DEAR Technical Note IR-21

# A New CCD Command Box

Ross King
Christian Müllner

IMEP, ÖAW, Vienna

May 2, 2000

## Motivation

A diagram of the present CCD readout system is shown in Figure 1A. This system, developed principally by IPH Neuchatel (Egger, Varidel) and ETH Zürich (Pollet), consists of many components:

- the data acquisition PC

  This is a PC running under Windows 3.11, on which the control software, storage, and online analysis programs are carried out.

- the VME interface

  An SBS/BIT-3 model 406 (or 406-1) AT/VME bus interface provides virtual memory

mapping of the VME backplane to the PC. The complete interface consists of an ISA-bus card in the PC and a 6U VME card in the VME crate.

- the fiber-optic interface

  The IPH system was designed such that the data acquisition PC and the VME crate would stand in the control room and the Command Box in the experimental area. A direct cable connection between the JVM and the Command Box works with a cable up to 30 meters long. However, for DEAR we require at least 70 meters between the control PC and the setup. Our solution is to increase the distance between the PC and the VME crate, using fiber-optic cables. An SBS/BIT-3 model 400-50 stand-alone unit next to the data acquisition PC and the model 400-5 VME card, connected by fiber-optic cables (BIT-3 model 500-500-100), allow the distance between the 406 VME interface components to be increased to up to 2000 meters.

- the JVM/VME crate

  The JVM (JUMBO VME Module) is the image buffer for the CCD image acquisition. When CCD data is converted in the ADCs, the digital results are written directly to the RAM of the JVM. After conversion is complete, the JVM data is compressed and written to magneto-optic disk by the data acquisition computer.

- the MAHOUT component

  This consists of a GESPAC model GESMPU-4B (a Motorola 68000 processor running at 8 MHz!) and a GESPAC model GESSIO-1B serial interface on a GESPAC GESBUS-16A backplane. In other words, it is a very primitive embedded computer. The CPU receives ASCII commands issued by the control computer over an RS-232 serial line, and writes the appropriate digital commands to the JUMBO backplane.

- the JUMBO unit

  This unit consists of several modules which control the timing of the CCD readout phases and the ADC conversion. Also present but unused is a temperature control and monitor

module. The combination of MAHOUT and JUMBO is referred to as the “Command Box” (or, in the old days, the “Blue Box”).

- the Readout Boxes

  These units consist in turn of two sub-units, the DRV and RDO cards. The DRV unit generates the pixel and line transfer phases for the CCD readout, given timing signals from the JUMBO components (LPH, RPH boards). The RDO unit is the dual slope integrating amplifier for the CCD signals.

This system (from now on referred to as the IPH system) has many shortcomings:

- the control software runs only under Windows 3.11

  In addition to requiring an outdated operating system, the source code is difficult to compile and port to Windows NT.

- the JVM is a poorly documented custom component

  In addition, there are numerous undocumented corrections to the circuit boards. These units are very sensitive to high-frequency noise, and they were very difficult to bring into operation. At present, we have three working modules (JVM2, JVM3, and JVM4), but only JVM2 has seen extended laboratory and experimental use, and thus the reliability of the latter two modules is unknown.

- the JVM is limited by 8 MB of image memory (RAM)

  At the time the IPH system was developed, 1 MB was still a lot of RAM. A custom-made image buffer was required to handle the massive data stream from four CCD-05-20 chips (8 MB) – thus the JVM was born. However, the 8 MB JVM memory limits the number of pixels that can be read out such that it is possible to read four CCD-05-20 chips (1152 x 770 pixels) or eight CCD-22 chips (600 x 600 pixels), but not four CCD-05-30 chips (1152 x 1242 pixels).

- the MAHOUT system is undocumented

  We have some undocumented source code for this system, but even the author (P. Pollet of PSI) does not know if the code we have is equivalent to that burned into the MAHOUT EPROMS. We also have no compiler for this CPU, so the only way do duplicate the system is to directly copy the original EPROMs in Command Box 2.

- commands are issued over a separate RS-232 cable

  In the original IPH design, commands issued by the data acquisition PC should have been transmitted to MAHOUT through the VME backplane. For some reason, this never worked and was never implemented. Instead, commands are written on the serial port of the control PC and travel over an RS-232 cable to the JVM. From there, they are transmitted over the JVM-VLK 64-pin cable and distributed to the MAHOUT serial interface. In principal, the RS-232 standard is only adequate for distances up to 15 meters. We did find in practice that this system continues to work even with 100 m cables. But there are two results: first, we require both the fiber-optic cable and the serial cable, and second, the 2000 m range of the fiber-optic cable is useless, since the real range is limited by the RS-232 cable.

Given these problems, the motivation for a new system is clear.

## Implementation

The availability of embedded CPUs with extensive on-board memory and fast ethernet allow a design that eliminates the JVM component completely. Refer to Figure 1B.

It is simple to eliminate the MAHOUT component of the IPH system: this CPU is so outdated by today's standards that its functionality is easily duplicated by the cheapest available CPU.

Also, as already mentioned, the JVM memory of 8 MB is easily improved by any off-the-shelf computer today, which are typically configured with 32 MB standard RAM and can be

expanded up to 256 MB or more. However, the digital I/O function of the JVM is not so easily replaced. The data stream consists of 16-bits of ADC/addressing data, 8 bits of data to define which CCDs are selected, and four bits of Command data. The required data transfer rate is a modest 14 kHz, due to the high-resolution slow readout of the CCD.

We selected the National Instruments DIO-6533 32-channel digital input/output card to handle the I/O functions. We needed a minimum of 28 channels, some input only, some output only, and some mixed. The DIO-6533 is an off-the-shelf component that allows 32 channels of mixed I/O with handshaking and DMA (Direct Memory Addressing). The unit operates on the Compact PCI bus, which fits in the same standard housing used by MAHOUT, and for which many embedded processors are available.

Given that the National Instruments drivers for the DIO-6533 are available for Windows and Macintosh operating systems only, the obvious choice for an embedded computer was an Intel processor running Windows NT – the former because a large market for such embedded PCs exists, the latter in order to be compatible with the DEAR network. We eventually chose the CSSIPC-PIIC from CSS Industrie Computer, Germany. This is a 266 MHz Pentium-II processor with 64 MB RAM and built-in video, IDE, and ethernet controllers. In addition, we purchased and installed a 6.4 GB hard drive and a CD-ROM drive (necessary for initial installations and setup). An aside: it might be interesting to have a Linux-based embedded PC (IMEP already has a LabVIEW for Linux license) if a stable driver for the DIO-6533 appears.

It was also necessary to design a new interface between the DIO-6533 and the JUMBO backplane; this effectively replaces the VLK module of the IPH system, which was the interface between the JUMBO backplane, the MAHOUT CPU, and the JVM. I refer to this card as the “BAILEY Link,” or BLK module.

Together, the data acquisition computer and the embedded computer form an ethernet network with two members. This assures that the data transfers are not interrupted by extraneous network traffic.

One drawback of the DIO-6533 is that the drivers (from the NIDAQ package) are only provided for Windows and Macintosh operating systems. Furthermore, register-level programming is not supported, which is to say, it is impossible (or at least, I was unable) to

get enough information to write drivers for a different operating system (such as Linux or VxWorks).

Given this constraint, and given the choice of Windows NT for the resident operating system, we could choose between LabVIEW or Microsoft Visual C++ (MVC++), the two Windows-based development systems available at IMEP, for the software platform. Because of my personal familiarity with LabVIEW, I chose to work with it, in order to shorten the development time. Specifically, the BARNUM software was developed using LabVIEW 5.1 (no data acquisition package required) and BAILEY was developed using LabVIEW 5.1 and NiDAQ 6.6.

It turns out that for time-critical routines, specifically the data compression, LabVIEW is too slow. For these routines, I used MVC++ with LabVIEW CINs (Code Interface Nodes).

*** IMPORTANT NOTE ***

The CSSIPC-PIIC is apparently not compatible with Windows NT Service Pack 5. Presently, Windows NT 4.0 with Service Pack 4 is installed. New Service Packs should not be installed before first contacting the manufacturer:

CSS Industrie Computer GmbH
Osterhofstr. 14
D-87600 Kaufbeuren
tel: +49 (0)8341 95137-0
fax: +49 (0)8341 95137-20
email: support@css-ic.de
web: www.css-ic.de

## Cost Comparison

The new design not only simplifies the Command Box design, but is also significantly less expensive:

| IPH system | Cost (Euro) | New System | Cost (Euro) |
|---|---|---|---|
| BIT-3 VME interface | 2180 | - | - |
| BIT-3 fiber-optics | 3650 | 10-base-T ethernet cable | 80 |
| VME crate | 2200 | AT Power Supply | 25 |
| JVM Module | 2200 | DIO-6533 | 1670 |
| GESPAC CPU | 655 | CSS Pentium II | 1950 |
| GESPAC Backplane | 435 | CompactPCI backplane | 240 |
| Serial Interface | 365 | - | - |
| - | - | Hard drive, CR-ROM | 190 |
| **TOTAL** | **11685** | | **4155** |

The above comparison demonstrates that the new design can replace a significant portion of the IPH system at one-third of the cost. Note that the remaining components of a complete Command Box system (JUMBO and the control PC) cost approximately 6000 Euros.

## Hardware

Many changes were necessary to the hardware in order to implement the new design. The major change is the replacement of the VLK module with the BLK module, but there are also numerous changes to the EPLD programs of several modules, and a few changes to circuit paths are also required.

The following modules are used in JUMBO4 (Command Box version 4):

PWC Switches and monitors all JUMBO and Readout Box power supplies
BLK provides an electrically isolated connection to the DIO6533 module of BAILEY
CTL most important function is to provide the REQ signal to the DIO6533

LOC controls the CCD selection and JUMBO commands in local or remote mode

MCL switches a clock signal on to the Command Bus, in order to “wake up” the ISO150 digital couplers on the BLK board (not the original function)

CLK provides clock signals

LPH provides the CCD line count

RPH provides the CCD row count

ADC multiplexes and digitizes the CCD signals

RLK provides the power and control signals to the Readout Box RDO modules

PLK provides the power and control signals to the Readout Box DRV modules

The following modules are not implemented in Command Box 4:

MON the temperature control module

DAC the digital to analog converter

HLK the heater link (also part of temperature control)

VLK the JUMBO/MAHOUT link, replace by BLK

## Software

The software structure is a client/server architecture using the TCP/IP protocol. The server program, called BAILEY, runs on the embedded PC, and the client program, BARNUM, runs on the data acquisition computer. Both programs were developed under LabVIEW and then promoted to stand-alone executables using the LabVIEW Application Builder.

(Concerning the program names: JUMBO is symbolized by an elephant, which reminded me of the circus, and the “Ringling Bros./Barnum & Bailey Circus” is a famous circus in the United States.)

**BARNUM**

The functionality of the old JUMBOctrl program is more or less reproduced on the BARNUM control panel. Instead of sending serial commands, BARNUM sends TCP/IP messages to the server program (BAILEY). The BAILEY routine is mostly a TCP/IP listener; it does little or nothing until it receives a request from the BARNUM routine.

When the BARNUM routine is started, the message “Searching for BAILEY...” is displayed. BARNUM then sends the CHECK command to BAILEY. If BAILEY does not respond, the BARNUM routine will terminate and display the message “Bailey not responding. (Most likely cause: not booted).” In fact, there are a few possibilities: The BAILEY CPU might not be booted, the program itself may not have started or may be hung, the network connection could be bad, or the **IP Address of BAILEY** field could be inconsistent with BAILEY’s true IP address.

It can be the case that BAILEY responds to the CHECK command, but it informs BARNUM that it was unable to establish a link to JUMBO, in which case BARNUM will terminate and the message “Bailey was unable to start. (Most likely cause: JUMBO off)” will be displayed. This could mean that either JUMBO has not been powered on, JUMBO is in local mode, the cable connection between BAILEY and JUMBO is bad, or that there is a more serious hardware failure.

Assuming the response to the CHECK command is error free, BARNUM then sends the INIT command, followed by the CCD type. As a result, in order to change the CCD type, one must first stop BARNUM, select the correct type in the **CCD Type** field of the Expert Panel, and then restart BARNUM. In principle, it is not possible to select an unknown CCD type, but if for some reason the type is not recognized, BARNUM will terminate with the message "Unknown CCD type." Note that it is also necessary that the CCD type be compatible with the LPH and RPH modules of JUMBO, and there is no way to check this with software. If these are not compatible, you may receive incomplete or distorted CCD images or unknown errors during the READ command. At this point, the VME OK LED will turn on, indicating that the indirect connection between BARNUM and JUMBO has been established.

Finally, BARNUM reads the entries of the **Save Parameters** fields from the file "brn.dat". This file must exist in the directory "c:\barnum\data", or else an error will occur. Then the main execution loop begins, in which BARNUM awaits a change in a panel control.

The various panel areas and individual controls are described in the following subsections:

Save Parameters

The **Directory** field establishes the directory in which to write data, *as seen by BAILEY*. For example, "d:\CCD Data" is a directory on the hard disk inside BAILEY, whereas "\\pcdearg\C$\CCD Data" is a network directory. The directory information is sent to BAILEY with every READ command. If the directory is invalid, BAILEY will return a error and the READ will not be executed.

The file name of the run will be constructed from the **Run Prefix** and **Run Number** fields. For example, if **Run Prefix** is set to "TEST" and **Run Number** is set to "237" then the corresponding file name is TEST0237.cd*n*, where *n* is one of the selected CCDs (as with the IPH system, the letters A-H corresponds to the numbers 1-8). In order to remain compatible with the IPH analysis software, the DOS naming convention is partially enforced, that is, BARNUM attempts to produce filenames of only 8 characters. This attempt will not succeed if the prefix is too long or if the run number is too large.

The current save parameters will be written to a file when one exits the BARNUM routine using the **STOP BARNUM** button (on the Expert Panel).

Select CCDs

These toggle switches mirror those on the JUMBO front panel. The CCD selection is sent to BAILEY with each CLEAR and READ command. BAILEY first writes the selection, then reads back what it has written. If this is inconsistent, an error is return and the command is aborted. Note that the CCD selection only influences which CCD data will be written to a file; the clear or read time is the same, whether eight or two CCDs are selected.

Read CCDs

First, BARNUM sends the READ command to BAILEY. If BAILEY receives and understands the command, it then queries the JUMBO status bus, to make sure that JUMBO is not in local mode, that a clear is not in progress, etc. If no error is returned, BARNUM then sends the CCD selection. If no error is returned, then BARNUM sends the desired data directory. If the directory is valid (as seen by BAILEY), then no error is returned, and BARNUM sends the file name. [Note: Since the file name is derived from the Save Parameters panel, this name is assumed to be valid, and there is no error checking. As long as the **Run Prefix** entry is not too long and does not include strange characters, this should work, otherwise BAILEY might hang with an “invalid filename” error.] BARNUM then waits for a message indicating that the read is complete (i.e. the LEDs on JUMBO go from green to yellow). BARNUM then waits again for a message indicating that the data was successfully written to the desired directory. WARNING: if data files with the same name already exist, they will be overwritten without warning (as with the JUMBOctrl system). If all these steps are completed without errors, BARNUM gives the message “Single Read complete.” The **Run Number** field is automatically incremented at the end of a successful read sequence.

Clear CCDs

BARNUM sends the CLEAR command to BAILEY. If BAILEY receives and understands the command, it then queries the JUMBO status bus, to make sure that JUMBO is not in local mode, that a clear is not in progress, etc. If no error is returned, then BARNUM waits for an

indication that the CLEAR was successfully completed, while displaying the message "Clearing CCDs...". This process is repeated **Number of Clears** times, with an appropriate message displayed at the end of each cycle (i.e. "CCDs cleared 1 time.", "CCDs cleared 2 times.", etc.).

Series Readout

When the **SERIES** button is pressed, the "Read CCDs" sequence described above is executed **Number of exposures** times, once every **Exposure Time** seconds. The **Run Number** field is automatically incremented at the end of each exposure. The present exposure number is displayed in the **Exposure #** field. During an exposure, the **Time Left** field displays the countdown and the message "Run sequence in progress..." is displayed. During this time, no other panel controls will respond except for the **stop series** button. The user can, however, press as many buttons as he/she wishes, which will cause a number of commands to be executed after the run sequence is complete. This could lead to unexpected consequences and errors, and it is not recommended.

I note that the first run of a run sequence has an exposure time equal to the "Exposure Time" plus the "Read/Write Time" [$t_{RW}$, the time required to read the CCDs and write the data files (about 30 seconds for one-half CCD-05 or the CCD-22)], whereas subsequent runs have an exposure time equal to MAX("Exposure Time", $t_{RW}$). This "feature" exists in the old JUMBOctrl program as well, and since we have been running this way for many years, I decided to leave it as it was. Nevertheless, when one wishes to calculate the exact exposure time of a run sequence, one must take this effect into account.

Message Window

Status and error messages are printed in this display.

Expert Panel

In order to establish a TCP/IP link to BAILEY, BARNUM requires both an IP address and a port number. The **IP Address of BAILEY** field must be set to the same IP address set in BAILEY's network control panel. The port number 2048 is hard-wired into both programs.

The **CCD Type** field controls the number of pixels to be read out and hence the initialization of the DIO-6533 data transfer by BAILEY. The pixel count must be consistent with the hardware of JUMBO, that is, the LPH and RPH boards, or else unexpected errors will occur.

The **reset JUMBO** button sends the RESET command to BAILEY, which first checks if JUMBO is in local mode. If not, the reset bit is asserted on the JUMBO backplane for one second. This may be useful for recovering from some errors, such as when a read refuses to terminate.

The **change mode** button toggles between the test mode and normal operation. During test mode, data is not written to the BLK from the JUMBO ADC, but rather from the line counters. This yields a nice black-to-blue gradient as a CCD image. It can happen that the **Test Mode** indicator does not accurately reflect the status of BAILEY, in which case an error message will be given when the **change mode** button is pressed, after which the correct mode will be indicated.

The **STOP BARNUM** button (surprisingly) halts the execution of BARNUM and writes the contents of the **Save Parameters** field to the text file “brn.dat”.

**BAILEY**

BAILEY is started automatically when the embedded PC is booted, albeit in a somewhat indirect manner. First, the login normally required by an NT system has been bypassed, by modifying the Windows Registry files. We then discovered that one cannot place the BAILEY routine directly in the Startup Menu, because such programs are started before important components of the operating system are in place (a “feature” of Windows NT). Therefore, a simple program called BaileyLauch was instead placed in the Startup Menu; this program executes during the boot sequence, and waits approximately twenty seconds before calling the BAILEY routine and then closing. The entire cold-start boot sequence requires just over two minutes (two minutes and seven seconds under normal circumstances).

When the BAILEY routine first starts, it checks the JUMBO status register, specifically, the JUMBO power-on bit. If this response fails, it indicates 1) the JUMBO module is not turned on, or 2) the cable between BAILEY and JUMBO is not connected, or 3) a more serious failure. In such a case, BAILEY defaults to a TCP/IP Listen mode, where it waits for a CHECK command from BARNUM, so that it can tell BARNUM something is wrong. Every thirty seconds, BAILEY again tries to establish the connection to JUMBO (thus giving the user a chance to turn on the power, connect the cable, etc.).

Once it is established that the BAILEY/JUMBO link is active, BAILEY writes a pattern to the CCD selection LEDs on JUMBO (copying the behavior of JUMBOctrl), finally deselecting all CCDs (no LEDs lit) and extinguishing the VME OK LED (if it was lit).

BAILEY then enters the main execution loop, where it functions as a TCP/IP listener, and executes the commands received from BARNUM. There are seven commands issued by BARNUM and handled by BAILEY:

INIT

This command initiates contact with BAILEY. First the JUMBO status is tested; if a problem exists, an error is returned. If the accumulator power is not on (either the toggle is not switched or the batteries are not charged) a warning message is given, but operation continues. BAILEY then reads the CCD type (and thereby calculates the size of the data buffers), and resets JUMBO.

CLEAR

First the JUMBO status is tested; if a problem exists, an error is returned. BAILEY reads the CCD selection, selects the desired CCDs, then checks to see if the status matches the selection. If not, returns an error. BAILEY waits for the clearing to finish before giving a successful return code.

READ

First the JUMBO status is tested; if a problem exists, an error is returned. BAILEY reads the

CCD selection, selects the desired CCDs, then checks to see if the status matches the selection. If not, returns an error. Then the desired data directory is read, and BAILEY checks to see that the directory exits; if not, an error is returned. Otherwise, the file prefix for the readout is read (no error checking), and then the READ command is sent to the JUMBO command bus. This initiates a pattern-generation type data transfer on the DIO-6533. (This means that timing is controlled entirely by JUMBO, which sends a signal when to latch data from the ADC bus. JUMBO does not wait for a ready signal, so this is not a true handshaking mode.) When the readout is complete, the data are compressed and the compressed files are written directly to the specified directory (network or local). The compressed files have the same format as those written by the IPH system, which means that IPH software (CCDView, Pion, and Hydrogen) and the new Tokyo software can also be used to analyze this data.

STEST

Sets the JUMBO test mode. Returns an error if the test mode is already set.

CTEST

Clears the JUMBO test mode. Returns an error if the test mode is not set.

RESET

Checks to be sure that JUMBO is not in local mode. If not, asserts the master reset signal on the JUMBO bus for one second.

CHECK

Checks whether JUMBO is responding to commands by testing the status bus.

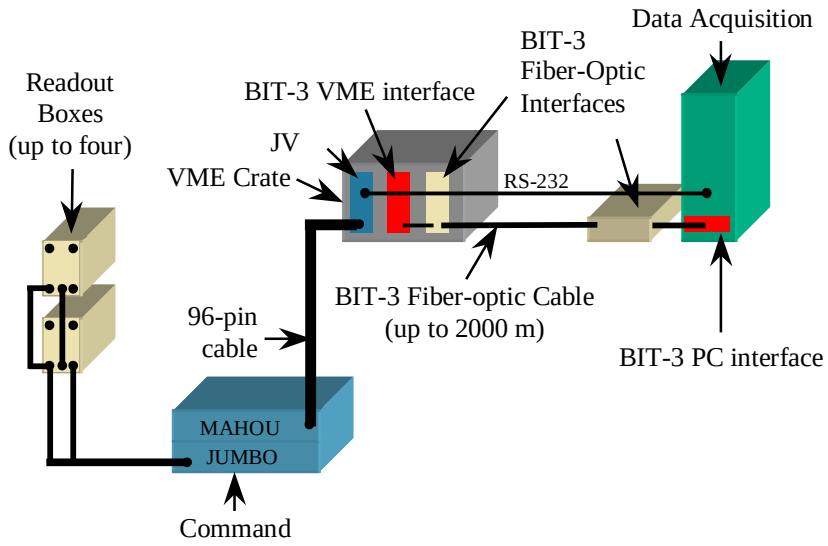
Note that the above commands are all sent as five-character command strings, so, for example, the READ command actually consists of the string “READ(sp)”.

BAILEY is of course designed to operate without keyboard, mouse, or monitor; all control is asserted over the ethernet connection. However, it is possible to attach all of these devices for debugging purposes. It is necessary to shutdown, attach the external devices, and reboot in order to do this. All aspects of a normal Windows NT system are also available when the keyboard and monitor are attached. This is useful for system updates, network settings, backups, etc.

The BAILEY control panel is actually mostly a display panel. There is only one control, the STOP button, and this button is of little use in the stand-alone version of the program. It is nevertheless essential for debugging purposes, because only by executing the stop command will the DIO-6533 resources be de-allocated, which then allows re-starting of the program. The only other way to de-allocate these resources is to remove power from the Compact PCI bus.

The various displays of the BAILEY panel are useful, in that one can see the most recent command sent, the directory and file name sent, and the status of the test mode and the JUMBO link.

## A) IPH System




## B) New Design

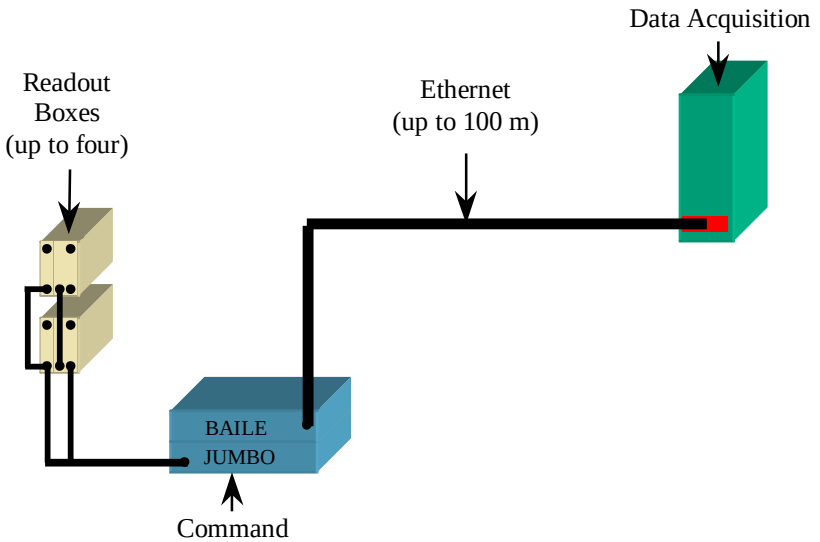



Figure 1