, even if it is from the same computer. A server can establish connections to several clients at the same time.

The Client programs provide the user interface, alarm services, logging etc. Every client program works on a set of signals, usually located at different servers. During startup, the required server addresses and the signal identifiers are loaded from a file and the server address of every signal is identified by searching for it, using the identifier. After that, the client program accesses the signals by sending command packets to the appropriate server. The packet will have the signal identification and the operation to be carried out and the reply will contain the requested information. The message packet implements functions like getState, setState, getValue and setValue, for managing analog and logical type signals.

A client program could be a small one just recording a single parameter or one that can be used for tuning the beam through the entire accelerator. A screen-shot of the most commonly used client program is shown in Figure 2. This GUI is implemented using X-Windows and Motif library. The total number of parameters is grouped into "Pages" for the convenience of displaying. The user defines the page layout by editing text files, which will have the parameters identified by the signal identifier mentioned earlier. All the available pages are loaded during startup and the server for each parameter is identified. The user requests the client to access the various parameters by clicking the mouse and by turning the Shaft Encoder knobs. Client sends requests to the servers for getting and setting various parameters controlling the operation of the accelerator and displays the results. Dependencies between parameters can be defined in a text file that is loaded during startup and the relationships examined periodically to take necessary actions. The interlocks, alarm and a diagrammatic display of the machine are currently implemented using this facility.

The Python Library

Python is an interpreted language with a simple syntax and it is relatively easy to learn. Providing a Python library to access the accelerator parameters enables people working on the accelerator development to write simple scripts for various applications [3]. The unique identifier for each signal makes this task very easy. For example, the Python program given below prints the filament current of the SNICS ion source.

```
import pelcon
p = pelcon.pserv()
print p.get_value('FILSN1', 'CR', 'A')
```

The program can run from any computer connected to the control system network. The only information required is access the signal is the identifier consisting of the three character strings. The controlling/monitoring are done mostly using the four functions, get_value(), set_value(), get_state() and set_state(). Multiple clients may try to access the same parameter but the server serializes the requests to avoid any conflict. The Python library is being used for developing programs for data logging and partial automation of the accelerator operation [4].

It is very easy to write GUI programs in Python using one of the toolkits like Tkinter, pyGTK ot PyQt. The introduction of Python language to the control system was due to our experience with it in another project, described in the next section.

RELATED DEVELOPMENTS

The function of an accelerator control system is very similar to that of the computer interfaced laboratory equipment employed in teaching science, the technologies used are more or less the same. We have developed a very low cost hardware device that provides several analog/digital channels where sensor/control elements can be connected. It consists of a micro-controller with a USB interface and some signal processing circuits. The design schematic is shown in Figure 3. The micro-controller runs a server program that listens for the commands coming from the computer over a USB link, performs the requested actions and sends the response back to the PC. The measurements, requiring real-time capability, are done by the

ISBN 978-3-95450-124-3

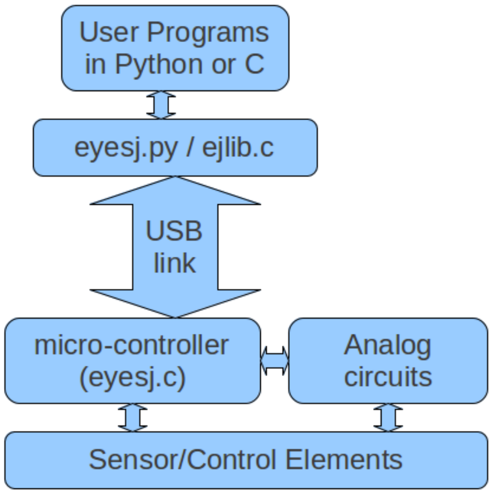
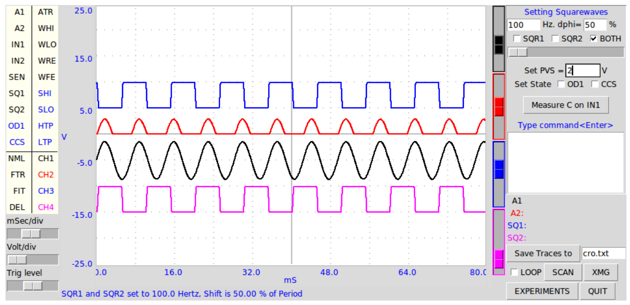


Figure 3: Schematic of expEYES design.


A1 ATR A2 WHI IN1 WLO IN2 WRE SEN WFE SQ1 SHI SQ2 SLO OD1 HTP CCS LTP NML CH1 FTR CH2 FIT CH3 DEL CH4
mSec/div
Volt/div
Trig level
25.0 15.0 5.0 -5.0 -15.0 -25.0 V
0.0 16.0 32.0 48.0 64.0 80.0 mS
Setting Squarewaves
Measure C on IN1
Type command<Enter>
A1 A2: SQ1: SQ2:
Save Traces to
LOOP SCAN XMG
EXPERIMENTS QUIT

100 Hz. dphi= 50 %
SQR1 SQR2 [✓ BOTH]
Set PVS = 2 V
Set State OD1 CCS
Save Traces to cro.txt
SQR1 and SQR2 set to 100.0 Hertz, Shift is 50.00 % of Period

Figure 4: expEYES working as an oscilloscope in the audio frequency range.

micro-controller running a C program. The data analysis and the user interface, written in Python or C, runs on the PC. A hardware communication library and several GUI applications are written in Python to access the sensor/control elements connected it.

The device is called expEYES and it supports more than fifty experiments starting from high school level. The design is open and the royalty-free hardware is currently available for around US$ 30/-, from several vendors. The hardware is shown in Figure 5 and a screen-shot of it working as a low frequency oscilloscope is in Figure 4. The details are given on the website, http://expeyes.in [5].

ExpEYES hardware has been used for measuring the frequency error signals from the LINAC resonators. It has been integrated into the system using the same scheme used for CAMAC and VME. This system can be used by engineering students to practice the basics of computer interfacing and some of the techniques used in writing control systems.

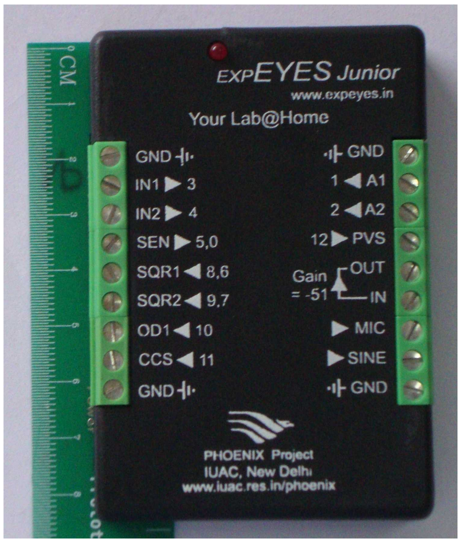

CM
EXPEYES Junior
www.expeyes.in
Your Lab@Home
GND IN1 3 IN2 4 SEN 5,0 SQR1 8,6 SQR2 9,7 OD1 10 CCS 11 GND
GND 1 A1 2 A2 12 PVS Gain = -51 OUT IN MIC SINE GND
PHOENIX Project
IUAC, New Delhi
www.iuac.res.in/phoenix

Figure 5: Photograph of expEYES hardware.

CONCLUSION

The distributed control system, using a simple message passing protocol, was operational from 1997. In the beginning only CAMAC interface was used but it was easy to add support for other hardware interfaces due to the distributed design. The addition of Python library has made program development much more easier and faster. The control system has demonstrated very high reliability,scalability and ease of use. The development work also resulted in some spin-off products that are now being used for teaching science.

REFERENCES

[1] Ajith Kumar et al., "The NSC 16 MV tandem accelerator control system", NIMA 343(1994) 327 – 330.

[2] Ajith Kumar et.al., "A distributed Control System for NSC Tandem-LINAC", PCaPAC2005, Hayama, Japan, 22-25 Mar 2005.

[3] Ajith Kumar, "A Python based interface for the Tandem-LINAC control system", INPAC 2011, IUAC, Delhi, Feb., 15-18, 2011

[4] B.K. Sahu et.al., "Automation of the Control Scheme for IUAC LINAC", INPAC 2011, IUAC, Delhi, Feb., 15-18, 2011

[5] `http://expeyes.in`