中 国 科 学 院 上 海 天 文 台 年 刊
2011 年 总第 32 期 ANNALS OF SHANGHAI OBSERVATORY ACADEMIA SINICA No. 32, 2011

在 FS 下观测站特有设备的控制

薛祝和
(中国科学院上海天文台，上海 200030)

摘 要：介绍了 FS 软件的基本特征和 FS 控制标准硬件的详细工作流程。还具体介绍了在 FS 软件下控制观测站各种特有硬件的方法、步骤。本文可供中国 VLBI 网观测站程序员参考。

关 键 词：天文观测设备与技术；甚长基线干涉测量；控制软件

中图分类号：TP317, P228.6

1 FS 的基本特征

VLBI 观测站主要由 4 部分组成 —— 天线、接收机、终端和钟房设备。FS 是一个比较庞大的软件包 [1−3]，用于对 VLBI 观测站的自动化控制和管理。该软件可人机交互式地接收或从观测纲要文件中接收命令，交互的接口是 SNAP 语言 [4](Standard Notation for Astronomical Procedures 的简称，创建于 1979 年)。FS 软件最初是美国 NASA 和 MIT 一批人于 1979 年编写并逐步发展起来的，开始时运行在 HP 小型计算机上 (操作系统为 RTE 实时操作系统)，现在运行于普通 PC 机。FS 的当前版本运行在多任务操作系统 Debian Linux 下。随着要求的不断增长，FS 软件不断升级，目前的版本是 9.10.4。国际上统一用该软件做 VLBI 观测。FS 支持的 VLBI 终端系统几乎包括了目前世界上所有的终端类型。通过挂靠到 FS 的本地站程序，FS 还可支持各站特有的各种硬件和天线、接收机系统。运行时，所有事件，包括发出的命令、命令的响应、出错情况等，都会显示于屏幕并实时记录到一个日志文件 [5] 里，以便事后查询。部分只用 logit 程序直接写入日志文件的内容不显示在屏幕上。

FS 软件早期运行在 HP 小型计算机上，编程语言是 Fortran 加系统调用；后期的程序都采用 C 语言编程，少量脱机程序用了 shell 语言。早期的程序始终包含在后期版本的 FS 里，没有用 C 语言进行重新编制。

FS 只接受 Snap 命令 [6,7] 和时间流语句 (FS 软件的交互接口 Snap 语言由 2 部分组成 —Snap 命令和时间流语句。其 Snap 过程到最后仍会转化为 Snap 命令)。Snap 命令有 2 种格式：

①命令名 = 参数 1，参数 2，……

命令名可能是某硬件的助记符也可能是某种处理过程功能的助记符。“=”号后为运行该命令所需的一个或多个以逗号分隔的参数。本形式的命令通常用来设置某硬件的一些工作模式或某种计算处理的模式定义。有些参数也可不定义而取缺省值以减少输入量。

除非在执行该命令时出现错误，否则 FS 完成该命令后不回显响应信息或仅回显少量信息。例：

收稿日期：2011-06-16； 修回日期：2011-07-07

2011.116.21:30:04.98;bbc01=610.99,a,2.000,2.000

bbc01 命令执行后 FS 不显示任何信息 (这里指执行过程中没出任何错误的情况)。

②命令名

这是第二种形式的命令，它被用于监测一个模块的状态。响应的格式为：

命令名/参数 1，参数 2，……

其“参数”表示某硬件模块的状态或某个处理过程的最后结果。例：

2011.116.21:30:04.98;ifdab
2011.116.21:30:04.00/ifdab/0,0,nor,nor,1,65535,5856,0,1pps

Snap 语言里，命令只能顺序执行，无法跳跃或者返回执行，但可周期循环执行。Snap 命令或 Snap 过程在执行的过程中无法中断，只能等它执行完才能执行下一个命令，只有如 terminate 等立即执行命令才可终止它们的运行。

log 文件的格式说明如下：

< 时间代码 >< 类型代码 >< 数据 >

时间代码 —— yyyy.ddd.hh:mm:ss.ss。其中“yyyy”为年，“ddd”为该年的第几天，接下来分别为时、分、秒。本时间码对应该命令执行的时刻。

类型代码 —— 1 字节关于信息类型的标识字符。类型代码定义如下：

: —— 来自于观测纲要文件的命令。

; —— 来自于操作员人工输入的命令。

@ —— 来自于时间调度的命令。

$ —— 来自于 Snap 过程内的命令。

? —— 错误信息标志。

& —— Snap 过程的具体内容。

—— 来自于非 FS 程序的信息内容。

/ —— 来自于已经执行了的 Snap 命令的响应信息。

数据 —— 该部分信息包含命令名、响应信息等。其格式有：

命令名/参数，参数……

或者：命令名 = 参数，参数……

或者：命令名

其中，“参数”是一系列参数值中的一个。这些参数可以是字符、整数或者浮点数。各参数之间用逗号隔开。“/”表示对于命令的响应。“=”表示要设置参数的命令。如果只有命令名，没有参数跟随，则表示请求一个响应 (读取有关参数)。例：

2011.150.00:21:45.18&ifd01/ifdab=0,0,nor,nor
2011.150.03:00:08.66?ERROR sc -24 setcl: Mark 5B syncerr_gt_3
2011.150.17:00:40.55/bbc01/879.99,a,8,8,1,agc,6.57,5.91,lock,16259,16647,1005,1pps

Snap 语言的时间流控制语句由符号“!”开始，如“!2011153220300”、“!*”、“!+3m”等。

VLBI 实验里有 3 种时间 [8]：格式器 (Formatter) 时间、FS 软件时间和计算机时间。

①格式器时间：MK4 或 MK5 格式器的时间，与观测数据一一对应，非常重要。通常我们使用 fmset 程序 (或 gpstime 命令) 来设置它。

② FS 时间：FS 软件时钟，用于控制观测纲要文件 (Snap 文件) 按时间序列执行，其误差应控制在 1 s 内。我们使用 setcl 程序来控制与监视它。

③计算机时间：操作系统使用的时间，与 VLBI 实验无直接的厉害关系，误差几分钟也无关系。但对于使用 MK5B 的情况，由于有时需用操作系统时间去对 MK5B 的钟，所以最好用时间服务器来一直保持计算机时间的始终正确。

通过观测纲要 Snap 文件、过程控制文件等，FS 可自动控制整个 VLBI 观测。观测纲要 Snap 文件通常包括了整个观测区间的一系列按时间序列执行的命令和过程。多个 Snap 命令可以组成一个过程，过程里也可套用过程。

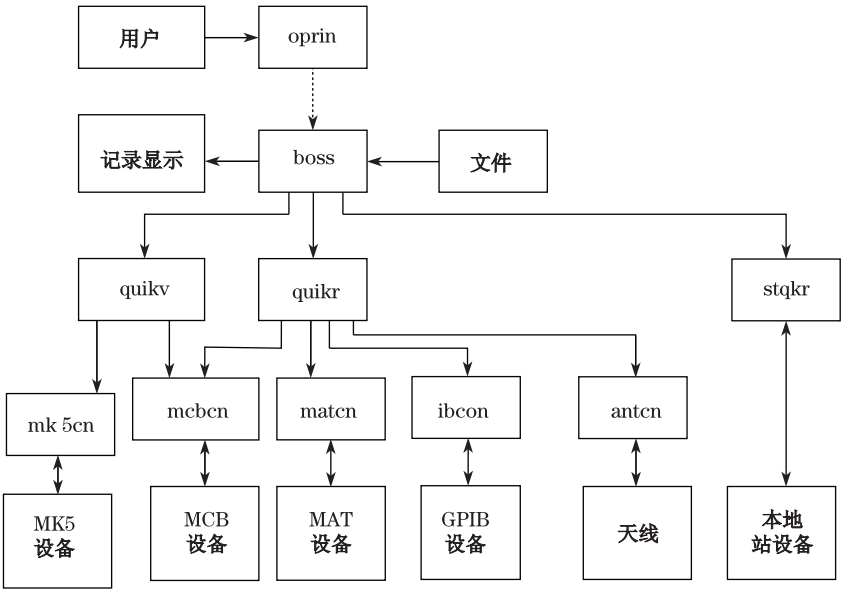

图 1 FS 软件工作流程图

FS 软件结构流程如图 1 所示。其用到的程序有：

oprin —— 人机交互程序。操作员通过 oprin 程序与 FS 软件进行交互通讯。

boss —— FS 主程序。它接收操作员通过键盘打入的或来自文件的 Snap 命令及过程名，然后对命令进行语法分析。如语法检查通过，就按照命令的类别分别传送给 quikv, quirk 或 stqkr 作进一步的处理。

quikr —— Snap 命令快速响应程序。它分析来自主程序 boss 的命令，将 Snap 命令转化为设备能够识别的一系列指令序列，传给 matcn[9]、ibcon[10] 这些设备通讯程序。主要控制 MK 系列设备、MK4 格式编制器、部分天线命令、HP 计数器等设备。

quikv —— MCB 命令 [11−13] 快速响应程序。它分析来自主程序 boss 的 MCB 类设备的

命令，将这些命令转化为设备能够识别的一系列 MCB 指令序列，传给 mcbcn 设备通讯程序。主要控制 VLBA 系列的中频分配器、中频视频变换器等设备。另外，MK5A、MK5B 记录终端等设备的命令处理入口也在该程序。

mk5cn —— MK5 命令通信程序。该程序将来自 FS 的 MK5 命令通过网络传给 mk5 设备，然后再接收 MK5 设备的响应信息并回传给 FS 软件。

mcbcn —— MCB 命令通信程序。该程序将来自 FS 的 MCB 命令传给设备，或者接收设备的响应信息然后再回传给 FS 软件。

matcn —— MAT 命令通信程序。该程序将来自 FS 的 MAT 命令传给设备，或者接收设备的响应信息然后再回传给 FS 软件。

ibcon —— GPIB 命令通信程序。该程序将来自 FS 的 gpib 命令传给 HP 设备，或者接收设备的响应信息然后再回传给 FS 软件。

antcn —— 天线命令通信程序。该程序将来自 FS 的天线控制命令传给天线，或者接收天线的响应信息然后再回传给 FS 软件。

stqkr —— 本地站程序 [14]。除通用设备外各观测站还有各自的专用设备，或者虽为通用设备但各自制作的硬件不同，所以各观测站还需为自己研制的设备开发相应的控制软件使其与 FS 软件联结起来。这种控制软件加上本地站的有关系统测试程序就是所谓的 VLBI 本地站程序。该程序控制一些专用设备，比如延迟计数器、噪声源装置、气象仪、接收机，相位校正信号装置、钟房设备等。

2　FS 控制硬件的具体步骤

Filed System 工作时通过标准的通用 FS 软件、本地站程序、/usr2/control/下的控制文件 [15]、本次实验的 snap 文件/usr2/sched/xxx.snp、本次实验的 proc 文件/usr2/proc/xxx.prc、本地站过程库/usr2/proc/station.prc[16] 来控制和管理本次实验的运转。

FS 启动后先做一些初始化工作，如：

A) 初始化 FS 预定义的共享变量 [17](共享变量清 0，共享变量的物理地址和范围大小已经由另外的程序在计算机启动时进行了设置。) 和其它有关变量。

B) 读入/usr2/control 目录下的控制文件，让 FS 知道本地观测站是什么类型的 VLBI 终端系统 (参见 equip.ctl)，有哪些 rs232 口 (参见 dev.ctl，但是有些本地站程序里的 rs232 口不一定在该文件里定义)，有哪些 VLBI 通用命令 (参见 fscmd.ctl)，有哪些本地观测站命令 (参见 stcmd.ctl)；系统里是否用到 GPIB 设备，如果用其代号和地址是什么 (参见 ibad.ctl)；MAT 和 MCB 设备的代号和地址 (参见 matad.ctl、mcbad.ctl, 如果本地站设备的控制里没有用到 matcn、mcbcn 通讯程序，这 2 个文件不需要修改)；flagr.ctl 文件里定义的参数即 FS 与 antcn 的通讯周期 (定时询问天线是否跟上了源，通常定义为 2 或 5 s)；FS 与哪些计算机通过网络相连 (参见 mk5ad.ctl。本地站程序的网络通讯可自定义控制文件)。skedf.ctl 里定义本站运行 drudg 时使用的缺省值。time.ctl 文件定义该 FS 计算机时钟的漂移系数，以用于 FS 软件钟的修正。FS 软件用到的出错代码和说明信息存放在 fserr.ctl 文件里，而本地站软件用到的出错代码 [18] 和说明信息存放在 sterr.ctl 文件里。

C) 启动并初始化各线程如 FS 的通用线程 (约 24 个，参见 fspgm.ctl，这些是随 FS 一起启

动的联机程序。FS 另外还有一些独立于它的脱机程序如 pfmed、setcl、logex、fmset、monitor 和一些 shell 程序等，这些脱机程序启动与否不影响 FS 软件)、本地观测站的线程 (理论上可以没有，但是一般至少有 1 个，即 stqkr，参见 stpgm.ctl。站 antcn 归类到 fspgm.ctl，因为 antcn 不能缺少。如果本地站里不用 antcn 控制天线，FS 会用自带的 1 个简单的 antcn 去模拟，但它与天线无实际通信，仅仅与 FS 的有关程序沟通信息，以免遇到 source 等天线命令时出错)。如果各线程初始化成功则各线程挂起然后等待输入的命令。如果有 1 个线程无法正常初始化，就会退出所有线程，然后退出 FS 主程序。

除了 FS 的标准检查程序 chekr 和本地站检查程序 cheks(若存在) 每 20 s 周期执行一次检查外，FS 初始化正常结束后即挂起，然后等待操作员输入的命令或来自 schedule 的命令。一旦收到命令 (无论来自键盘、文件或网络) 即进行语法检查，看该命令是否合法。检查的先后次序为：

①是否是本地站 Snap 命令，范围是/usr2/control/stcmd.ctl 文件内容；

②是否是 FS 标准 Snap 命令，范围是/usr2/fs/control/fscmd.ctl 文件内容 (有的 FS Snap 命令有多个命令处理入口，FS 会根据本地观测站的一些控制文件，如 equip.ctl 等，决定应该到哪个入口)；

③是否是本次实验过程库文件/usr2/proc/xxx.prc 里的过程；

④是否是本地站过程库文件/usr2/proc/station.prc 里的过程。

其中，本地站 Snap 命令的优先权最高。如果输入的命令在以上 4 个文件里都找不到符合点，则认为该命令非法。具体检查过程为：

表 1 /usr2/control/stcmd.ctl 文件例子

COMMAND	SEG	SBPA	BO	EQ	
cal	stq	0101	01	FFFFFFFFF	/* 28V except new S/X receiver */
cable	qkr	0402	01	FFFFFFFFF	
cablelong	qkr	0402	01	FFFFFFFFF	
fmout-gps	qkr	0402	01	FFFFFFFFF	
gpstime	stq	1301	01	FFFFFFFFF	
⋮	⋮	⋮	⋮	⋮	
wx	stq	3101	01	FFFFFFFFF	
clockn	stq	3201	01	FFFFFFFFF	
clocknow	stq	3201	01	FFFFFFFFF	
rcv	stq	3301	01	FFFFFFFFF	
xdb	stq	3501	01	FFFFFFFFF	
sdb	stq	3601	01	FFFFFFFFF	
power	stq	3701	01	FFFFFFFFF	
xdw	stq	3702	01	FFFFFFFFF	
⋮	⋮	⋮	⋮	⋮	
sif2	stq	3711	01	FFFFFFFFF	
⋮	⋮	⋮	⋮	⋮	

(一) 如果收到的命令与①类的/usr2/control/stcmd.ctl 文件第一列里的命令匹配，则参照表 1 做以下工作：

⑴　根据“SEG”列的内容决定该命令转入 stqkr 还是 quikr、quikv 程序处理；

⑵根据“SBPA”列的内容决定该命令转入 stqkr(或 quikr、quikv) 程序第几个命令 case 的第几个分支处理；

⑶“BO”列和“EQ”列的内容对本地站程序员来说无大影响，通常放“01”和 9 个“F”即可。其中“BO”数目代表该命令在 FS 命令处理时分在第几类，FS 通过“EQ”数确定该命令适用于哪个终端、哪种记录系统。

(二) 若收到命令与①类不符，则再与②类的/usr2/fs/control/fscmd.ctl 文件第一列里的命令匹配，若匹配，则：

⑴　根据“SEG”列的内容决定该命令转入 quikr、quikv 程序处理还是由 boss 程序处理。其中，“xxx”代表由 boss 程序直接处理，“*”打头的代表立即处理类命令 (优先权最高)；

⑵根据“SBPA”列的内容决定该命令转入 quikr 或 quikv 程序第几个命令 case 的第几个分支处理；

⑶“BO”列和“EQ”列的内容同①类的说明。

表 2　/usr2/fs/control/fscmd.ctl 文件例子

COMMAND	SEG	SBPA	BO	EQ
form	qkr	0101	01	001FFFFFF
form4	qkr	0102	01	FFFFFFFFF
decode4	qkr	0103	01	FFFFFFFFF
vc01	qkr	0201	01	205FFFFFF
vc02	qkr	0202	01	205FFFFFF
⋮	⋮	⋮	⋮	⋮
cont	*xx	0000	02	FFFFFFFFF
halt	*xx	0000	03	FFFFFFFFF
log	xxx	0000	04	FFFFFFFFF
schedule	xxx	0000	05	FFFFFFFFF
xlog	*xx	0000	06	FFFFFFFFF
xdisp	*xx	0000	07	FFFFFFFFF
echo	*xx	0000	08	FFFFFFFFF
terminate	*xx	0000	10	FFFFFFFFF

(三) 若收到的命令非 Snap 命令，即不符合①、②类，则查本次实验的过程库文件/usr2/proc/xxx.prc (如果当前没有进入到 schedule 则直接跳到 (四))，核对是否与 xxx.prc 里定义的过程名符合。若相符，则对该过程里的每条命令做①、②类的命令核查，直到确定是①、②类的命令之一为止。

(四) 如果收到命令与①、②、③类的命令不匹配，则查④ —— 本地站的过程库文件/usr2/proc/station.prc (如果此前发过 proc=yyy 命令，则先查找/usr2/proc/yyy.prc 文件进行匹配检查)，核对是否与 station.prc 里定义的过程名符合。若相符，则对该过程里的每一

条命令做①、②类的命令核查，直到确定是①、②类的命令之一为止。

若输入的命令在以上 4 个文件里都找不到符合点，则认为该命令非法，作出错处理。对合法的命令则逐一依次执行。根据①、②类的命令核查可以知道该命令应转入 quikr、quikv、stqkr 还是 boss 程序处理。这些程序的主程序里都有一段命令入口判别 switch 处理语句，其中的每一个 case 语句对应 1 条或数条 Snap 命令。对应 1 条还是数条由 fscmd.ctl 或 stcmd.ctl 里该命令的"SBPA" 数据决定。每个 case 语句里至少有 1 个函数用于处理该命令。如：

```
case 13: /* get GPS(Shanghai GPS) time & set formatter time */
    gpscnsh(&command,ip);
    break;
```

如果该命令与硬件控制有关，处理该命令的函数会调用相关的通讯程序与硬件通讯。标准的 MK 设备通过 matcn 通讯程序与硬件通讯，标准的 VLBA 设备通过 mcbcn 通讯程序与硬件通讯。非标准的站设备通过自编的通讯程序与硬件通讯。若站设备的接口定义与 MAT 或 MCB 一致，也可通过 mcbcn 或 matcn 与硬件通讯。

FS 与硬件通讯的流程为：

⑴ 当要求硬件做某一动作时，将对应的信息放到 buffer(通常是放到 FS 在初始化时已定义好的共享变量之一) 里；

⑵ 将 buffer 内容按照该硬件动作的接口定义转换整理为一个或一组数据流信息放到 class I/O 的 buffer 里。class I/O 是沿用了早期 HP 小型计算机 RTE 实时操作系统里的说法。class I/O 提供先入先出消息排队，用于程序间传递信息；

⑶ 调用 matcn 或 mcbcn 通讯程序；

⑷ 通讯程序通过 class I/O 得到数据流信息，并与硬件通讯 (通过串口线发送数据流信息到硬件，并回收来自硬件的回答信息。matcn 或 mcbcn 都使用串口线，MK5 设备使用网络。)；

⑸ 通讯程序将响应信息放到 class I/O buffer；

⑹ 调用通讯程序的控制程序通过 class I/O 获得响应信息，然后进行判别。如果控制正确执行了，就对有关共享变量进行变更，然后进行显示 (在计算机屏幕上和 log 文件里)。如果控制执行有错，编制出错代码 (包括错误分类符和错误号码。分类符和号码与 fserr.ctl 文件里的定义有对应关系)，然后将出错代码放入 IP 数组 (IP 数组是与上一级函数通讯的 1 个参数)；

⑺ 返回到调用该控制程序的上一级程序里。

3 如何在 FS 里增加对观测站新设备的控制

一个观测站总有一些自己研制的设备，如天线、接收机、气象仪、相位校正信号装置等，因此需为其编制相应的本地站程序并挂到 FS 上 [19,20]。其研发步骤如下：

① 要在 FS 运行时增加对新设备的控制 (这里不包括脱机程序)，首先必须按功能将其定义为一条或一组 Snap 命令。即使以过程的形式出现，最终也须化为一条条 Snap 命令来执行。定义格式为：

命令名 = 参数 1，参数 2，……

参数多寡由本命令功能的需要决定，有的功能不需参数 (如 gpstime)。对那种有参数的命令，在进入到对应的控制程序时，先要对参数的个数和合法值范围进行判别。命令执行后可能返回空 (无消息回馈即是已正常执行)，也可是“/ack”或其它文字说明。

另一种格式为：

命令名

本形式的命令多用于获得某硬件模块的状态或处理结果。对于操作后大于 1 s 才能得到结果的情况，通常先用第一种格式的命令设置好工作模式，然后再用第二格式的命令去取结果。如有多个结果用逗号隔开。如：命令名/参数 1，参数 2，……，结果 1，结果 2，……。对于不需要事先设置参数的取状态命令 (如 wx 命令)，其返回格式为：命令名/结果 1，结果 2，……。

② 在/usr2/control/目录下建立 3 个文件 —stpgm.ctl、stcmd.ctl 和 sterr.ctl。stpgm.ctl 例子如下：

* Put site-specific programs here that should be started by the Field System.
stqkr n stqkr &
sendl n sendl &

这里有 2 个随 FS 一起启动的本地站程序“stqkr”和“sendl”，多数情况下只需要“stqkr”一个就够了。天线通讯程序 antcn 划归/usr2/fs/control/fspgm.ctl 管。sterr.ctl 里定义本地站程序运行时出错的号码和说明，如：

〃 〃
ST -609
Failed to run cal=on/off command
〃 〃
ST -610
You missed parameters after XDB command ……

③ 在/usr2/control/stcmd.ctl 控制文件里增加新的命令名以及处理入口参数等。因为 FS 本体程序由美国 NASA 的 NVI 公司负责修改升级，所以我们不能将命令直接增加到/usr2/fs/control/fscmd.ctl 文件中。另外，stcmd.ctl 的优先权比 fscmd.ctl 高，所以当两者里有重名的命令时按 stcmd.ctl 定义的执行。

④ 在本地站主程序 stqkr.c 里增加新设备通信接口程序的初始化部分，需特别注意新增的变量不要与 stqkr.c 原有变量或 stqkr.c 里用到的共享变量重名，除非新旧变量的意义和用途完全一致；增加命令处理入口，即在 stqkr.c 程序里增加对应该 Snap 命令的 swith –case 语句处理，其中 case 的号码必须与/usr2/control/stcmd.ctl 文件里的命令处理入口号码相同。例：

power	stq	3701	01	FFFFFFFFF	(stcmd.ctl 文件内容)
⋮	⋮	⋮	⋮	⋮	
sif2	stq	3711	01	FFFFFFFFF	

```
初始化工作……          (stqkr.c 程序内容)
switch (isub)
{ ……
case 37:
rcv_pwr(&command,ip,&lurcv,itask);
break;
```

其中，“isub”的内容应是37；“itask”的内容可为1，2，…，11。itask参数的应用使rcv_pwr程序可以同时处理power，…，sif2多条Snap命令，这对于多个命令可公用一部分相同代码的情况能节省不少空间和时间。

另外“command”用来传递命令的各参数给rcv_pwr程序。结构“command”定义如下：

```
struct cmd_ds command;
#define MAX_ARGS 100    /* maximum number of args after '=' */
struct cmd_ds {         /* command data structure */
char *name;             /* pointer to command name STRING */
char equal;             /* '=' if '=' follows command name,'\0' otherwise */
char *argv[MAX_ARGS];   /* pointers to argument STRINGS, valid data terminated by
};                          a NULL pointer */
```

“ip”的定义为“long ip[5];”，该数组用于调用与被调用程序间互传运行的有关参数，如程序工作模式、class号、出错号码等。除在调试阶段外，无论FS还是本地站程序里的子程序都不允许直接在终端上输入或输出。所有信息通过“command”结构、“ip”和“class I/O”互传。

“lurcv”与特定的程序有关，有的程序无该参数，有的有多个参数。

⑤增加新设备的通信接口程序和控制程序

通信接口程序指自编的串口或网络通信程序，一般与FS无关联。控制程序里可以使用FS的共享变量，但一般只允许读不允许写，因为改变共享变量值后很容易引起FS程序混乱。如果确需写FS里的共享变量，必须仔细核查哪些FS程序会使用和改变该共享变量值，以及在共享变量值改变后是否会影响到FS其它程序的调用。有些本地站程序里得到的数据，需要送入FS的共享变量以供FS的程序使用，如wx得到的温度、湿度、气压需要写入FS的tempwx、humiwx、preswx以供FS的monit程序使用。例：

```
strncpy(output,p,5);
output[5]='\0';
sscanf(p,"%f",&shm_addr->tempwx);
p = &data[6];
strncpy(output,p,6);
output[6]='\0';
sscanf(p,"%f",&shm_addr->preswx);
p = &data[13];
strncpy(output,p,5);
```

```
output[5]='\0';
sscanf(p,"%f",&shm_addr->humiwx);
```

其中“output”的内容为本地站程序采样到的气象数据。

本地站程序使用 FS 共享变量前需要在程序里作如下定义 (尽管其中大部分的变量不会用到)：

```
#include "../../fs/include/params.h"
#include "../../fs/include/fs_types.h"
#include "../../fs/include/fscom.h"
#include "../../fs/include/shm_addr.h"    /* shared memory pointer */
……
struct fscom *fs;    定义 FS 指针到 fscom 结构
……
setup_ids();    调用 setup_ids 程序来连接 FS 的共享内存区域
```

一般情况下应定义本地站程序自己的共享变量，以免与 FS 程序冲突。定义本地站程序共享变量的步骤如下：

⑴ /usr2/fs/st.default/st 目录下 FS 提供了一个很简略的例子。为了借用 FS 里 /usr2/fs/st.default/st/stlib/stm_util.c (共享内存实用程序)、setup_st.c (共享内存设置程序) 等实用程序和减少打字输入，可在 prog 帐户下把该目录下的/include、/stalloc、/stlib 子目录里的内容拷贝到/usr2/st 对应的子目录里。

⑵ 根据本地站程序所用共享变量的多少，在/usr2/st/include/stparams.h 里定义共享内存块的大小，如 4096 或 8192 等。增量为 4096 字节。

⑶ 将本地站程序所用的共享变量都定义在名为/usr2/st/include/stcom.h 的文件里 (stcom 结构)。

⑷ 用“extern struct stcom *stm_addr;”定义指针到 stcom 结构

⑸ 调用 setup_st() 程序来连接共享内存区域。如：

```
#include "../include/stparams.h"
#include "../include/stcom.h"
……
struct stcom *st;
……
setup_st();
```

⑹ 在 prog 帐户下：

```
cd /usr2/st/stlib
make    (编译装配 stlib 共享内存实用程序)
cd /usr2/st/stalloc
make   (编译装配 stalloc 共享内存设置程序)
```

编译装配 stqkr、antcn 等所有用到本地站共享变量的程序。所有本地站程序编译装配后

的立即可执行代码都放在/usr2/st/bin 目录里。

⑺ 重启 FS 计算机。/etc/init.d/fs 指向/usr2/fs/misc/rc.fs，rc.fs 里有启动运行 /usr2/st/bin /stalloc 的命令。只有重启后，新的共享内存设置才成立。注意：因有的共享变量参数变动后对 stlib 里的程序有影响，所以为确保万无一失，一旦与本地站共享变量有关的参数变动后，请重新执行步骤⑹和⑺。如果共享内存设置正常，计算机重启时会显示类似“Stm_get:id=1, size is 4096(或 8192 等)bytes ……”的信息。

本地站程序如调用到 FS 里的通信程序，如 matcn、mcbcn，则必须考虑是否会影响到 FS 程序的运行，即是否可能出现与 FS 里的程序同时调用 matcn、mcbcn 的情况。通常要求每次调用的执行时间周期控制在 1 s 之内。大于 1 s 的命令可将其拆开成 2 条，第 1 条为带参数的设置命令，第 2 条为获取回答信息的命令。对于要花较长时间才能完成的命令，也可单独编一个与 stqkr 并行的程序，如上述 stpgm.ctl 里的 sendl 程序。sendl 程序运行后通过 FS 的共享变量与 FS 沟通信息。

增加非设备控制类的 Snap 命令方法与上述类似，只是不用编写设备通信接口程序。

⑥ 站程序执行某 snap 命令的出错处理

站程序需要报告执行结果给 FS。如执行正常，在 ip[2] 里放“0”。如有错，在 ip[2] 里放出错号码，该号码必须是对应 sterr.ctl 里的一个错误码；并在 ip[3] 里放字符串“st”，表示该错误码对应 sterr.ctl 文件。FS 通过 ip 数组可识别运行是否有错，并将有关信息输出到屏幕和 log 文件。

另外，如要用 FS 控制本地站天线还需自编 antcn.c 程序, 因为 FS 不可能把世上不同接口约定的所有天线的控制命令放入一个 antcn 程序里。用 FS 控制本地站天线涉及到 antcn 程序、fspgm.ctl、flagr.ctl、dev.ctl 文件和天线计算机的控制软件。

FS 通过 antcn 程序控制观测站的天线。该程序将来自 FS 的天线控制命令，如 source=xxx …、 onsource、track、radecoff=xxx …、azeloff=xxx …等，传给天线控制计算机，然后再接收天线的响应信息回传给 FS 软件。antcn 是 /usr2/fs/control/fspgm.ctl 里的联机程序之一，随 FS 一起启动。/usr2/control/flagr.ctl 文件定义的参数是天线状态检查时 FS 与 antcn 的通讯周期 (一旦发送过 source=xxx 命令，FS 会定时询问天线是否跟上了源，然后在屏幕和 log 文件里显示出来)。该参数通常定义为 2 或 5 s。/usr2/control/dev.ctl 文件定义天线计算机与 FS 计算机的通讯接口线，如“/dev/ttyS4”。天线计算机的控制软件与 FS 的 antcn 程序互通信息，并控制天线实际动作。

antcn.c 里通常有以下 7 种工作模式，最重要的是“1”,“2”,“3”,“5”和“7”模式。

```
/* IP(1)   = mode
      0    = initialize LU
      1    = pointing (SOURCE command)
      2    = offset (RADECOFF commands)
      3    = on/off source status (ONSOURCE command)
      4    = direct communications (ANTENNA command)
      5    = on/off source status for pointing programs
      6    = reserved for future focus control
      7    = log tracking data (TRACK command) ·····
```

模式“0”：一般做通讯接口和软件参数的初始化，然后将 FS 的“ionsor”共享变量置为 0(即当前未对准源)。注：antcn 仅可修改“ionsor”共享变量，对其它 FS 的共享变量只能读不能写。

模式“1”：通过 FS 的共享变量 lsorna[10] 、ra50、dec50、ep1950、cwrap 获得 source 命令的源名称 (最多 10 个字符)、赤经 (弧度)、赤纬 (弧度)、历元、顺时针/逆时针转 (0 或 1)，然后格式化这些信息成天线控制软件能识别的格式并传给它。这些信息传给天线控制软件后 antcn 并不等它完成，而是在置 ionsor 为 0 后立即返回挂起。FS 里另有程序会周期性询问天线是否跟上了。将 ionsor 置为 0 表示在准备跟踪新源，当前还未对准源。

模式“2”：通过 FS 的共享变量 RAOFF、DECOFF、AZOFF、ELOFF 获得 radecoff 或 azeloff 命令的偏移参数 (单位：弧度)。格式化这些信息成天线控制软件能识别的格式并传给它。这些信息传给天线控制软件后 antcn 并不等它完成，而是在置 ionsor 为 0 后立即返回挂起。

模式“3”：发送“onsource”命令给天线控制软件并立即回收天线的回答，根据回答内容置 ionsor 为 1(对上源了) 或 0(未对上源)，然后返回挂起。误差小于 0.1 个波束宽度为对上源, 或者另外定义一个适用各波段的统一误差阀。

模式“4”：发送“ANTENNA=xxx ···”命令给天线控制软件。本命令可让站里自行定义需要的任意控制内容。

模式“5”：本命令与模式“3”基本相同，唯一的差别是并不将天线的信息写入 log 文件。本模式主要用在天线指向程序里。

模式“7”：发送“track”命令给天线控制软件，并立即回收天线的回答。本命令与模式“3”类似，但要求得到更多的天线状态信息，通常要求回收天线计算机当前的时间、赤经命令位置、赤纬命令位置、历元、赤经命令偏置、赤纬命令偏置、天线方位 (当前实际位置，地平坐标)、天线俯仰 (当前实际位置，地平坐标)、方位偏差、俯仰偏差、当前时刻电缆转绕位置 (0～ 外圈，-270°；1～ 内圈，+270°) 等参数。

antcn 编程需要用到 FS 的共享变量，其连接、使用方法请参阅上述内容。如用到站共享变量也如上述。antcn 的具体编程可参考各站现有 antcn 目录下的程序。

FS 包含有天线的指向和灵敏度测量软件包 [21−24]，再结合站里的 antcn，观测站操作员可方便地进行自动化天线测量。

4 结束语

FS 软件功能强大，各类终端系统做 VLBI 实验所需的功能都齐备，还有许多辅助功能，如 log 文件检查与画图 [25]、相位信号提取支持、各种监视等，观测站程序员增加自己的软件也比较方便。这些都极大地方便了 VLBI 用户。它的缺点 (或说不完美的地方) 是包含的东西多了点，软件弄的很庞大。老的程序不变 (即使不用了，如磁带机控制等)，又不断地新增内容，把许多的程序都弄到 1 个软件里。这样有时会不方便。比如，当其中的某个程序发现有 bug 时，就需升级整个 FS 版本。当有人想要了解或修改某程序时，有时需花很多时间去找其中的子函数，或共享变量的出处与使用此变量的各有关程序。

最后，建议学习本地站程序编写的人员，从 fscmd.ctl 和 stcmd.ctl 定义开始再读一下 bbc(或 ifd)、stqkr 程序，这样可了解 FS 和本地站程序更加具体、清晰的编程过程。

参考文献：

[1] Himwich W E. Field System programs，NVI, Inc./GSFC, 1993
[2] Himwich W E. Field System Architecture，NVI, Inc./GSFC, 1993
[3] Mujunen Ari, Himwich W E, Vandenberg N R. Computer Reference，EVN, NVI Inc./GSFC, 1995
[4] Vandenberg N R. SNAP language, Crustal Dynamics Project/GSFC, 1989
[5] Himwich W E, Vandenberg N R. Field System Log File Format，NVI, Inc./GSFC, 2006
[6] Himwich W E, Vandenberg N R. SNAP Commands, NVI, Inc./GSFC, 2007
[7] Himwich W E. SNAP Command Subroutines，NVI, Inc./GSFC, 1993
[8] Himwich W E. Operational use of time in the FS ，NVI, Inc./GSFC, 2008
[9] Vandenberg N R. MAT Communications Protocol，NVI, Inc./GSFC, 1993
[10] GPIB-232/485CT-A User Manual & Programming Guide,NI Cor. 1999
[11] Vandenberg N R. MCB Protocol，NVI, Inc./GSFC, 1993
[12] Clark B G. Monitor and Control Bus at VLBA Stations, NRAO VLBA Project 1984
[13] Vandenberg N R. MCB Protocol Reference Manual, Space Geodesy Project, 1993
[14] Himwich W E. Station Programs, NVI, Inc./GSFC, 1993
[15] Himwich W E, Vandenberg N R. Control Files and Field System Initialization，NVI, Inc./GSFC, 1993
[16] Himwich W E, Vandenberg N R. Standard Procedures，NVI, Inc./GSFC, 1993
[17] Himwich W E, Vandenberg N R. Field System Common/Shared Memory，NVI, Inc./GSFC, 1993
[18] Himwich W E. Gregorich G A. Error Messages, NVI, Inc./GSFC, 1992
[19] 薛祝和. 上海天文台年刊, 2003，24：118
[20] 薛祝和. 上海天文台年刊, 2009，30：168
[21] Himwich W E. Antenna Calibration Programs，NVI, Inc./GSFC, 1993
[22] Himwich W E. Antenna Calibration Programs User Guide，NVI, Inc./GSFC, 1993
[23] Himwich W E. Pointing Model Derivation，NVI, Inc./GSFC, 1993
[24] Vandenberg N R. Antenna Performance，EVN, NVI Inc./GSFC, 1993
[25] Lindenau Fredrik. logpl: Plot Log Data，NVI, Inc./GSFC, 1997

The Controlling of the Station Non-standard Equipments in FS

XUE Zhu-he

(Shanghai Astronomical Observatory, Chinese Academy of Sciences, Shanghai 200030)

Abstract: This paper provides a description of the basic features of Field System software (FS). And the working flow of standard equipments controlling in FS is described. The steps and method for controlling station non-standard equipments are described also. This paper supplies reference to the station programmers of China VLBI network.

Key words: astronomical facilities and technique; Very Long Baseline Interferometry (VLBI); control software