



Robert C. Byrd Green Bank Telescope NRAO Green Bank

Toney Minter

3rd December 2001
GBT SOFTWARE PROJECT NOTE 17.2

Creating A Device Setup Glish Script

HTML version Available¹

Contents

1	Introduction	2
2	Device Parameter Lists	2
3	Device Glish Script Template Generation	2
4	Device Setup Glish Script Template	3
5	Pre-observing Review	5
6	Pre-observing Testing	5
7	Commissioning Tests	6
8	Final Approval	6
9	Accessing Glish Scripts from a GO Table	6

Abstract

This document is intended for a commissioner who wants to add a new device setup glish script to the GBT's observer interface (GO). The methodology of how to create a device setup glish script is covered. A review process for inclusion of the glish script into the Monitor and Control software system is also presented.

¹<http://www.gb.nrao.edu/GBT/MC/doc/dataproc/gbtGOglisScripts/gbtGOglisScripts.html>

1 Introduction

This document is intended for a commissioner who wants to add a new device setup glish script to the GBT's observer interface (GO). The device setup glish scripts will be checked into the GBT Monitor and Control system under CVS control only after the scripts have been thoroughly tested and approved. Only approved scripts will be made available for including using a GO Table.

Each glish script will contain commands to setup one and only one device. The scripts should allow the complete setup of the device into a known and predictable state. The glish scripts will also print out if they succeeded or failed the setting of the parameters.

2 Device Parameter Lists

The first step in creating a device setup glish script is to know what parameters are available to be set in the device manager. You can obtain a device parameter list using the following formula:

```
source /home/gbt/gbt.bash
glish -l getDeviceParameters.g
```

Note: If you are using a shell other than bash then you replace bash by csh or sh in the "source" line. Then once glish has started:

```
deviceInfo('XXX')
deviceInfo('XXX', 'filename')
```

The first deviceInfo() command sends the output to the terminal screen. The second deviceInfo() command will send the output to the desired filename. You should replace XXX with the desired device name. If you want the output to go to a specified file you should replace filename with the full path and filename of the desired output file.

deviceInfo() will list all parameters that an commissioner could be interested in setting (actually more than they typically need to set) except for the Spectral Processor. The Spectral Processor Manager does not yet conform to the Monitor and Control system standard for manager/sub-manager interactions. Anyone interested in creating a device setup glish script for the Spectral Processor should contact Toney Minter for further information.

Setup commands for all parameters should be included in the device setup glish script. Each parameter should be well documented with all possible legal values listed. Following these two instructions will allow previously accepted device setup glish scripts to act as a starting point for new device setup glish scripts.

3 Device Glish Script Template Generation

An alternative way of determining the parameter lists for a device is to use the deviceGlishScript() function in getDeviceParameters.g. This function will create a "first attempt" device setup glish script. It will provide some documentation including possible values for enumerated data types. This function is not guaranteed to be completely correct and its outputs should be thoroughly checked over. Parameters will also have to be moved around in the created file to match the template defined below.

To use this function follow this formula:

```
source /home/gbt/gbt.bash
glish -l getDeviceParameters.g
```

Note: If you are using a shell other than bash then you replace bash by csh or sh in the "source" line. Then once glish has started:

```
deviceGlishScript('XXX')
deviceGlishScript('XXX', 'filename')
```

The first deviceGlishScript() command sends the output to the terminal screen. The second deviceGlishScript() command will send the output to the desired filename. You should replace XXX with the desired device name. If you want the output to go to a specified file you should replace filename with the full path and filename of the desired output file.

4 Device Setup Glish Script Template

It is important that the template defined below be followed to ensure that variables in GO are not overwritten or corrupted when a device setup glish script is used.

First, all the parameters should be set in a function. The name of the functions should be the same as the name of the glish file (minus the .g of course). In the template below the file name is rcvr1_2setup.g and the function name is rcvr1_2setup.

The first if statement in the function makes sure that the glish session can talk with the device manager.

Next, the parameters that are to be changed are listed. Comment statements either before the lines of code setting the parameter or on the line setting the parameter should indicate the valid values for the parameter.

Parameters which provide monitor information or or automatically are set by another device (such as the scan coordinator) are put in the next section. The code in this area should also be commented out.

Secondary parameters which will rarely be changed from their default values will be listed in the next section of code. Once again this section of code should have all its lines of code commented out.

The final line of code in the function should be device.prepare() function call which will actually perform the changes requested for the device.

A version of the checklist will be contained in the header of the glish file to help keep track of its history. This will help ensure that the correct version of the file is included into the Monitor and Control system.

```
# 1.15-1.75 GHz Receiver setup from glish for XXXXXXXX

# checklist information
# requestor's name: Rick Fisher
# Device: Rcvr1_2
# Description: setup the 1.15-1.75 GHz Receiver for default standard observing
# Date Started: Dec 26, 2000
# Date parameter list obtained: Dec 26, 2000
# Date of completion of initial version: Dec 26, 2000
#
# Date of review: Jan, 2 2001
# Reviewer: Dana Balser
# Comments: Forgot segmentEnable, linCircPhaseShift doesn't need to be set
#           for this particular setup
# Accepted by Dana Balser on January 5, 2001
#
# Date of non-observing test: Jan 7, 2001
# Name of tester: Ron Maddalena
# Comments: worked ok except for loOrHiCalSel - should use lowCal
```

```

# Accepted by Ron Maddalena on Jan 9, 2001
#
# Date of observing test: Jan 11, 2001
# Name of commissioner: Glen Langston
# Comments: OK
# Accepted by Glen Langston on Jan 11, 2001
#
# Date of final review: Jan 15, 2001
# Name of reviewer: Mark Clark
# Comments: function name did not match file name
# Accepted by Mark Clark on Jan 17, 2001
#
# Included into Monitor and Control on Jan 20, 2001
#

rcvr1_2_setup := function ( )
{
  if (!is_defined('thisdev')) {
    thisdev := ygor('Rcvr1_2');
  }

  # set parameter for error checking
  ok := T;

  ok := ok & thisdev.setParameter('cpuLoCalPwrSw', 'swOn'); # swOff, swOn
  ok := ok & thisdev.setParameter('loOrHiCalSel', 'lowCal'); # lowCal, highCal
  ok := ok & thisdev.setParameter('xlExtToMCBCtrlSel', 'ctlExt'); # ctlExt, ctlMcb
  ok := ok & thisdev.setParameter('yrExtToMCBCtrlSel', 'ctlExt'); # ctlExt, ctlMcb
  # following are option for rightIfFilterSwitch and leftIfFilterSwitch
# 1 = 1.10-1.80 GHz
# 2 = 1.60-1.75 GHz
# 3 = 1.30-1.45 GHz
# 4 = 1.10-1.45 GHz
# 5 = Spare
  ok := ok & thisdev.setParameter('rightIfFilterSwitch', 3);
  ok := ok & thisdev.setParameter('leftIfFilterSwitch', 3);
  # following are options for polarizationSelect
# polLinear, polCircular
  ok := ok & thisdev.setParameter('polarizationSelect', 'polLinear');
  ok := ok & thisdev.setParameter('xferSwitch', 'tsThru'); # tsThru, tsCrossed
  ok := ok & thisdev.setParameter('xferSwCtlMode', 'ctlMcb'); # ctlExt, ctlMcb
  ok := ok & thisdev.setParameter('linCircPhaseShift', 45.35);
  ok := ok & thisdev.setParameter('raCryoAmp', -3);
  ok := ok & thisdev.setParameter('lbCryoAmp', 2);
  ok := ok & thisdev.setParameter('yrCPUNoiseSwCtrl', 'swOff'); # swOff, swOn
  ok := ok & thisdev.setParameter('xlCPUNoiseSwCtrl', 'swOff'); # swOff, swOn
# refrigOff, refrigHeat, refrigCool, refrigPump
  ok := ok & thisdev.setParameter('cryoState', 'refrigCool');
  ok := ok & thisdev.setParameter('raBiasSwitch', 'swOn'); # swOff, swOn
  ok := ok & thisdev.setParameter('lbBiasSwitch', 'swOn'); # swOff, swOn
  ok := ok & thisdev.setParameter('segmentEnable', F);

## Monitor/Auto Parameters:
# ok := ok & thisdev.setParameter('cryoMonitorRate', 'mr2Sec');

```

```

#   ok := ok & thisdev.setParameter('cryoStatusMonitorRate', 'mr2Sec');
#   ok := ok & thisdev.setParameter('cryoCtlMonitorRate', 'mr30Sec');
#   ok := ok & thisdev.setParameter('switchStatusMonitorRate', 'mr200MS');
#   ok := ok & thisdev.setParameter('noiseSourceMonitorRate', 'mr2Sec');
#   ok := ok & thisdev.setParameter('loPowerMonitorRate', 'mr2Sec');
#   ok := ok & thisdev.setParameter('gregorianMonitorRate', 'mr1Min');
#   ok := ok & thisdev.setParameter('supplyMonitorRate', 'mr2Sec');
#   ok := ok & thisdev.setParameter('cryoAmpMonitorRate', 'mr2Sec');
#   ok := ok & thisdev.setParameter('biasSwitchMonitorRate', 'mr30Sec');
#   ok := ok & thisdev.setParameter('vacIonPumpMonitorRate', 'mr30Sec');

## Secondary parameters:
#   ok := ok & thisdev.setParameter('calStateCntl', 0);
#   ok := ok & thisdev.setParameter('biasPwr', 0);
#   ok := ok & thisdev.setParameter('receiverSegment',
# [segmentLength=[seconds=0.02, MJD=0, flags=0, refFrame=1, units=1],
# highCalPower=F, lowCalPower=T, rcpNoiseExt=T, lcpNoiseExt=T,
# noiseStateL=T, noiseStateR=T, beamXferSwABL=F, beamXferExt=T,
# beamXferSwABR=F, beamXferSwCDL=F, beamXferSwCDR=F]);
#   ok := ok & thisdev.setParameter('stopTime',
# [seconds=0, MJD=0, flags=0, refFrame=1, units=1]);
#   ok := ok & thisdev.setParameter('polXferExtSigRefCtl', 6); # 0:255

thisdev.prepare();
thisdev := [=];

if (ok) {
    print ``rcvr1_2_setup.g was successful``
} else {
    print ``rcvr1_2_setup.g FAILED``
}

return ok;
}

```

5 Pre-observing Review

Before the device setup glish script is used on any part of the GBT system or the Monitor Control Mockup system the script should be reviewed informally by a knowledgeable NRAO GB astronomer. The purpose of this review is to make sure that the script is complete and will setup the device as specified. This step should also help reduce the number of bugs in the glish script.

6 Pre-observing Testing

The pre-observing test should be carried out using the Monitor and Control system and is aimed at making sure that all the commands in the glish script are valid. This provides another round of debugging the glish script. The author of the glish script should sign-up for GBT test time at the GBT Equipment Reservations web page (<http://arcturus.gb.nrao.edu/cgi-bin/we3.4/webevent.cgi>). The scripts should be tested only during the requested test time. No scans or observations should be done during these tests. The goal of this test is to make sure that the device gets setup correctly from many and various initial conditions for the device. This will also test the glish

script for completeness, i.e. will the script cause the device to be setup the same way every time for the desired observation.

7 Commissioning Tests

This test will be run by a commissioner (other than the author). The commissioner will use the device setup glish script to configure the GBT device for the desired type of observation. Observations will then be carried out and the ensuing data will be reduced to confirm that valid data were obtained in the desired fashion.

8 Final Approval

Once the device setup glish script is fully tested and ready for inclusion into the Monitor and Control system it will undergo one final review. This review will be carried out by several members of the Monitor and Control group. The purpose of this review is to check the format of the glish script to make sure that it can be included into the Monitor and Control system.

9 Accessing Glish Scripts from a GO Table

The following demonstrates how glish scripts may be used in setting up the GBT for an observation using a GO Table. The GO Table might appear as follows:

```
#
# setup the hardware for the observations using a standard setup
#
# setup Rcvr1_2 for HI, 2 polarization observations
include rcvr1_2_1420mhz_2pol.g
# setup LO1 for HI, 2 polarizations
include lo1_1420mhz_2pol.g
# setup ConverterRack for HI, 2 polarizations
include converter_rack_1420mhz_2pol.g
# setup AnalogFilterRack for HI, 2 polarizations
include analog_filter_rack_1420mhz_2pol.g
# setup ScanCoordinator for HI, 2 polarizations, total power with cal
include scan_coordinator_rack_1420mhz_2pol_tpwcal.g
# setup Spectrometer for 64,000 channels, 2 polarizations,
include spectrometer_rack_64000channels_2pol.g
#
# now tweak the setup for my observations
#
....
#
# setup complete, now define the observations
#
....
```