Remote control system of VME crates based on CAN-Bus in PITZ experiment in DESY Zeuthen.

K. Abrahamyan[$], I. Chachkhunashvili[&], G. Trowitzsch,
DESY, Zeuthen, Germany.
[$]on leave from YerPhI, 375036, Yerevan, Armenia.
[&]on leave from HEPI, 380086, Tbilisi, Georgia.

Abstract

At many facilities in DESY, including PITZ (Photo Injector Test facility at Zeuthen) or TTF (Tesla Test Facility at Hamburg), lots of VME crates are used in control systems. Sometimes it is necessary to switch crates 'on' or 'off', to generate system resets or to check the current status of crate like temperatures, power channels, voltages and currents, etc.. Normally the operator has to do it directly on the control panel of every VME crate, but it is not convenient or simply impossible if the crate is located in an area with restricted access. Therefore it is necessary to have uninterruptible and simultaneous remote access to control the VME crates. On the basic level we decided to use the CAN-Bus connection to the VME crates provided by the crate manufacturer. It is a robust field bus supporting up to 127 nodes per segment. As the host platform a PC equipped with a PCAN-PCI controller running Windows NT was chosen, because the solution was cheap, easy to implement and completely independent from the target systems. The TINE Protocol [4] is used to integrate it into the PITZ control system, to access it from different client applications. A first Win32 based client application provides all functions to monitor the VME crates and control first parts. The most important functions are: reading the status of a crate; switching power 'on' or 'off'; changing fan speeds interactively; monitoring temperatures, voltages, currents and errors.

1. Introduction

The Photo Injector Test facility at DESY Zeuthen (PITZ) [1], as shown on Figure 1, was built to develop electron sources for the TESLA Test Facility Free Electron Laser (TTF-FEL) and future linear colliders.

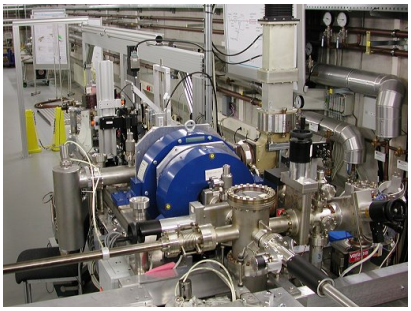

Fig.1: Photo Injector Test facility Zeuthen

VME crates are playing an important role in control systems of many experiments and accelerators itself. The task was to design and implement an efficient and simple remote control system for VME crates, to monitor parameters and to control the device. It becomes more important for crates located at places with restricted access near to the accelerator. The goal was to integrate it into the common PITZ control system.

1. Hardware base

1.1 The VME Crates

Only crates from one manufacturer are used [2]. The VME crate consists of power supply (UEP 4020), bin tray (UEV 4020) and fan tray (UEL 4020). All components are pluggable and easily to exchange. The fan tray contains the CAN interface (Fig. 2). Using external cooling a VME-control and monitoring module is necessary to provide the remote control block.

Fig. 2

All crates contain the CAN-Bus based remote control block in our installation and therefore a standard PCI based CAN-Bus controller was chosen. We are using the PCAN-PCI-Card manufactured by "PEAK-System Technik" [3].

Figure 3 shows an overview of the crate remote control system.

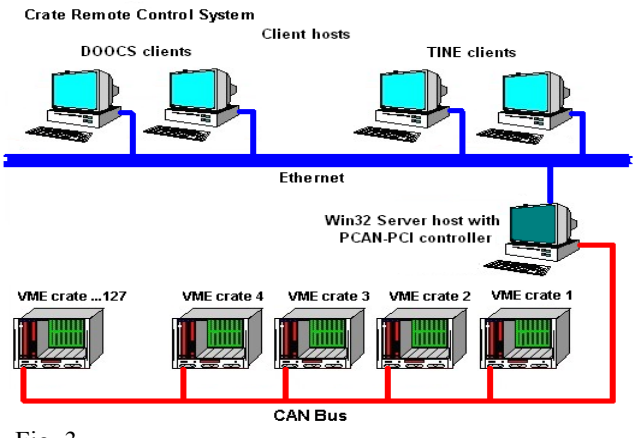

Fig. 3

2.2 CAN-Bus controller

The card (Fig. 4) is equipped with the CAN-controller SJA1000 and the driver 82C251. The connection to the CAN-bus is done via 9-pin SUB-D plug, whose pin assignments corresponds to the CiA-recommendation DS 102-1. The driver of the PCI card is only available for Windows 9x/ME and Windows NT/2000. The control server application was implemented on NT4 to integrate it in our DESY NT domain. We decided to use the TINE (Three-fold Integrated Networking Environment) Protocol [4] as the network transportation layer. It is easy to implement on Win32 and a framework to create slow control application server was available. Furthermore it was easy to integrate into DOOCS (Distributed Object Oriented Control System) [5] that is used at PITZ.

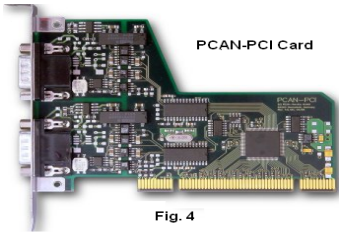

Fig. 4

3. Software structure

3.1 TINE Slow Control Framework

TINE protocol is one of base protocols used at DESY. A lot of Slow Control Servers were and will be developed for different tasks. We created a TINE based Slow Control Framework on Borland C++ Builder. This Framework saves time on developing the communication part of the server and the programmer can concentrate on developing device specific parts of code that can be easily embedded into the framework. The framework contains a list of objects that has to be assigned to the specific "Device Controller" objects. All that has to be done is to change constructor calling (if necessary) and to add project files that describe the device specific objects. To add new TINE properties to the server, one has to edit the function "ServerEqpFcn". This function processes TINE messages on server side and one has to add properties to the configuration file of the application. The configuration file should be named like the application, but with the extension INI. Furthermore it is used to configure the device objects. The framework will take care about registering the properties and the server itself on the TINE Name Service (ENS). It also provides function to simplify communication with the client, to handle errors and to provide a debug window at runtime.

3.2 CrateServer(TINE Server).

Figure 5 shows the structure of the application.

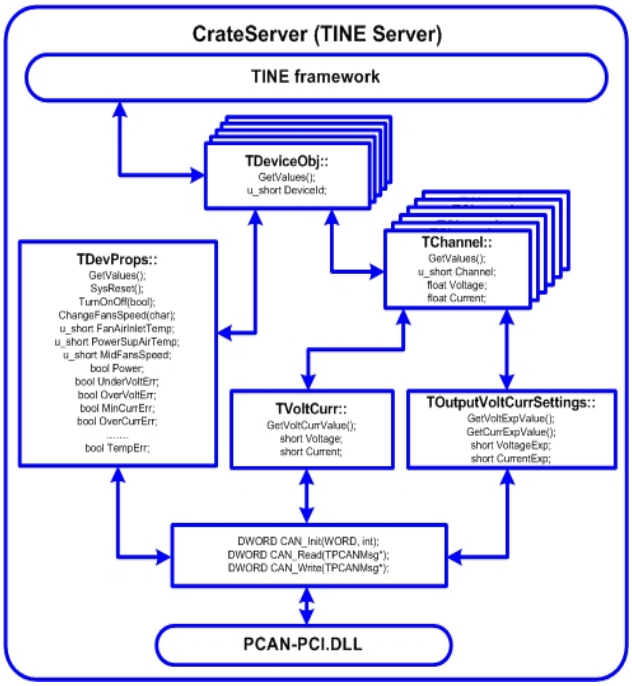

Fig. 5: Basic structure of CrateServer

For each crate a device object (TDeviceObj) is created. The properties are divided in two classes: TChannels (currents/voltages for power channels) and TDevProps (states, temperatures, fans speed, commands, etc.). For TChannel we have also two types of objects: TVoltCurr (to get voltages and current from the VME crate, depends on the power channel) and TOutputVoltCurrSettings (to get the exponents of the float values of voltages and currents). The functions CAN_Init(), CAN_Read() and CAN_Write() are used to send data to or receive data from the PCAN-PCI Card driver (Pcan_pci.sys) by using the dynamic library PCAN-PCI.DLL. For CAN_Init() the arguments are the speed of CAN-Bus (depends on the length of the CAN-Bus) and type of CAN-Message (normal or extended). CAN_Read() and CAN_Write() use a pointer to the CAN-Message structure (TPCANMsg).

3.3 Client application (GUI)

The DOOCS Graphical User Interface based on DOOCS Data Display (DDD) [6] is a simple example of a client application (Fig. 6).

It shows five VME crates with the following data: power state, temperatures, fans speed, voltages and currents for three power channels. The power channel configuration depends on the power supply of the particular crate. It is adjustable by the configuration file. In our case, from the 8 possible power channels only three are used.

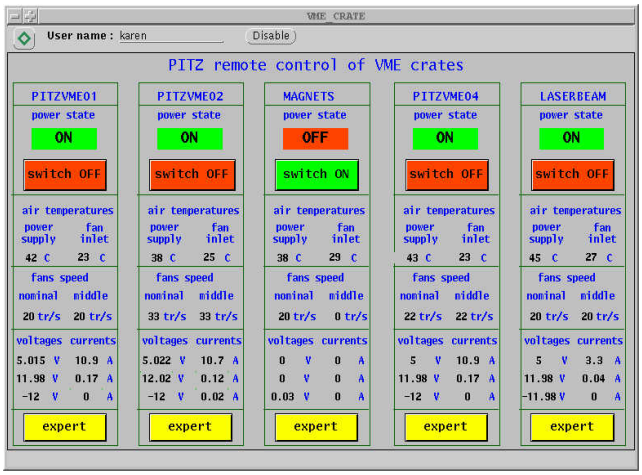

Fig. 6: DOOCS GUI

The DOOCS GUI allows to protect functionality by user name and password. In this case, switching on/off was restricted. Figure 7 shows the authentication dialog box.

Fig. 7

As the protection password the users UNIX password is used (PAM Authentication Method).

4. Future developments

The following developments are planned:

- storing history data for some properties like power states of crates, CAN Bus errors, etc.
- implementation of watchdog applications to generate soft alarms for important parameters like over-/under-voltage, min-/over-current, temperatures, to act on hard error of crates and to control parts like fans speed depending on monitoring data
- improvement of existing client applications

5. Acknowledgements

The authors wish to express their gratitude to our colleagues at PITZ and DESY Zeuthen, helping with ideas and own experience. Philip Duval, author of TINE, helped us a lot to include TINE support and to get it running properly.

References

[1] http://desyntwww.desy.de/pitz/
[2] http://www.wiener-d.com/
[3] http://www.peak-system.com/
[4] http://desyntwww.desy.de/tine/
[5] http://tesla.desy.de/doocs/
[6] http://tesla.desy.de/doocs/doocs_gen/ddd.html