DRAFT August 18, 1998

The BLIP card Driver
Steve Eichblatt, Fermilab
Steve Furlanetto, Fermilab

1 Introduction

This document describes the basic functionality of the BLIP card, and the software tools that perform them. This can be found at
`http://fnphyx-www.fnal.gov/experiments/cdms/software/GPIB/BLIPboard/BLIPdriver.ps`.

2 Design

The BLIP driver performs in conjunction with the `GPIB protocol`. See ref [1] for a detailed description. The basic idea is that commands are issued from a perl script and are sent *via* tcp/ip to a server which issues the GPIB commands to perform the action on the card.

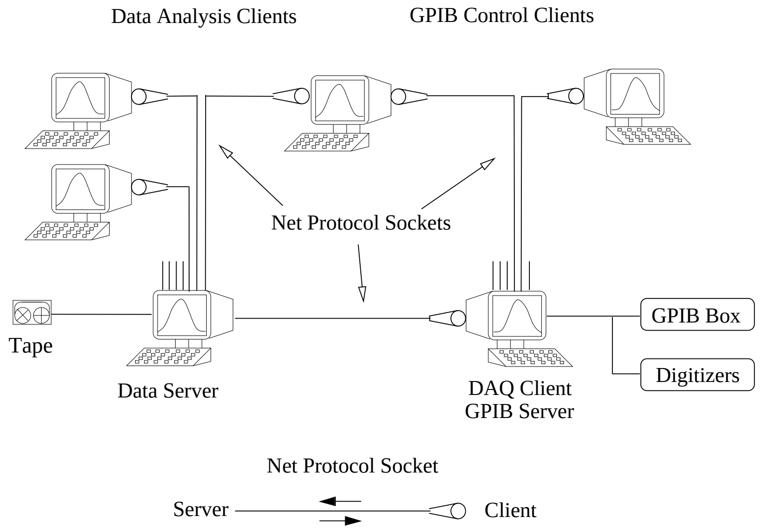

Figure 1: *A possible configuration of machines connected by the netProtocol*

3 Instructions

The following sections list the methods that apply to instances of the BLIPcard class. The program is designed to use the `rack.pl` interface. Instructions for use of this interface can be found in ref citeinterface.

This software also contains a useful persistence feature. Upon exiting the program, it will record the card status in a file called `/tmp/hostname-BLIPcard-subrack-module.out`, where `subrack` is the subrack of the card and `module` is the module number of the card. If the `-s` switch is used, `hostname` becomes `script`. When `rack.pl` is run again, it will automatically search for this file and reload those values into memory. This feature is designed to ensure that the hardware and software remain synchronized. If the rack is powered down, be sure to re-initialize the settings when the program is run again.

3.1 General Information

help (**String** `topic`)
prints the subset of the commands that can be issued to the BLIP card which contain the string `topic`. If no argument, prints all the help information.

print (**void**)
prints Information about the GPIB box and the card including module number and the status bits of the Control Registers.

cardinfo (**void**)
Reads back the type of card, serial number of the card, and the version number of the card.

cardtype (**void**)
Prints card type that computer believes card to be.

saveScript (**String** `filename`, **int** `overwrite`)
Saves all subsequent GPIB commands to `filename`. Commands are saved in format readable by LABView interface and Perl script reading routine. The `overwrite` field is optional (if omitted, and relevant, user will be prompted for it). The possible choices are: `1` to append to existing `filename`, `2` to overwrite `filename`, or `3` to cancel if `filename` exists.

endScript (**void**)
Ceases saving commands to a file.

enableBox (**void**)
Enables communication with GPIB box. Note that this only works for current box.

disableBox (**void**)
Disables communication with GPIB box. Note that this only works for current box.

accumulate (**void**)
Creates a queue of commands for the GPIB box. Will not execute them until "execute" command is given; then it will do them all at once.

unaccumulate (**void**)
Turns off accumulation feature. Note that the remaining items in the queue are NOT executed.

execute (**void**)
Executes all commands in the GPIB box queue.

powerUp (**void**)
Resets control registers and DACs to the values that are believed by the software to be true. Designed for use when power is shut off to board without reloading program.

toFile (**String** `filename`)
Saves current card settings to file `filename`.

fromFile (**String** `filename`)
Reads card settings from `filename` and sets registers and DACs to those settings.

init (**void**)
Resets registers and DACs to initial values (DACs set at zero, filters selected to be 0, and all channels disabled).

3.2 Setting up the DAC's

QODAC (**float** `Voltage`)
sets the Outer Charge Bias DAC to `Voltage` Volts.

QIDAC (**float** `Voltage`)
sets the Inner Charge Bias DAC to `Voltage` Volts.

PS1DAC (**float** `Voltage`)
sets the Phonon Sensor 1 DAC to `Voltage` Volts.

PS2DAC (**float** `Voltage`)
sets the Phonon Sensor 2 DAC to `Voltage` Volts.

ImplantDAC (**float** `Voltage`)
sets the IBAPACAP's Implant DAC to `Voltage` Volts.

LEDDAC (**float** `Voltage`)
sets the LED DAC to `Voltage` Volts.

CinnerDAC (**float** `Voltage`)
sets the IBAPACAP's Inner Charge DAC to `Voltage` Volts.

CouterDAC (**float** `Voltage`)
sets the IBAPACAP's Outer Charge DAC to `Voltage` Volts.

3.3 Setting up the Multiplexers

Mux1/2 (**String** `whichMux`)
sets Multiplexer 1/2 to output its `whichMux`'th output. `whichMux` is one of: "vamp1","vamp2","QoImon","QoVmon", "QiImon","QiVmon","LEDV","LEDI","PS1Bias","PS2Bias", "TSense", or "FETHtr".

Mux1/2 (**void**)
Returns the index of which output is currently selected on multiplexer 1/2.

3.4 Setting up the Control Registers

3.4.1 Charge & LED Control Register

gndQI/O (**int** `gndstate=1`)
ground/unground the Inner/Outer Charge Bias. By default, `gndstate`=1, which grounds the bias. With `gndstate`=0, the bias voltage is applied.

gndPS1/2 (**int** `gndstate=1`)
ground/unground the Phonon Sensor 1/2 Bias. By default, `gndstate`=1, which grounds the bias. With `gndstate`=0, the bias voltage is applied.

Qi/oVmon (**void**)
Monitor Voltage of inner/outer Charge channel.

Qi/oImon (**void**)
Monitor current of inner/outer Charge channel.

Qpol (**int** `qpol=0`)
sets the polarity of the QAmps. Default=0 (positive).

LEDPulse (**int** `ntimes=1`)
Pulse the LED `ntimes` times.

LEDOn/Off (**void**)
Turn on/off the LED.

LEDplus (**void**)
Select the positive LED leg.

LEDminus (**void**)
Select the negative LED leg.

3.4.2 Phonon Sensor Control Registers

divPS1/2 (**int** `divide=0`)
selects/deselects the divider in Phonon Sensor 1/2. By default, `divide`=0, which does *not* divide the output.

varPS1/2 (**int** `variable=0`)
selects/deselects the variable input to Phonon Sensor 1/2. By default, `variable`=0, which deselects the variable input.

filterPS1/2 (**int** `select=1`)
selects/deselects the low pass filter in Phonon Sensor 1/2. By default, `select` is such that the filter is selected.

refPS1/2 (**int** `which=1`)
selects which reference to use for the modulation of the Phonon Sensor Bias.
0 = 1 kHz
1 = 10 Hz
2 = +15 V
3 = -15 V

tenHz (**void**)
selects the 10 Hz input to Phonon Sensors 1 & 2. Also selects the low-pass filter on both channels.

kHz (**void**)
selects the 1 kHz input to Phonon Sensors 1 & 2. Also selects the low-pass filter on both channels.

3.4.3 Lockin and Vamp Control Register

vampFast (**int** `fast`=1)
sets the fast/slow Voltage amp setting. If `fast`=0, set to slow mode.

vampSlow (**void**)
sets the slow Voltage amp setting. Defined as `$self->vampFast(0);`.

azFast (**int** `fast`=1)
sets the fast/slow autozero on the lockin amps. If `fast`=0, use the slow averaging, to get higher quality DC offset subtraction. If `fast` =1 (default), use the fast averaging.

azSlow (**void**)
sets the autozero to the slow mode.

bypass (**int** `bypass`=1)
bypasses or not bypasses the lockins (both 1 and 2). If `bypass`=0, the lockins are not bypassed.

3.4.4 IBAPACAP Control Register

coPulse (**void**)

ciPulse (**void**)
Pulse inner/outer charge on IBAPACAP.

coRange (**void**)

ciRange (**void**)
Sets the inner/outer charge range on IBAPACAP.

pPulse (**void**)
Simulate a phonon pulse on IBAPACAP.

pRange (**void**)
Set the phonon range.

trigDelay (**void**)
Set the trigger Delay.

References

[1] Steve Eichblatt, Sae Woo Nam CDMS note 9712002. *The CDMS Net Protocol*

[2] Steve Eichblatt, Steve Furlanetto *General Information on the Perl-based Electronics Interface*