SSRF 光束线控制系统 技术文档	编号： BC-16-002 日期： 2016-03-04 第 1 页 共 11 页

名称	Galil 运动控制器技术报告
作者	刘平 贾文红

摘要

Galil 2183 是美国 Galil Motion Control 公司的经济型多轴运动控制器。高度 1U，可控制步进电机、伺服电机。还提供多路 I/O 端口，可用于同步触发外部事件。

为了和束线控制组的运动控制标准软硬件平台集成，控制组开发了一个 3U 标准上架机箱和相关 I/O 信号接插件，命名为“Galil 运动控制箱”。本文档介绍 Galil 控制器的技术应用。

日期	修订内容

目录

1、简介

Galil 2183 是美国 Galil Motion Control 公司的经济型多轴运动控制器。高度 1U，可控制步进电机、伺服电机。还提供多路 I/O 端口，可用于同步触发外部事件。

为了和束线控制组的运动控制标准软硬件平台集成，控制组开发了一个 3U 标准上架机箱和相关 I/O 信号插头，命名为“Galil 运动控制箱”。由于该控制器基于网口，最后的控制箱成品比较轻便，非常适合携带出差，外出到厂家调试设备。

前期使用 Galil-3-0 版本的 IOC 软件，有时存在通信异常的情况。于是采用了新的 3-1 版本软件。

2、软件安装与编译

下载好 base-3.14.12.4 和 synApps-5-7，另外从 http://motorapp.github.io/Galil-3-0/ 下载 Galil-3-1 版本软件。一台安装了 CentOS6.4 的计算机。

Galil-3-1 版本软件是基于 ASYN 的最新软件，它舍掉了厂家原配的通信驱动模块，直接用 ASYN 来开发，也就是说这是一次大改版。因此要正确编译该软件，需要计算机安装”C++ 2011”语法版本的 GCC 编译器，它比 CentOS 自带的 GCC 编译器版本高，而且需要用 Calc-3-4-2 代替 synApps-5-7 中的 Calc-3-2 模块。

由于支持 C++ 2011 语法的新 GCC 编译器离线安装比较复杂，选择直接联网在线安装。切换到超级用户，执行下面的具体指令：

```
wget http://people.centos.org/tru/devtools-2/devtools-2.repo -O /etc/yum.repos.d/devtools.repo
yum install devtoolset-2-gcc devtoolset-2-binutils devtoolset-2-gcc-c++
cd /usr/local/bin
ln –s /opt/rh/devtoolset-2/root/usr/bin/* .
cd /usr/bin
mv gcc gcc_centos
mv g++ g++_centos
```

ln –s /usr/local/bin/gcc gcc
ln –s /usr/local/bin/g++ g++

安装编译 base-3.14.12.4，如果已经存在，请重新编译。

删除 synApps-5-7 里面的 Calc-3-2 目录，拷贝添加 Calc-3-4-2 目录。把 synApps-5-7 目录改名，比如改成 galil-support-5-7，然后重新编译。一些编译出错的模块，比如 DXP、motor 模块、areaDector 等模块，直接注释掉。这样的改动，只适合 Galil 的应用。当需要常规的其它应用时，请重新解压一个 synApps-5-7，然后进行 RELEASE 配置，最后编译。这里的区别主要是 calc 的版本不一样，会带来一些编译错误。而新 GCC 编译器都适用。

如果在 linux-x86_64 系统下运行，请注意配置 base 的 CONFIG_SITE 文件及修改 st.cmd 的第一行，改成

#！../../bin/linux-x86_64/GalilTestApp

3、具体应用

BL14B1 衍射仪的 TTH、Phi 和 TH 三个轴改成了 Galil 控制。Galil 运动控制的软 IOC 放在了如下图中的 MOXA-DA682 计算机中。

三个轴都是使用 5 相驱动器 SMD5002，第 3 档细分（10 细分），F 档驱动电流（1.35A），内部 PCB 板上的拨码开关设置如下：

位 1	2	3	4	5	位 6
Low（向上）			Low		
	High（向下）	High		High	High

Galil 控制器的前面板设置了“Home”信号的 8 位拨码开关、“Limit” 信号的 8 位拨码开关，如下图。它们的作用是：当 Home 开关和 Limit 开关的电平不满足 Galil 的内部逻辑要求时，通过调节前面板的拨码开关来切换信号的高低电压，以满足电平逻辑要求。这个处理方法比重新焊接硬件的常开或常闭触点简单快捷。

3.1 启动 IOC

远程登录 MOXA 计算机（10.30.55.78）的 blctrl 账户，进入 BL14B1-EX-Galil-SoftIOC 目录。在该目录下，已经创建了一个 start-Galil-Motor.sh 文件，直接执行该文件就启动了 IOC。一般地，使用如下的命令使该 IOC 在后台运行：

screen –dmS Galil ./start-Galil-Motor.sh

3.2 启动脚本 galil.cmd

在 st.cmd 脚本文件里，又调用了 galil.cmd 脚本，主要的配置都在该脚本文件。

```
#Load motor records for real and coordinate system (CS) motors
dbLoadTemplate("$(TOP)/GalilTestApp/Db/galil_motors.substitutions")

#Load extra features that have controller wide scope (eg.  Limit switch type, home switch type, output compare, message consoles)
dbLoadTemplate("$(TOP)/GalilTestApp/Db/galil_ctrl_extras.substitutions")

#Load extra features for real axis/motors (eg. Motor type, encoder type)
dbLoadTemplate("$(TOP)/GalilTestApp/Db/galil_motor_extras.substitutions")

#Load digital IO databases
dbLoadTemplate("$(TOP)/GalilTestApp/Db/galil_digital_ports.substitutions")

#Load user defined functions
dbLoadTemplate("$(TOP)/GalilTestApp/Db/galil_userdef_records.substitutions")
```

```
# GalilCreateController command parameters are:
# 1. Const char *portName    - The name of the asyn port that will be created for this controller
# 2. Const char *address        - The address of the controller
# 3. double updatePeriod- The time in ms between datarecords 8ms minimum.  Async if controller + bus supports it, otherwise is polled/synchronous.
# - Specify negative updatePeriod < 0 to force synchronous tcp poll period.  Otherwise will try async udp mode first

# Create a Galil controller
GalilCreateController("Galil2183", "192.168.1.36", 18)

# GalilCreateAxis command parameters are:
# 1. char *portName Asyn port for controller
# 2. char    axis A-H,
# 3. int     limits_as_home (0 off 1 on),
# 4. char    *Motor interlock digital port number 1 to 8 eg. "1,2,4".   1st 8 bits are supported
# 5. int     Interlock switch type 0 normally open, all other values is normally closed interlock switch type

# Create the axis
GalilCreateAxis("Galil2183","A",0,"",1)
GalilCreateAxis("Galil2183","B",0,"",1)
GalilCreateAxis("Galil2183","C",0,"",1)
GalilCreateAxis("Galil2183","D",0,"",1)
GalilCreateAxis("Galil2183","E",0,"",1)
GalilCreateAxis("Galil2183","F",0,"",1)
GalilCreateAxis("Galil2183","G",0,"",1)
GalilCreateAxis("Galil2183","H",0,"",1)


# GalilStartController command parameters are:
#
# 1. char *portName Asyn port for controller
# 2. char *code file(s) to deliver to the controller we are starting. "" = use generated code (recommended)
#                                Specify a single file or to use templates use: headerfile;bodyfile1!bodyfile2!bodyfileN;footerfile
# 3. int     Burn program to EEPROM conditions
#                0 = transfer code if differs from eeprom, dont burn code to eeprom, then finally execute code thread 0.
#                1 = transfer code if differs from eeprom, burn code to eeprom, then finally execute code thread 0.
#                It is asssumed thread 0 starts all other required threads
```

4. int display code. Set bit 1 to display generated code and or the code file specified. Set bit 2 to display uploaded code

5. int Thread mask. Check these threads are running after controller code start. Bit 0 = thread 0 and so on

if thread mask = 0 and GalilCreateAxis appears > 0 then threads 0 to number of GalilCreateAxis is checked (good when using the generated code)

GalilStartController("Galil2183", "BL14B1-EX-Galil.gmc", 0, 0, 0)

3.3 寻零程序

Galil 的寻零功能，由源代码编译产生一个以“.gmc”为后缀的文件，在启动脚本中配置，然后下载到 Galil 控制器中运行，gmc 文件类似 PLC 程序，是扫描执行。

Galil-3-1 原版本的启动脚本中函数 GalilCreateAxis 的第三个参数是 limits_as_home ，它用来设置是否把正负限位作为 Home 参考点。如果设置成“1”，则无论触动正限位还是负限位，都会把当前电机位置自动置 0。考虑到我们的使用习惯，及其 Home 开关寻零与编码器寻零的区分需要，在源代码中更改了寻零程序。在 gen_homecode()及 home()函数中，创建了一个叫“hsw”的变量，该变量根据脚本中的 limits_as_home 参数值，赋值成 0 或 1。在寻零功能被激活时，根据“hsw”的值，如果为 1，则进行 Home 开关寻零；如果为 0，则进行编码器寻零。注意这里的 limits_as_home 已经失去了原作者定义的含义。在底层 C++寻零代码中不能使用“FE”指令，否则电机只能按照“CN”指令确定的方向进行单向寻零，也就是说无论点击“HomR”还是“HomF”，电机都往同一个方向运动进行寻零。

在 motor_extras 的 db 模板中，要让$(P):$(M)_JAH_CMD 的值为 0，否则如果等于 1，电机寻到零位后，会自动走到限位，而不是停在零点位置。成功寻到零后，motor 域“ATHM”没有置 1，查看并修改了 GalilAxis.cpp 里面的 poll（）函数，使其寻到零后“ATHM”域为 1。

下面是具体的寻零代码： （为了篇幅，只保留了 A、B 轴，其它轴类似）

```
#AUTO
XQ #THREADB,1
XQ #THREADC,2
XQ #THREADD,3
XQ #THREADE,4
XQ #THREADF,5
XQ #THREADG,6
XQ #THREADH,7
#THREADA
IF (homeA=1)
IF ((hjogA=0) & (hswA=1))
hjogA=1;WT50;BGA;
ENDIF
```

```
IF ((hjogA=0) & (hswA=0) & (ueipA=1))
hjogA=1;FIA;WT50;BGA;
ENDIF
IF ((hjogA=1) & (hswA=1) & (_HMA=0))
STA;WT50;hjogA=2;
ENDIF
IF ((hjogA=1) & (hswA=0) & (_LRA=1) & (_LFA=1) & (ueipA=1) & (_BGA=0))
hjogA=3;
ENDIF
IF ((hjogA=2) | (hjogA=3))
hjogA=0;homeA=0;homedA=1;WT50;MG "homedA",homedA;MG "homeA",homeA;
ENDIF
IF ((hjogA=1) & ((_LRA=0) | (_LFA=0)))
homeA=0;hjogA=0;
ENDIF
ENDIF
counter=counter+1
IF (mlock=1)
II ,,dpon,dvalues
ENDIF
JP #THREADA
#THREADB
IF (homeB=1)
IF ((hjogB=0) & (hswB=1))
hjogB=1;WT50;BGB;
ENDIF
IF ((hjogB=0) & (hswB=0) & (ueipB=1))
hjogB=1;FIB;WT50;BGB;
ENDIF
IF ((hjogB=1) & (hswB=1) & (_HMB=0))
STB;WT50;hjogB=2;
ENDIF
IF ((hjogB=1) & (hswB=0) & (_LRB=1) & (_LFB=1) & (ueipB=1) & (_BGB=0))
hjogB=3;
ENDIF
IF ((hjogB=2) | (hjogB=3))
hjogB=0;homeB=0;homedB=1;WT50;MG "homedB",homedB;MG "homeB",homeB;
ENDIF
IF ((hjogB=1) & ((_LRB=0) | (_LFB=0)))
homeB=0;hjogB=0;
ENDIF
ENDIF
JP #THREADB
……..
```

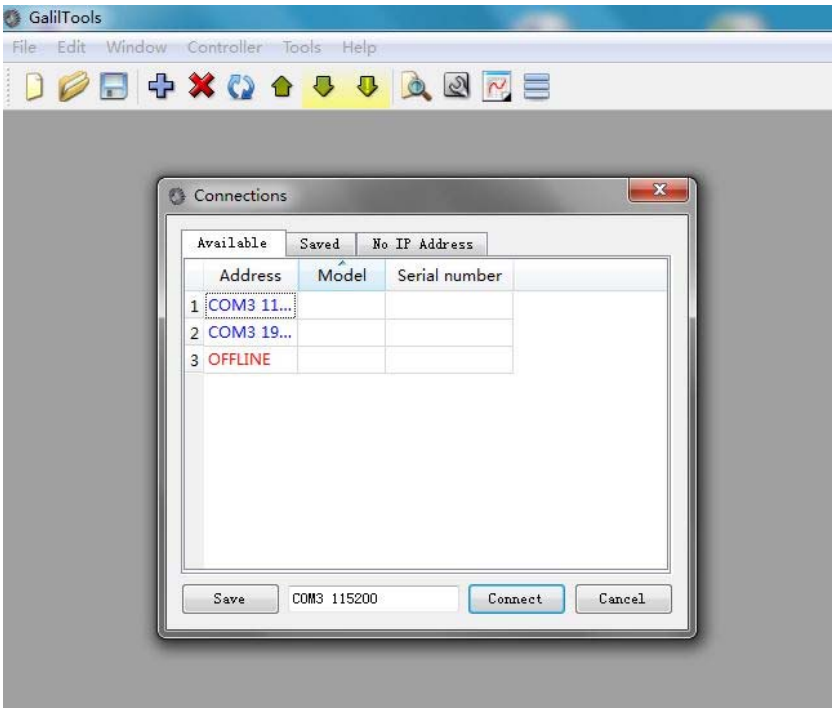
```
……...
……
#CMDERR
errstr=_ED;errcde=_TC;cmderr=cmderr+1
EN
#LIMSWI
IF (((_SCA=2) | (_SCA=3)) & (_BGA=1))
oldecelA=_DCA;ocds=_VDS;ocdt=_VDT;DCA=limdcA;VDS=limdcA;VDT=limdcA;STA
DCA=oldecelA;VDS=ocds;VDT=ocdt;ENDIF
IF (((_SCB=2) | (_SCB=3)) & (_BGB=1))
oldecelB=_DCB;ocds=_VDS;ocdt=_VDT;DCB=limdcB;VDS=limdcB;VDT=limdcB;STB
DCB=oldecelB;VDS=ocds;VDT=ocdt;ENDIF
IF (((_SCC=2) | (_SCC=3)) & (_BGC=1))
oldecelC=_DCC;ocds=_VDS;ocdt=_VDT;DCC=limdcC;VDS=limdcC;VDT=limdcC;STC
DCC=oldecelC;VDS=ocds;VDT=ocdt;ENDIF
IF (((_SCD=2) | (_SCD=3)) & (_BGD=1))
oldecelD=_DCD;ocds=_VDS;ocdt=_VDT;DCD=limdcD;VDS=limdcD;VDT=limdcD;STD
DCD=oldecelD;VDS=ocds;VDT=ocdt;ENDIF
IF (((_SCE=2) | (_SCE=3)) & (_BGE=1))
oldecelE=_DCE;ocds=_VDS;ocdt=_VDT;DCE=limdcE;VDS=limdcE;VDT=limdcE;STE
DCE=oldecelE;VDS=ocds;VDT=ocdt;ENDIF
IF (((_SCF=2) | (_SCF=3)) & (_BGF=1))
oldecelF=_DCF;ocds=_VDS;ocdt=_VDT;DCF=limdcF;VDS=limdcF;VDT=limdcF;STF
DCF=oldecelF;VDS=ocds;VDT=ocdt;ENDIF
IF (((_SCG=2) | (_SCG=3)) & (_BGG=1))
oldecelG=_DCG;ocds=_VDS;ocdt=_VDT;DCG=limdcG;VDS=limdcG;VDT=limdcG;STG
DCG=oldecelG;VDS=ocds;VDT=ocdt;ENDIF
IF (((_SCH=2) | (_SCH=3)) & (_BGH=1))
oldecelH=_DCH;ocds=_VDS;ocdt=_VDT;DCH=limdcH;VDS=limdcH;VDT=limdcH;STH
DCH=oldecelH;VDS=ocds;VDT=ocdt;ENDIF
RE 1
EN
```

3.4 IP 设置方法

对于新的 Galil 控制器，出厂没有缺省的 IP 地址，需要利用 Windows 系统下安装的软件 GalilTools。

先配置好计算机的 IP 地址，比如 192.168.127.9。打开 GalilTools 软件，鼠标点击图标栏中的“+”，就会弹出“Connections”通信连接功能设置界面，如下图。再点击第三个标签页“No IP Address”，选中列表中的 DMC-2XXX 行，下面的“Assign”及它前面的输入框就会激活（不在显示灰色）。输入框中自动填入的是 Host 主机的 IP，请修改成自己需要的 IP（比如 192.168.127.100），点击“Assign”按钮，列表中的 DMC-2XXX 行就会消失。再点击回“Available”标签页，在该标签页的最下方，会出现刚才设置的 IP 地址，点击它后边

的“Connect”按钮，就建立好了 Galil 控制器与 Host 主机的网络通信，本设置界面会自动退出。通信建立后，在主界面的最右侧，会出现一些信息及一个冒号提示符“：”，冒号后面闪烁光标，意味着请在这里输入 Galil 允许的指令。



保存 IP：

如果当前 IP 是自己想要的，在闪烁提示符后面，输入“BN”指令回车，烧录 IP 信息到 EPROM。自从，IP 设置过程完成，可以断电重启，检查刚才的设置是否起效。

修改 IP：

如果当前 IP 不是自己想要的，比如要修改成 192.168.127.222，在闪烁提示符后面，输入“IA 192,168,127,222”指令回车，注意 IP 的数值是用逗号“,”隔开而不是点号，这里很容易犯错。再输入“BN”指令回车，烧录 IP 信息到 EPROM。自从，IP 设置过程完成，可以断电重启，检查刚才的设置是否起效。

还有一种修改 IP 的情况，即别人已经使用过 Galil 控制器，它已经有 IP 地址，自己希望更改一个新的 IP。这种情况，先把计算机的 IP 改成与 Galil 的 IP 同一个网段，然后打开 GalilTools 软件，鼠标点击图标栏中的“+”，弹出“Connections”设置界面。因为两者为同一网段，软件会自动寻找到 Galil 控制器，在“Available”标签页选中它，再点击“Connect”按钮，就建立好了通信连接。在主界面最右侧的“：”闪烁提示符后面，键入需要的 IP，比如“IA 10,30,40,6”。如果新的 10.30.40.6 与 Host 主机的 IP 不在同一网段，该软件会提示红色通信错误，不管它，此时网络已经因为跨网段而不通了。在之后，要保持 Galil 控制器不断电，因为刚才新 IP 是临时写入内存，还没有存入闪存。请再次修改 Host 计算机的 IP，使其与新 IP 同一网段，然后打开 GalilTools 软件，建立好通信连接。与前面的操作一样，键入“BN”指令回车，烧录 IP 信息到 EPROM。自从，IP 修改过程完成，可以断电重启。

3.5 保持电流

1）电机是否需要有保持电流，看设备需要及正确配置。

2）如果 SMD200X 拨码开关第 3 位置为“OFF”，则一直有保持电流。如果 SMD5012 (SMD5002)的拨码开关第 3 位置为“High”，则一直有保持电流。

3）如果上述的驱动器拨码开关没有设置一直有保持电流，则看 motor 记录的“POST”域的配置。如果“POST”域设置为“MOm”（m 是轴名，只能是大写的 A、B、C、D、E、F、G、H，依次代表 8 个轴），则电机停止后会切断电流，也就不在有保持电流。如果“POST”域为“SHm”（m 是轴名，只能是大写的 A、B、C、D、E、F、G、H，依次代表 8 个轴），则电机停止后有保持电流。如果“POST”域为空白，则要看 IOC 启动时

3.6 电机使能

电机要运动，必须要让电机使能。如果(P)(M)_AUTOONOFF_CMD 的值为“on”，则电机会自动使能。如果该 PV 为“off”，一是看另一个 PV：(P)(M)_ON_CMD，如果这个 PV 为“on”，则电机能动；二是看 motor 记录的“PREM”域是如何配置。如果“PREM”域设置为“SHm”（m 是轴名，只能是大写的 A、B、C、D、E、F、G、H，依次代表 8 个轴），则电机会运动，如果“PREM”域为空白，则电机不会动。

3.7 注意事项

1）“SH”指令是使马达进入 Enable 可励磁状态，“MO”指令是使马达进入 OFF 状态，根据需要在“PREM”和“POST”域设置。这一点不同于 MAXv8000 的 MN、MF 指令。

2）编码器没有专门配置指令，让相应马达的 UEIP=1 即启用了编码器。

3）注意${P}_ESTALLTIME_SP 和${P}_EDEL_SP 的时间设定值，如果太小，在这个时间内编码器当前读数与目标值误差大，就会报错：encoder stalled

4)马达的软限位必须设置，如果设置为 0，则电机不动。这一点不同于 MAXv8000。

5）Galil 源代码的 WLP（WrongLimitProtection）保护功能是当电机失控，方向跑反，触点硬限位，就会停止电机并报 WLP 错误。目前修改了 GalilAxis.cpp 文件，使得电机不管往哪个方向，只要触点一个硬限位，就停止电机，在界面报 WLP 错误。只要点击切换“WLP”开关按钮成“OFF”，才能恢复运动。等电机走出限位，记得再次点击切换“WLP” 开关按钮成“ON”状态。

6）motor 记录的“PREM”和“POST”域很重要，如果设置是空白，则看(P)(M)_AUTOONOFF_CMD 和(P)(M)_ON_CMD 的配置，如果有设置，则看这个设置，因为这个设置是最后起效的。

7）使用 SMD200X 驱动器时，注意使用方法，Motor type 不要设置成“3”——Low active stepper，而要设置成“5”——Rev Low active stepper。否则可能会出现电机反向离开限位开关时，无脉冲输出（Busy 灯不亮）。