VMEDAQ

发布 *2019.12.28*

Hongyi Wu(吴鸿毅)

2019 年 12 月 29 日

content:

感谢您选择 VMEDAQ。更多开源程序，请访问我们的 Github 和 吴鸿毅的 Github

CHAPTER 1

简介

本程序为 **北京大学实验核物理组**当前使用的 VME 获取。

该获取基于 RIKEN 的获取发展而来。我们已经对原本程序进行较大的修改。如果使用本程序，请严格使用本程序包内程序，请勿随意升级/替换程序包内部程序/固件。

我们将尽快完善本说明书。

1.1 版本

我们建议用户下载稳定版本

1.1.1 稳定版本

稳定版本 2018.12.28

下载最新版本，请点击: VMEDAQ stable

网页版说明书请访问: 稳定版说明书

1.1.2 准预览版本

准预览版本 2018.12.29

程序下载请访问: VMEDAQ

网页版说明书请访问: 说明书

1.2 关于

本程序历史维护：

- 李智焕
- 李晶
- 臧宏亮
- 吴鸿毅 (wuhongyi@qq.com / wuhongyi@pku.edu.cn)

说明书贡献者（按姓名笔画排序）：

- 王东玺（中国原子能科学研究院）
- 孙立杰（中国原子能科学研究院）
- 吴鸿毅

1.3 性能介绍

- 本获取经过 Scientific Linux 6/7 系统测试。
- 支持多个机箱同步获取。将插件分散在多个机箱，可大大减少数据传输的死时间。
- 本获取分软件 busy 跟硬件 busy 两种模式。
- **对软件 busy 模式**
  - 该模式下，一个事件的死时间由 trigger 门宽，7 us 左右模数转换时间，20 us 数据传输中断请求及数据传输时间组成。其中除了数据传输时间，其它三个时间是固定的，大约为 30 us。
  - 限制该模式下计数率的因素为数据传输时间，数据越大，所需传输时间也就越长。
  - 以一个机箱，300-500 路左右输入为例，平均 10000 个触发能够记录 5000-6000 个事件，效率在 50-60%
  - 如果以两三个插件为例，则能够达到 70%+ 以上
- **对硬件 busy 模式**
  - 该模下式，一个事件的死时间由 trigger 门宽，7 us 左右模数转换时间两部分组成。
  - 意味着该模式下一个事件的死时间大约在 11 us 左右。
  - 该模式模数转换及数据传输同步进行，因而数据高速传输产生的高频信号会对前放/主放的信号带来微小的影响。
  - 通过适当抬高阈值可消除该影响。
  - 该模式下获取效率极高，平均 10000 个触发能够记录 9000+ 个事件，效率达到 90%

1.4 目录

文件夹内有以下文件/文件夹：

- analysis（一些用来辅助分析的代码）
- anaroot（底层库，用来将原始数据转为 ROOT 及在线统计）

- checkcnt（自动检查数据事件关联情况）
- cutpedo（自动拟合推荐合适 pedestal）
- DAQConfig（获取控制包）
- firmware（固件）
- httponline（基于网页的在线监视）
- online（在线监视能量，能谱）
- r2root（数据转换）
- source（babirl 源码，将会配置自动化安装脚本）
- statistics(时时监视每路信号的计数率，每 10 ns 更新一次)
- README.rst（本文件）
- docs(网页版说明书)
- README（rst 版说明书）
- README.pdf（pdf 说明书）

CHAPTER 2

软件安装

本页面安装软件放置在 source 文件夹内，里面包括 **获取驱动、依赖库等以及自动安装脚本**。

2.1 系统要求

本获取经过 Scientific Linux 6/7 系统测试。建议采用 CentOS 6/7 或者 Scientific Linux 6/7。

本获取要求 CERN ROOT 5/6，建议优先选择 ROOT 6。

如果没有合适的系统，可参考我们的获取系统安装 Install Scientific 7 。安装好系统之后，还需要对基础依赖工具做一些安装及升级，可以下载执行 自动化安装脚本 自动配置或者按照教程手动安装。

2.2 CAEN Lib

本程序依赖 CAENVMELib/CAENComm/CAENUpgrader 三个库文件。

其中 CAENVMELib/CAENComm 为获取运行必须的库。CAENUpgrader 用来更新固件。

进入 source 文件夹内，在 ROOT 权限下执行 setup.sh 脚本，将会自动安装以上三个依赖库。

```
# 在 source 文件夹内，ROOT 权限下执行以下命令
sh setup.sh    # 需要 ROOT 权限
```

2.3 检查 CAENVMELib 安装

进入 CheckRegisterToolByV2718 文件夹，make 编译里面程序，如果生成一个名为 pku 的可执行文件，则软件安装成功。

```
cd CheckRegisterToolByV2718
make
```

2.4 检查 CAENUpgrader 安装

安装后在终端中输入

```
CAENUpgraderGUI
```

将会弹出 CAEN Upgrader GUI 的图形界面。

2.5 V1718

如果您使用 V1718，则需要安装 USB 驱动。

```
tar -xzvf CAENUSBdrvB-1.5.2.tgz
cd CAENUSBdrvB-1.5.2
make
make install      # 需要 ROOT 权限
```

2.6 A2818 驱动

如果您使用 A2818，则安装以下驱动。

```
# A2818Drv-1.20-build20161118.tgz
# 将该文件夹复制到 /opt 并安装在该位置
tar -zxvf A2818Drv-1.20-build20161118.tgz
cp -r A2818Drv-1.20 /opt            # 需要 ROOT 权限
cd /opt/A2818Drv-1.20               # 需要 ROOT 权限
cp ./Makefile.2.6-3.x Makefile      # 需要 ROOT 权限
make                                # 需要 ROOT 权限

# 设置开机自动执行该脚本
# 在文件 /etc/rc.d/rc.local 中添加以下一行内容
/bin/sh /opt/A2818Drv-1.20/a2818_load
# 或者在开启电脑之后执行以上命令
```

重启机箱后，在终端内输入 **dmesg|grep a2818** 将会看到以下的 A2818 驱动加载信息

```
a2818: CAEN A2818 CONET controller driver v1.20s
a2818:   Copyright 2004, CAEN SpA
pci 0000:05:02.0: enabling device (0000 -> 0003)
pci 0000:05:02.0: PCI INT A -> GSI 19 (level, low) -> IRQ 19
a2818: found A2818 adapter at iomem 0xf7800000 irq 0, PLX at 0xf7900000
a2818:   CAEN A2818 Loaded.
a2818:   CAEN A2818: 1 device(s) found.
```

2.7 A3818 驱动

如果您使用 A3818，则安装以下驱动。安装该驱动时，电脑机箱必须插入 A3818 卡，否则将会报安装失败。

```
tar -zxvf A3818Drv-1.6.1.tgz
cd A3818Drv-1.6.1
make
make install        # 需要 ROOT 权限
```

然后在终端内输入 dmesg 将会看到以下的 A3818 驱动加载信息

```
fuse init (API version 7.14)
CAEN A3818 PCI Express CONET2 controller driver v1.6.0s
  Copyright 2013, CAEN SpA
pci 0000:02:00.0: PCI INT A -> GSI 16 (level, low) -> IRQ 16
  alloc irq_desc for 33 on node -1
  alloc kstat_irqs on node -1
pci 0000:02:00.0: irq 33 for MSI/MSI-X
pci 0000:02:00.0: setting latency timer to 64
Found A3818 - Common BAR at iomem ffffc900067d4000 irq 0
Found A3818 with 1 link(s)
found A3818 Link 0 BAR at iomem ffffc900067d6000 irq 0
  CAEN A3818 Loaded.
  CAEN PCIe: 1 device(s) found.
```

2.8 RIKEN babirl

babirl 自动化安装方法

```
# 在个人用户目录下安装理研 babirl 库
# 在普通权限下执行以下脚本
sh autoinstallbabirl.sh
```

安装脚本会自动添加环境变量安装结束后查看.bashrc 文件，最后将多了三行如下内容

```
PATH=$PATH:/home/wuhongyi/babirl/bin/
export TARTSYS=/home/wuhongyi/VMEDAQ/anaroot
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$TARTSYS/lib:$TARTSYS/sources/Core
# 其中 wuhongyi 为电脑当前用户名
```

```
# 在 ROOT 权限下执行以下脚本
sh afterinstallbabirl.sh [user name]      # 需要 ROOT 权限

# 其中这里的 [user name] 换成你的帐号用户名，例如我的用户名为 wuhongyi
# sh afterinstallbabirl.sh wuhongyi
```

2.9 初始化 babicon

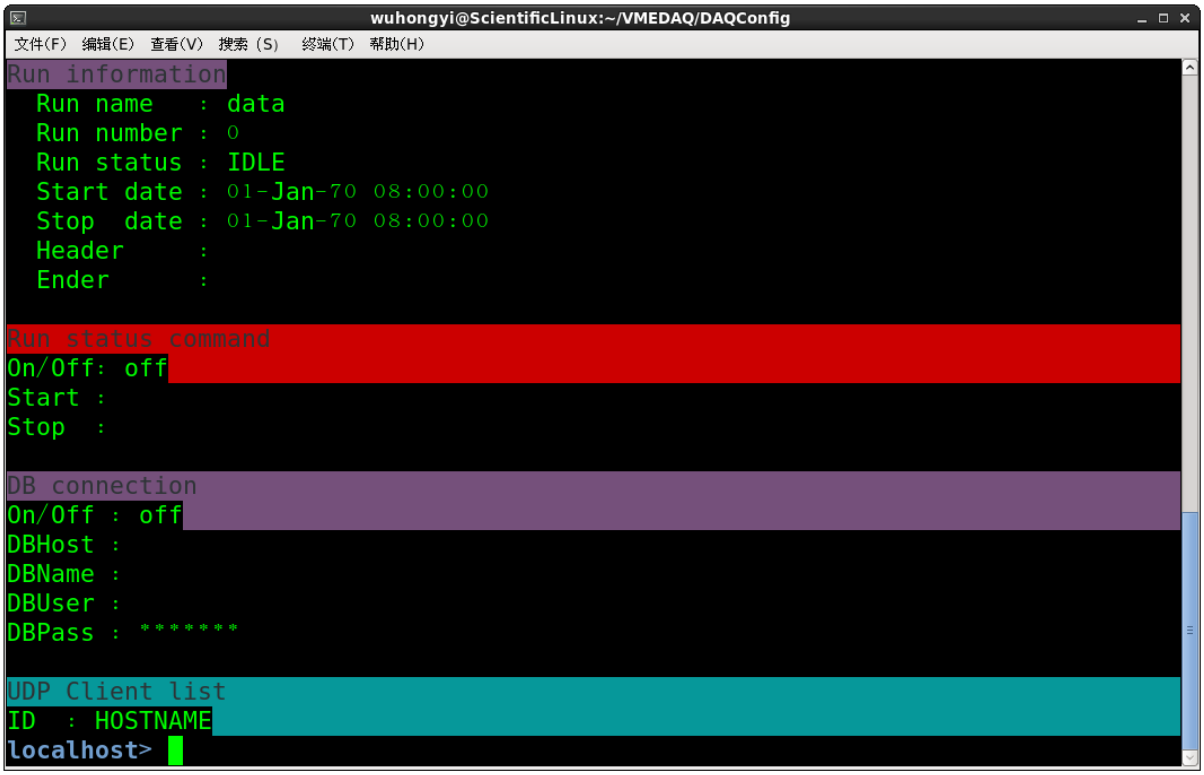
执行 DAQConfig 中的 StartDAQ.sh 开启进程

运行 babicon(安装后第一次需输入以下初始化)

新打开一个终端，然后输入

```
babicon
```

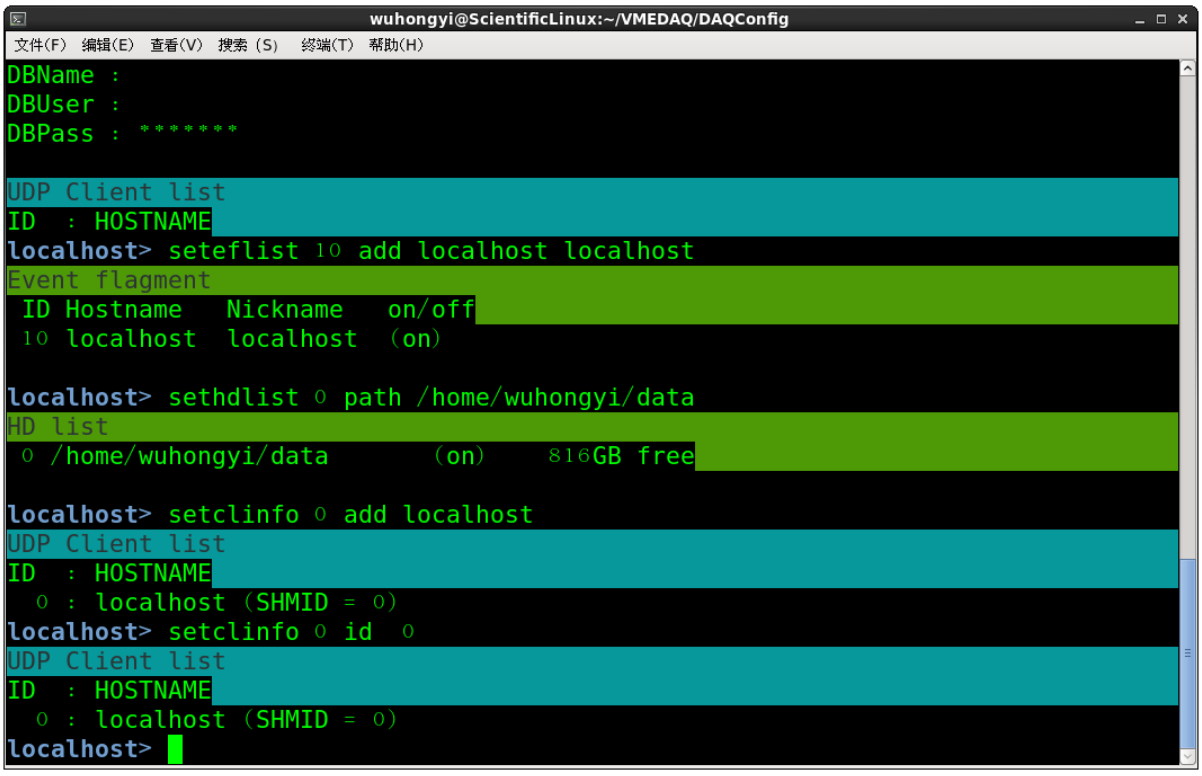
回车之后将看到以下界面



以下进行基本的变量设置

```
seteflist 10 add localhost localhost
sethdlist 0 path /home/wuhongyi/data     # 这里为数据存储路径
setclinfo 0 add localhost  #localhost 为本机器
setclinfo 0 id  0

# 如果设置给远程电脑
setclinfo 0 add [ip]    #[ip] 为接收端电脑 IP
setclinfo 0 id  0
```



2.10 防火墙设置

将共享数据发送到 Online 电脑，需要做以下设置或者关闭防火墙

对 Scientific Linux 6，终端 ROOT 权限下输入 **setup** ，选择 **防火墙配置**，去掉 **启用**。对 cientific Linux 7，ROOT 权限下终端输入以下信息关闭 firewall

```
systemctl stop firewalld.service # 停止 firewall
systemctl disable firewalld.service # 禁止 firewall 开机启动
firewall-cmd --state # 查看默认防火墙状态（关闭后显示 notrunning，开启后显示 running）
```

如果机器不联网，可以不需要开启以下 iptables 防火墙，反正不会被黑

```
# 在 /etc/sysconfig/iptables 添加以下一行（不能放到最后一行，其中 IP 替换为发送 DAQ 电脑的
IP）
-A INPUT -p udp -m state --state NEW -m udp --dport 17500:17510 -s 222.29.111.201 -
↪j ACCEPT
```

之后在 ROOT 权限下执行以下命令

```
systemctl restart iptables.service # 最后重启防火墙使配置生效
systemctl enable iptables.service # 设置防火墙开机启动
```

CHAPTER 3

固件要求

注意

- 请确保所使用的所有插件固件版本与以下一致。
- 我们尽可能及时更新保证采用较新/最新的固件。
- 由于新版本软件/固件我们需要经过大量评估测试，用户请不要随意升级我们未推荐的版本。

3.1 当前固件版本

```
V1718   2.14
V2718 FW CONET2 Compliant    2.14_1.5

A2818  新版的 CONET2 1.0  旧版的 CONET1 0.8
A3818   0.5
v1190   1.1
MADC32  0224
```

3.2 查看固件版本

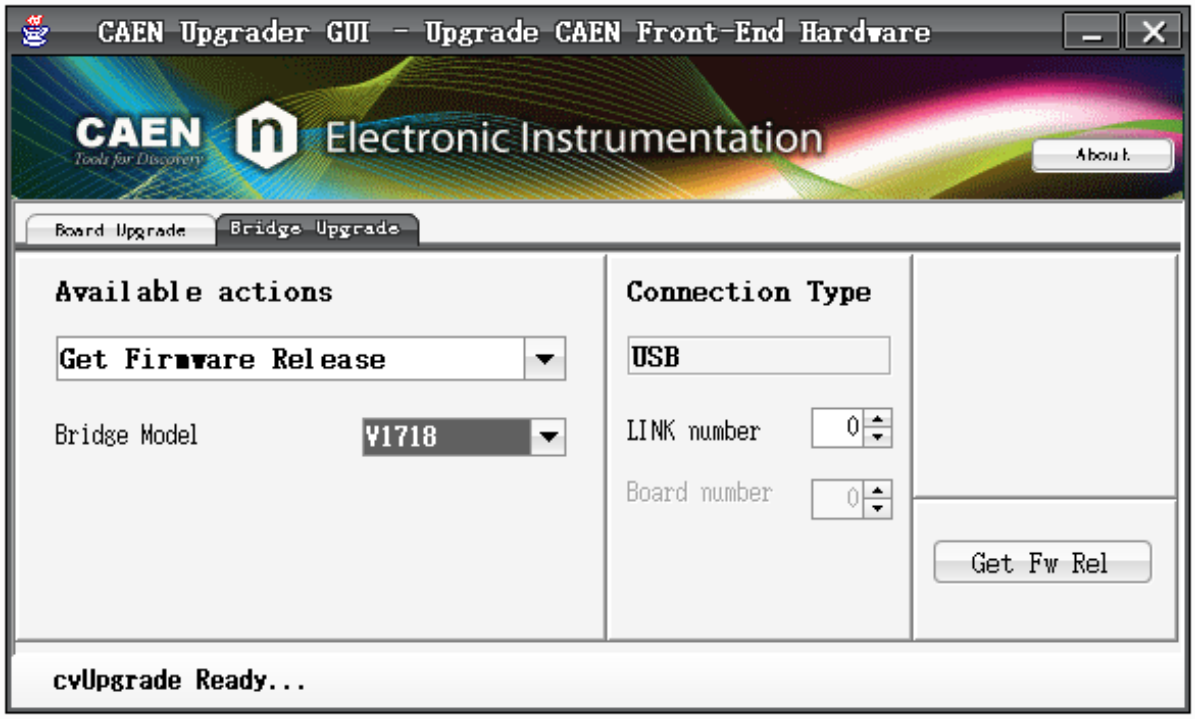
V1718/V2718/A2818/A3818 查看固件版本采用 CAENUpgraderGUI 程序，V1718/V2718/A2818/A3818/V1x90 升级固件版本同样采用 CAENUpgraderGUI 程序。即在终端中执行

```
CAENUpgraderGUI
```

升级固件时候，Browse 选择固件之后会弹出一个警告窗口，提示你“You have chosen to use a raw binary file”，点击确认，然后点击右下角的 Upgrade。等待升级结束，将会有一个窗口提示你重启。

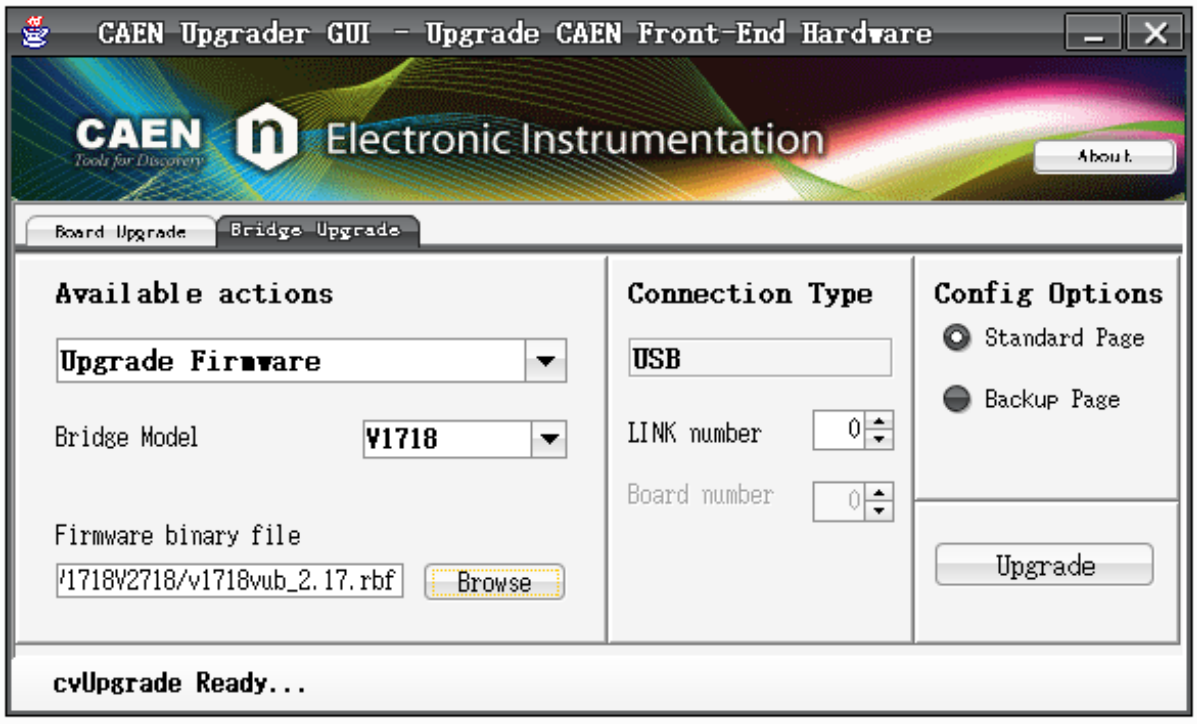
3.2.1 V1718

如下图，查看 V1718 的固件版本，点击 *Get Fw Rel* 按钮。



如果该固件版本不是 **当前固件版本**所列版本，则升级固件。

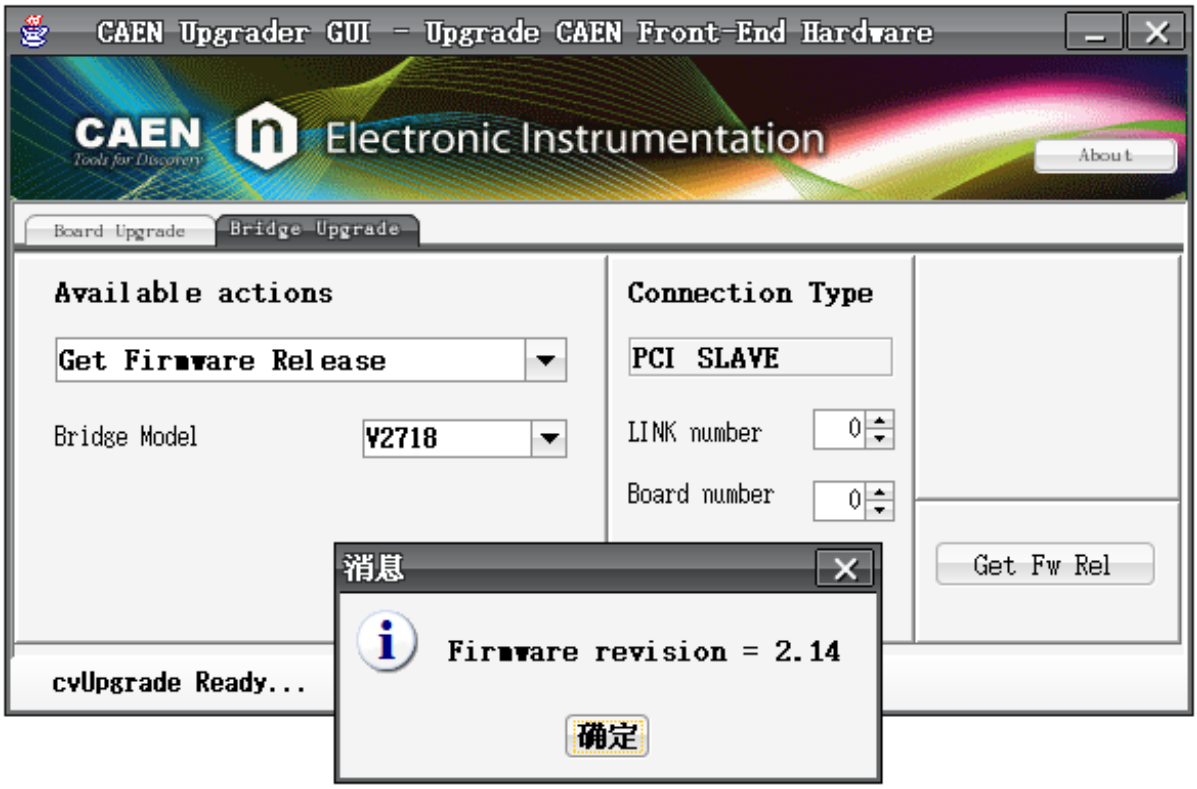
升级界面如下图所示：



3.2.2 V2718

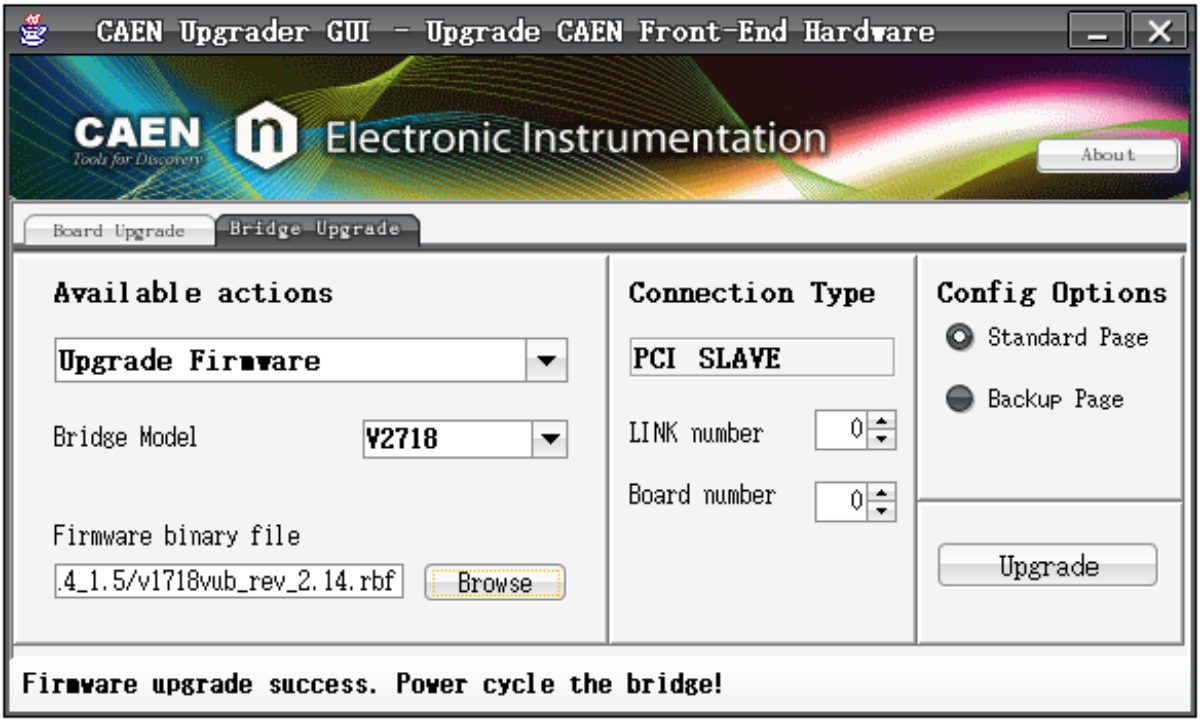
V2718 上固件包括主板 V2718 及子板上的 A2719。

如下图，查看 V2718 主板的固件版本，点击 *Get Fw Rel* 按钮。

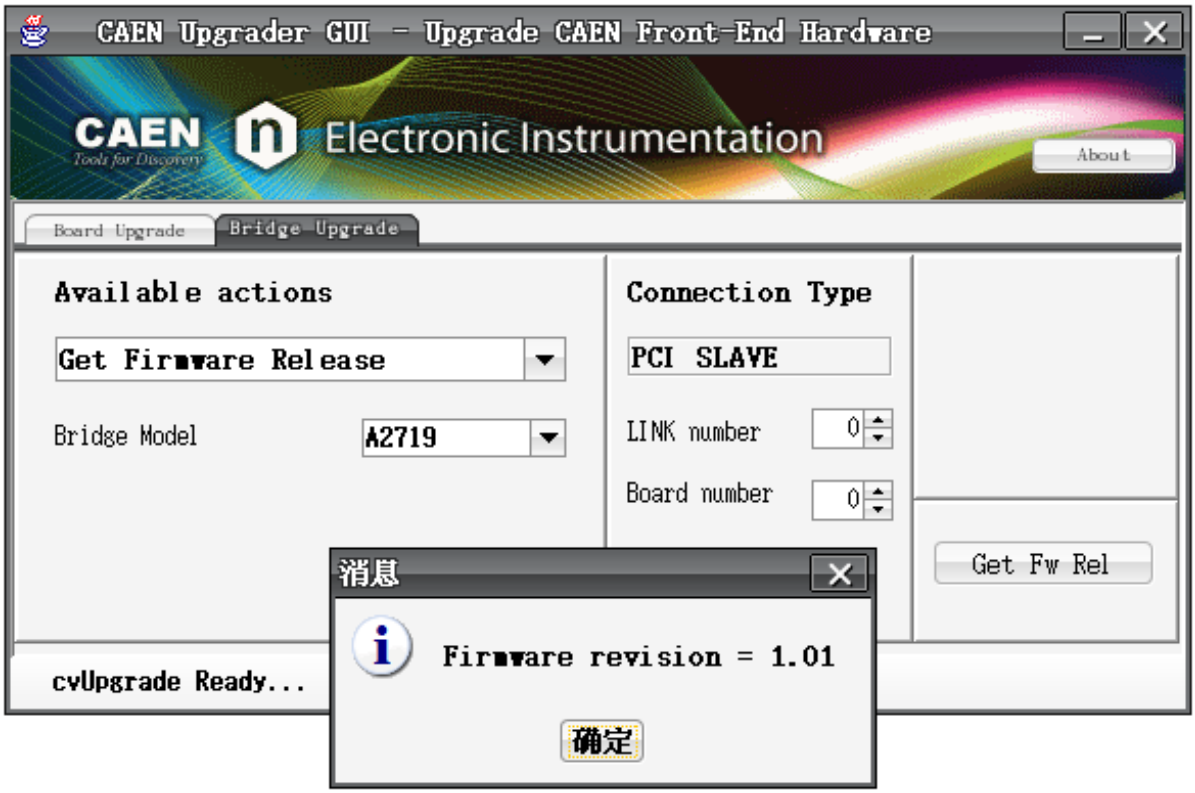


如果该固件版本不是 **当前固件版本**所列版本，则升级固件。

升级界面如下图所示：



如下图，查看子板 A2719 的固件版本，点击 *Get Fw Rel* 按钮。

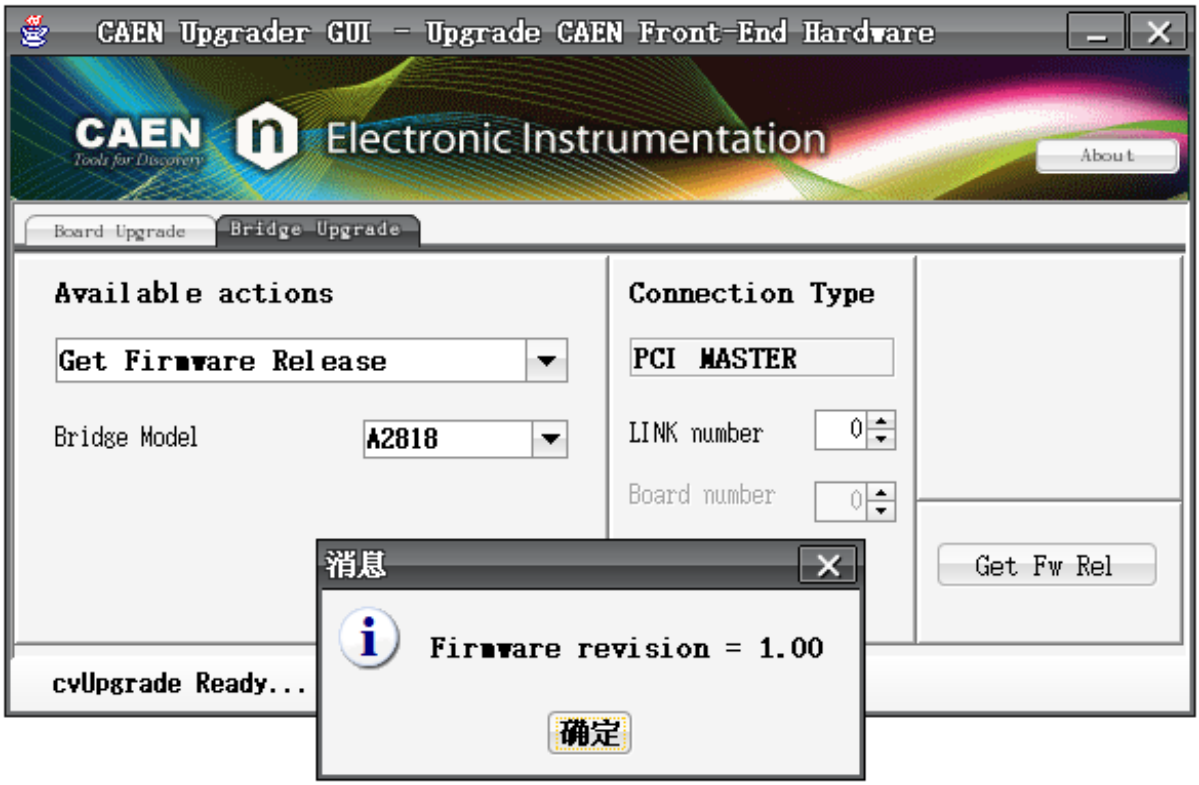


如果该固件版本不是 **当前固件版本**所列版本，则升级固件。

升级界面如下图所示：

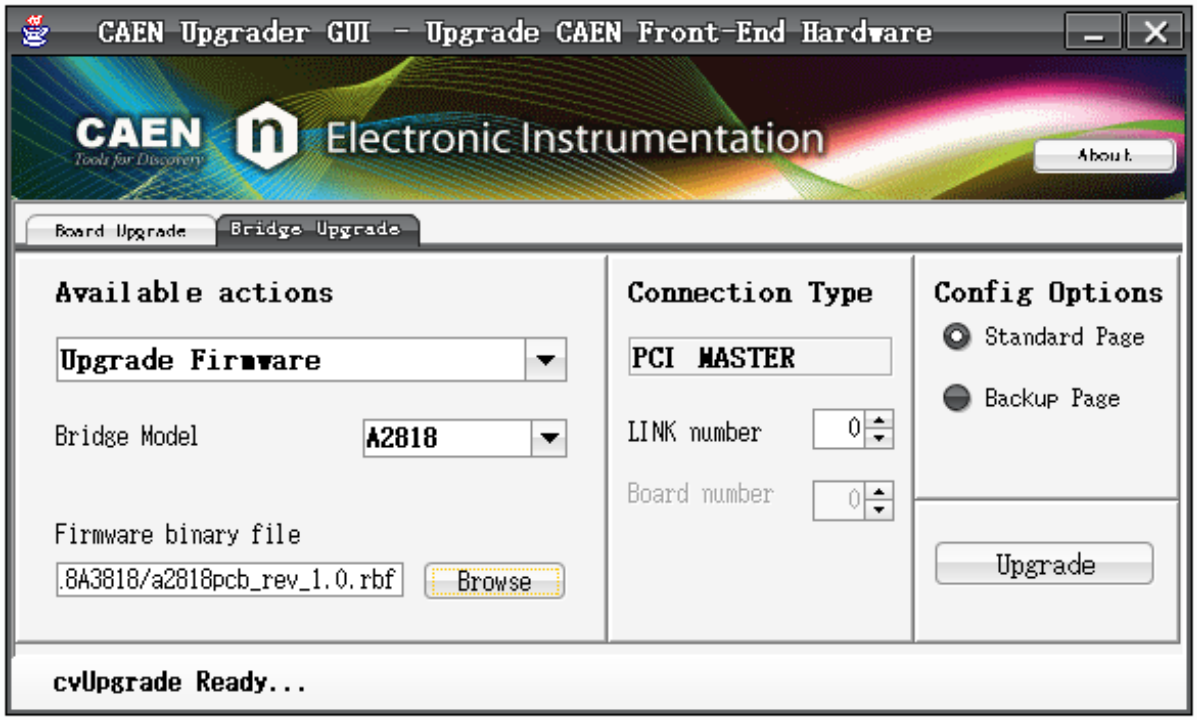
3.2.3 A2818

如下图，查看 A2818 的固件版本，点击 *Get Fw Rel* 按钮。



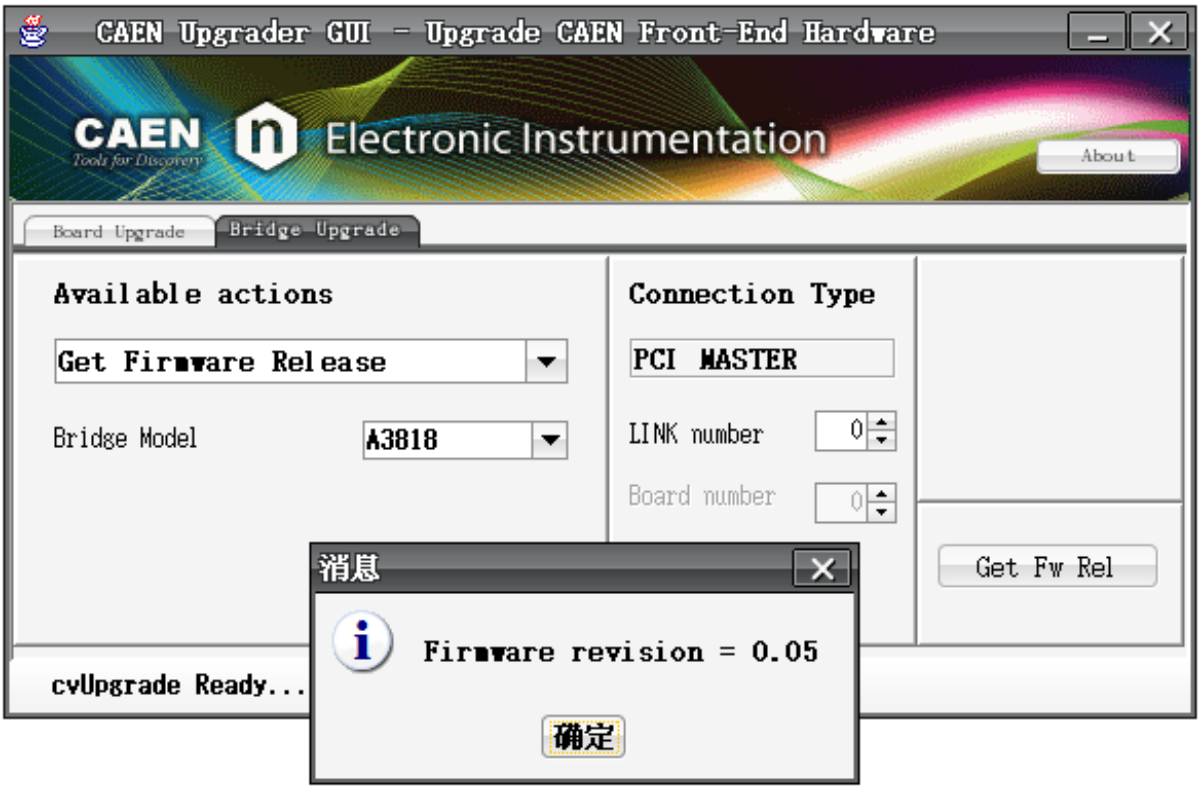
如果该固件版本不是 **当前固件版本**所列版本，则升级固件。

升级界面如下图所示：



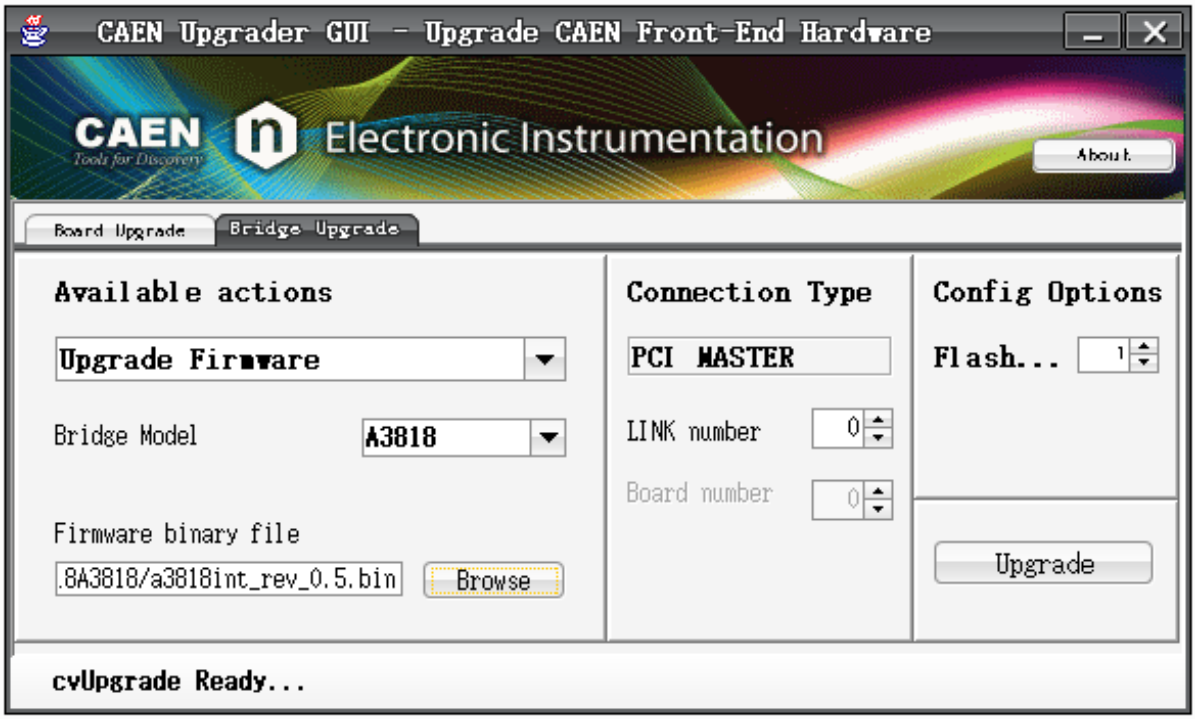
3.2.4 A3818

如下图，查看 A3818 的固件版本，点击 *Get Fw Rel* 按钮。



如果该固件版本不是 **当前固件版本**所列版本，则升级固件。

升级界面如下图所示：



3.2.5 V1x90

- **V1190/V1290**
  - Firmware Revision Register(Base Address + 0x1026, read only, D16)

  - This register contains the firmware revision number coded on 8 bit.

待补充

3.2.6 MADC32

- **madc32**
  - 0x600E firmware_revision

待补充

CHAPTER 4

获取配置

4.1 模块安放顺序

为了方便用户配置 DAQ，这里我们建议用户按照以下顺序依次插入采集模块（如果没有某些类型模块，则调过相应类型的模块）：

- 控制器 V1718/V2718
- 定标器 V830
- V7xx
- V1x90
- MADC32

4.2 程序修改建议顺序

- anaroot/CBLT.hh
- DAQConfig/StartDAQ.sh
- DAQConfig/StopDAQ.sh
- DAQConfig/bbcaenvme/babies/bbmodules.h
- DAQConfig/bbcaenvme/babies/start.c
- DAQConfig/bbcaenvme/babies/evt.c
- DAQConfig/bbcaenvme/babies/clear.c
- DAQConfig/bbcaenvme/babies/stop.c
- DAQConfig/bbcaenvme/init/daqinitrc.sh

修改程序，请先仔细阅读 DAQConfig 页面中的说明。

4.3 V1718/V2718

V1718/V2718 PCB 板上 DIP 开关：Prog: 0 off, 1 off, 2 off, 3 on, 4 off, I/O NIM

v1718/V2718 前面板 5 个 LEMO 输出口，分别为 0-4

通电时候输出口 0-3 处于高电平，输出口 4 处于低电平。因此软件 BUSY 模式时候采用输出口 4，硬件 BUSY 模式采用输出口 3。

4.4 软件 BUSY 模式

待补充

4.5 硬件 BUSY 模式

待补充

CHAPTER 5

analysis

存放辅助分析程序，当前只放置一个 MakeProcess 模板。

CHAPTER 6

anaroot

如果采用 CBLT 模式读取数据，则先修改 *CBLT.hh* 文件，不采用 CBLT 模式则不用修改。设置好之后，执行该目录下的自动编译、安装脚本 *autoPKU.sh* 即可

```
sh autoPKU.sh
```

修改 **CBLT.hh** 文件，其中设置应该与 CBLT 模式下的插件设置顺序一致。

当前 CBLT chain 支持 v830、v7xx、v1190、v1290、madc 五种类型的插件，如下所示：

```
#define v830m
#define v7xxm
#define v1190m
#define v1290m
#define madcm
```

获取中如果没有哪一种类型插件，则需注释掉该类型的定义！！！

以下 *xxxn* 为启用插件的数据顺序，从 0 开始编码，如果五种类型插件都有，则为以下设置：

```
#define v830n 0
#define v7xxn 1
#define v1190n 2
#define v1290n 3
#define madcn 4
```

如果只含有 v7xx、madc 两种类型的插件，则定义如下：

```
#define v7xxn 0
#define madcn 1
```

如果只含有 v830、v7xx、madc 三种类型的插件，则定义如下：

```
#define v830n 0
#define v7xxn 1
#define madcn 2
```

以下定义用来指定每种类型插件的个数

```
#define v830num
#define v7xxnum
#define v1190num
#define v1290num
#define madcnum
```

以下是 v830 的其它设置

```
#define v830chn 8 // 这里设置 830 开启路数
#define v830head 1 // 不要修改
#define v830geo 0 // 不要修改
```

CHAPTER 7

checkcnt

用来辅助检查文件中事件是否关联。执行程序之后将会在该文件夹内生成一个 pdf 文件，检查该文件内每张图数值是否有异常。

CHAPTER 8

cutpedestal

用来辅助设置 pedestal 数值。高斯拟合 pedestal，并给出三倍 sigma 的上限作为推荐数值，并生成初始文件夹 init 内脚本。

CHAPTER 9

DAQConfig

首先修改 **StartDAQ.sh/StopDAQ.sh** 两个文件，将文件内的 *wuhongyi* 替换成当前 LINUX 的用户名。然后修改 bbcaenvme 文件夹下 babies、init 文件夹内文件。

修改文件之前，我们需要先理解硬件地址与 GEO 编号。每个插件模块侧面都有四个拨盘，每个拨盘代表一个 16 进制位，例如，当四个拨盘从左到右分别为 1，2，3，4 时，表示其硬件地址为 0x1234。控制器与模块的通讯依靠该硬件地址来寻址，每个控制器下的模块地址应该具有唯一性。同时，我们可以通过软件对每个模块设置一个 GEO 编号，该模块输出数据中都会带有该 GEO 标记，方便我们对数据进行解码。GEO 编号范围为 0-31。

这里我们对硬件地址设置进行如下约定（当然以下约定不是强制要求，用户可以任意修改），

- v7xx 模块硬件地址从 0x1000 开始，然后 0x1001，有多少个模块就依次往后设置。
- MADC32 硬件地址从 0x2000 开始，然后 0x2001，有多少个模块就依次往后设置。
- v1x90 硬件地址从 0x4000 开始，然后 0x4001，有多少个模块就依次往后设置。
- v830 模块硬件地址从 0x5000 开始，然后 0x5001，有多少个模块就依次往后设置。

这里我们对 GEO 地址设置进行如下约定（当然以下约定不是强制要求，用户可以任意修改），

- 一般 v830 使用 1-2 个模块而已，因此 GEO 编号 30/31 我们预留给 v830。第一个 v830 的 GEO 为 30，第二个 v830 的 GEO 为 31。
- 实验中一般 v7xx 和 MADC32 模块使用较多，如果每种模块均不超过 10 个的话，我们默认 0-9 预留给 v7xx，10-19 预留给 MADC32。如果某种模块超过 10 个话，那么 v7xx 和 MADC32 的 GEO 按照 0-19 的编号依次往下进行分配。
- 实验中 v1x90 模块的使用数量一般不会超过 5 个，这里我们为 v1x90 预留 20-24，编号从 20 开始依次往后进行分配。
- 剩余的 GEO 编号 25-29 进行机动使用。

整个 DAQ 的程序配置中，需要在文件 babies/bbmodules.h 中进行硬件地址设置。然后还需对文件夹 init 内的文件进行硬件地址和 GEO 的设置。

9.1 babies/bbmodules.h

修改 ADCADDR、MADCADDR、V1x90ADDR、SCAADDR 使之与硬件地址匹配（可以多余设置，不可少设置）。其它不要修改。

这里我们按照之前的约定，V7xx 硬件地址从 0x1000 开始编号，用 ADC[x]ADDR 来表示不同模块。MADC32 硬件地址从 0x2000 开始编号，用 MADC[x]ADDR 来表示不同模块。V1x90 硬件地址从 0x4000 开始编号，用 V1x90ADDR[x] 来表示不同模块。V830 硬件地址从 0x5000 开始编号，用 SCAADDR[x] 来表示不同模块。（其中 [x] 代表不同的数字）

如果您依照我们的约定来设置，则不需要修改本文件。

9.2 babies/start.c

根据文件内提示设置，有该类型插件则开启对应代码，开启对应类型 busy 代码。其它不要修改。

busy 模式

如果您使用软件 busy 模式时，则开启以下代码行，如果您使用硬件 busy 模式时，则注释掉以下行代码。

```
//....ooo0000ooo........ooo0000ooo........ooo0000ooo........ooo0000ooo......
// 以下部分用户需要修改

// 软件 busy
v2718_init_ioport(4,0,0);

// 以上部分用户需要修改
//....ooo0000ooo........ooo0000ooo........ooo0000ooo........ooo0000ooo......
```

V830

```
//....ooo0000ooo........ooo0000ooo........ooo0000ooo........ooo0000ooo......
// 以下部分用户需要修改

// 有 V830 插件
v830_clear_all(SCAADDR0);

// 以上部分用户需要修改
//....ooo0000ooo........ooo0000ooo........ooo0000ooo........ooo0000ooo......
```

用户需要修改以上代码段，如果您不使用 V830 模块，则注释掉以上区域的代码。

如果您使用一个 V830 模块，则添加代码：

```
v830_clear_all(SCAADDR0);
```

如果您使用两个 V830 模块，则添加代码：

```
v830_clear_all(SCAADDR0);
v830_clear_all(SCAADDR1);
```

V7xx

```
//....ooo0000ooo........ooo0000ooo........ooo0000ooo........ooo0000ooo......
// 以下部分用户需要修改

// 有 V7xx 插件
// 每个插件单独设置
v7xx_rst_counter(ADC0ADDR);
v7xx_rst_counter(ADC1ADDR);
v7xx_rst_counter(ADC2ADDR);
// v7xx_rst_counter(ADC3ADDR);
// v7xx_rst_counter(ADC4ADDR);
// v7xx_rst_counter(ADC5ADDR);
// v7xx_rst_counter(ADC6ADDR);
// v7xx_rst_counter(ADC7ADDR);
```

(下页继续)

(续上页)

```
v7xx_clear(ADC0ADDR);
v7xx_clear(ADC1ADDR);
v7xx_clear(ADC2ADDR);
// v7xx_clear(ADC3ADDR);
// v7xx_clear(ADC4ADDR);
// v7xx_clear(ADC5ADDR);
// v7xx_clear(ADC6ADDR);
// v7xx_clear(ADC7ADDR);

// 以上部分用户需要修改
//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......
```

用户需要修改以上代码段，如果您不使用 V7xx 模块，则注释掉以上区域的代码。

如果您使用一个 V7xx 模块，则添加代码：

```
v7xx_rst_counter(ADC0ADDR);
v7xx_clear(ADC0ADDR);
```

如果您使用两个 V7xx 模块，则添加代码：

```
v7xx_rst_counter(ADC0ADDR);
v7xx_rst_counter(ADC1ADDR);
v7xx_clear(ADC0ADDR);
v7xx_clear(ADC1ADDR);
```

使用更多 V7xx 则依次类推。

V1x90

```
//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......
// 以下部分用户需要修改

// 有 V1190/V1290 插件
// 每个插件单独 clear
// v1190_clear(V1x90ADDR0);
// v1290_clear(V1x90ADDR1);

v1190_clear(V1x90ADDR0);
v1190_clear(V1x90ADDR1);
// v1290_clear(V1x90ADDR0);
// v1290_clear(V1x90ADDR1);

// 以上部分用户需要修改
//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......
```

用户需要修改以上代码段，如果您不使用 V1x90 模块，则注释掉以上区域的代码。

如果您只使用一个 V1190 模块，则添加代码：

```
v1190_clear(V1x90ADDR0);
```

如果您只使用两个 V1190 模块，则添加代码：

```
v1190_clear(V1x90ADDR0);
v1190_clear(V1x90ADDR1);
```

如果您使用一个 V1190，一个 V1290，则添加代码：

```
v1190_clear(V1x90ADDR0);
v1290_clear(V1x90ADDR1);
```

更多模块使用的组合，请以此类推。

MADC32

```
//....ooo00000ooo........ooo00000ooo........ooo00000ooo........ooo00000ooo......
// 以下部分用户需要修改

// 有 MADC32 插件
madc32_mclear(MSTMDCADDR);
madc32_mirq_level(MSTMDCADDR,0);
madc32_mreset_ctra_counters(MSTMDCADDR);
madc32_mfifo_reset(MSTMDCADDR);
madc32_mstart_acq(MSTMDCADDR);

// 以上部分用户需要修改
//....ooo00000ooo........ooo00000ooo........ooo00000ooo........ooo00000ooo......
```

用户需要修改以上代码段，如果您不使用 MADC32 模块，则注释掉以上区域的代码。如果您使用了 MADC32 模块，不管使用了多少个模块，只需要开启以上代码即可对所有的模块完成初始化。

busy 模式

```
//....ooo00000ooo........ooo00000ooo........ooo00000ooo........ooo00000ooo......
// 以下部分用户需要修改

// 硬件 busy / 软件 busy 中的多机箱同步
// v2718_clear_ioport(3);

// 软件 busy
v2718_pulse_ioport(4);

// 以上部分用户需要修改
//....ooo00000ooo........ooo00000ooo........ooo00000ooo........ooo00000ooo......
```

如果您是软件 busy 模式，则开启代码：

```
v2718_pulse_ioport(4);
```

如果您是硬件 busy 模式或者软件 busy 模式下的多机箱同步方案下，则需要开启代码：

```
v2718_clear_ioport(3);
```

9.3 babies/evt.c

根据文件内提示设置。其它不要修改。

```
//....ooo00000ooo........ooo00000ooo........ooo00000ooo........ooo00000ooo......
// 以下部分用户需要修改

// 软件 BUSY 模式下 6036->0x1 不需要以下清除，6036->0x3 需要以下清除，6036->0x0 需要以下清除
// 硬件 BUSY 模式下只能采用 6036->0x3，需要以下清除

// 有 MADC32 插件
madc32_mclear(MSTMDCADDR);

// 以上部分用户需要修改
//....ooo00000ooo........ooo00000ooo........ooo00000ooo........ooo00000ooo......
```

用户需要修改以上代码段，如果您不使用 MADC32 模块，则注释掉以上区域的代码。如果您使用了 MADC32 模块，不管使用了多少个模块，只需要开启以上代码即可对所有的模块完成清除。

当然，在软件 busy 模式下，对每个模块的寄存器进行相应的寄存器配置，可以不用以上清除指令自动进行清除，此时每个事件能够节约 20 us 左右的时间，该方案建议对 DAQ 比较熟悉的用户使用。

9.4 babies/clear.c

根据文件内提示设置，有该类型插件则开启对应代码，开启对应类型 busy 代码。其它不要修改。

如果您使用软件 busy 模式时，则开启以下代码行，如果您使用硬件 busy 模式时，则注释掉以下行代码。

```
//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......
// 以下部分用户需要修改

// 软件 busy
v2718_pulse_ioport(4);

// 以上部分用户需要修改
//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......
```

9.5 babies/stop.c

根据文件内提示设置，有该类型插件则开启对应代码，开启对应类型 busy 代码。其它不要修改。

MADC32

```
//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......
// 以下部分用户需要修改

// 有 MADC32 插件
madc32_mstop_acq(MSTMDCADDR);

// 以上部分用户需要修改
//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......
```

用户需要修改以上代码段，如果您不使用 MADC32 模块，则注释掉以上区域的代码。如果您使用了 MADC32 模块，不管使用了多少个模块，只需要开启以上代码即可对所有的模块发送结束采集指令。

busy 模式

如果您是硬件 busy 模式或者软件 busy 模式下的多机箱同步方案下，则需要开启代码，否则注释掉以下代码：

```
//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......
// 以下部分用户需要修改

// 硬件 busy / 软件 busy 多机箱同步
// v2718_set_ioport(3);

// 以上部分用户需要修改
//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......
```

9.6 init/daqinitrc.sh

修改该文件内对应脚本，使之与获取插件对应，用来初始化插件。

重点是修改 cblt.hh 文件，对启用的插件设置 CBLT ADDR 为 0xbb，其中 MADC 还得设置 MCST ADDR 为 0xdd。还得设置每一个插件在 CBLT 中的顺序，first、mid、last。至少得两个插件才能组成 CBLT.

init/daqinitrc.sh 文件包含以下内容：

```
#!/bin/sh

/bin/sh ./v830.sh
/bin/sh ./v7xx_all.sh
# /bin/sh ./v7xx_thres.sh
/bin/sh ./v1190_0.sh
/bin/sh ./v1190_1.sh
# /bin/sh ./v1290.sh
/bin/sh ./madcall.sh
# /bin/sh ./madc_thres.sh
/bin/sh ./cblt.sh
```

如果您使用了 V830 则开启以下代码，否则注释掉以下代码：

```
/bin/sh ./v830.sh
```

待补充

CHAPTER 10

httponline

基于网页的在线监视。

CHAPTER 11

online

时时监视每路信号的能量信息。

按照提示修改 Online.cc 文件

图形化界面开发中。。。

CHAPTER 12

r2root

仅仅需要修改插件定义即可，无需修改其它代码。
修改文件 UserDefine.hh，按照提示修改即可。

CHAPTER 13

statistics

用来监视每路的计数率。