WBDC User's Guide

T. B. H. Kuiper

Revision of March 1, 2015

Preface

Chapter 1, Introduction, is a quick overview of WideBand DownConverter receiver class and its capabilities.

Chapter 2, Command Line, gives basic commands for direct control of the equipment but also gives a simple, high-level view of the software organization.

Chapter 3, Monitor and Control System, provides the details of the monitor and control system, such as component addresses and latch[1] logic states.

Chapter 4, Configurations, explains how a WBDC is one element, the `Receiver`, in an overall observing system. Configurations describe how the signals flow between these elements.

A convention used in this document is that if an object is shown capitalized and in typewriter font, it is also a Python *class*[2]. In general, a single word in typewriter font refers to an attribute while one followed by empty parentheses is a method.

The most basic class is `Device` which has the attributes `name`, `inputs` and `outputs` which are instance of `Port` classes, and `data` about the device, such as the location of a telescope or the bandwidth of a receiver. Most classes discused here are sub-classes of `Device`, which means that they "inherit" these attributes.

In the hope that it is more enlightening than confusing, Snippet 0.1 (page ii) gives an example of a class definition. The `__init__()` method creates an instance of this class, which inherits attributes from the class `Device`. The second line creates the object and the rest of the code assigns values to its attributes.

[1] A latch is an electronic logic device that retains the state (0 or 1) assigned to it.

[2] A class is a programming unit that serves as a template for objects that are similar to each other. It has "attributes", such as variables, and "methods" (functions). A class is a natural way of describing a real-world object which has properties and can do things.

```
class Telescope(Device):
  def __init__(self, obs, dss=0, LO=None, active=True):
    name = "DSS-"+str(dss)
    Device.__init__(self, name)
    self.inputs = {obs.name:obs}
    self['longitude'], self['latitude'], self['elevation'], tz, name, diam = \
            get_geodetic_coords(dss=int(dss))
    self['geo-x'],self['geo-y'],self['geo-z'] = \
            get_cartesian_coordinates('DSS '+str(dss))
    self.outputs[self.name] = Port(self, self.name, signal=Beam(str(dss)))
    self.outputs[self.name].signal['dss'] = dss
```

Snippet 0.1: Stripped-down definition of the `Telescope` class.

Contents

1 Introduction **1**
- 1.1 Hardware Overview . . . 1
  - 1.1.1 Input Switching . . . 1
  - 1.1.2 Polarization Conversion . . . 1
  - 1.1.3 Sideband Separation . . . 2
- 1.2 Monitor and Control . . . 3
  - 1.2.1 LabJack 1 . . . 3
  - 1.2.2 Labjack 2 (and 3) . . . 3

2 Command Line **5**
- 2.1 High-level Commands . . . 5
  - 2.1.1 Feed Switch Control . . . 5
  - 2.1.2 Polarization Mode . . . 6
  - 2.1.3 Down-conversion Mode . . . 6
  - 2.1.4 Analog Monitoring . . . 7
  - 2.1.5 Other . . . 7
- 2.2 Mid-level Commands . . . 7
  - 2.2.1 Feed Switch Control . . . 7
  - 2.2.2 Polarization Mode . . . 8
- 2.3 Low Level Commands . . . 9
  - 2.3.1 LabJacks . . . 9
  - 2.3.2 Digital Monitor and Control . . . 9
  - 2.3.3 Attenuator Control . . . 10

3 Monitor and Control System **11**
- 3.1 Motherboard Control . . . 11
  - 3.1.1 Digital Module 1 . . . 12
  - 3.1.2 Digital Module 2 . . . 14
  - 3.1.3 WBDC1 Analog Monitoring . . . 14

3.1.4 WBDC2 Analog Monitoring 14

4 Configurations **17**

A LabJack Configuration **19**
A.1 LabJack 1 19
A.2 LabJack 2 (and 3) 19
A.3 Configuration Check 19

List of Figures

1.1 Interior of WBDC2. 2

List of Code Examples

0.1 Stripped-down definition of the `Telescope` class. ii
2.1 Sensing and controlling the transfer switch. 6
2.2 Sensing and controlling the polarization mode. 6
2.3 Sensing and controlling the down-converter mode. 7
2.4 Sensing and controlling the crossover switch directly. 8
2.5 Sensing the state of the individual crossover switches. 8
2.6 Control of individual polarization sections 8
2.7 Commands for sensing and setting a down-converter I/Q hybrid. 8
2.8 Setting a latch group. 9
2.9 Two ways of instantiating and setting a DAC. 10
4.1 Configuration used by `wbdc2_prompt.py` 18
A.1 Commands for checking the LabJack I/O configuration. 20

Acronyms and Technical Terms

ADC - analog-to-digital converter.
AIN - LabJack® analog input port.
DAC - digital-to-analog converter.
DIO - LabJack® digital I/O.
IF - intermediate frequency (signal)
I/O - input/output
I/Q - in-phase/quadrature-phase, the two components of a complex signal.
K-band - the frequency range 18-26.5 GHz.
LO - local oscillator.
L/U - lower sideband/upper sideband, a signal pair obtained from I/Q.
MSB - most significant bit (or byte, depending on context).
PIN - P-type/insulator/N-type, a type of diode.
RF - radio frequency (signal)
WBDC - wide-band down-converter, a class of receiver.

Chapter 1

Introduction

1.1 Hardware Overview

A Wide-Band Down-Converter processes dual-polarization K-band signals from two feed horns. There are two versions of the WBDC. WBDC1 has two dual-polarization radio frequency sections after the band-splitting filters, one for 22 and 24 GHz, two switch-selectable local oscillators and one intermediate frequency section. WBDC2 has five dual-polarization RF sections and 20 IF sections (2 feeds $\times$ 2 polarizations $\times$ 5 bands).

Figure 1.1 shows the interior of WBDC2. The motherboard is on the lid. The RF enters through four SMA bulkhead feed-throughs on the left. The IFs exit on the right.

The feed switching, polarization selection, and down-conversion are basically the same in both WBDC designs. All functions are controlled by a motherboard through LabJack® general-purpose I/O interfaces. The WBDC1 LabJacks are accessed through a USB hub by an external computer. WBDC2 LabJacks are controlled with a Raspberry Pi embedded computer accessed via ethernet.

1.1.1 Input Switching

The first stage of the WBDC allows the two dual-polarizationdown-converters to be switched between the two feeds. The switches are the two modules in the center to the far left in Figure 1.1.

1.1.2 Polarization Conversion

The four RF signals are split into five bands each. The splitters are the larger blue modules at the top and bottom on the far left on Figure 1.1. After the band-selecting RF filters (row of modules to the left of center), the linearly polarized signals from each feed can be switched into a quadrature hybrid (to the right of the black DB9 connectors) to be converted to circular

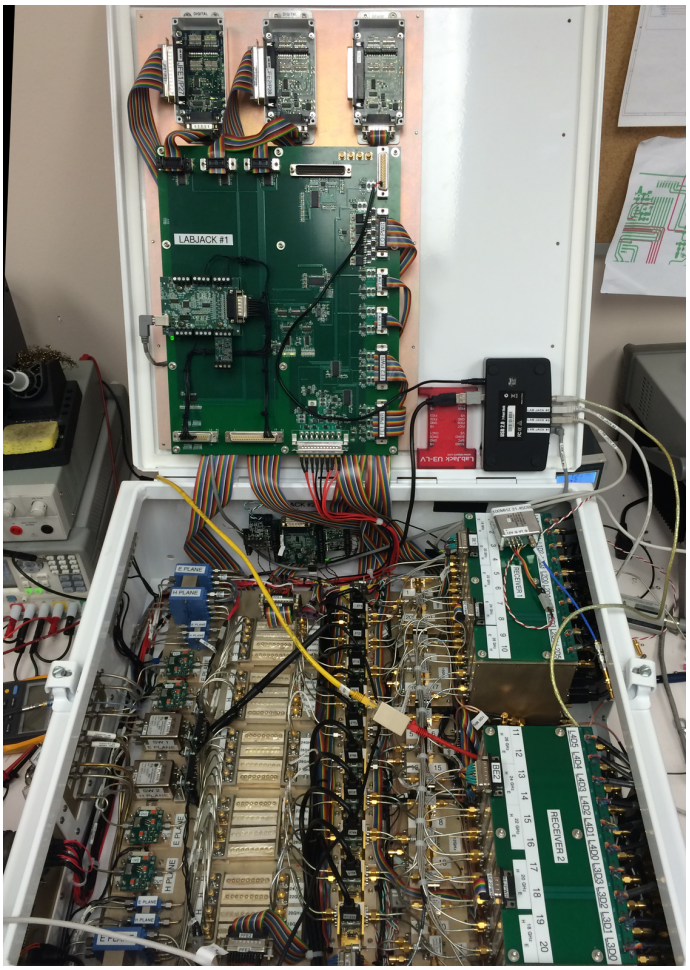

Figure 1.1: Interior of WBDC2.

polarization. PIN diode attenuators are provided (under the black connectors) to balance the signals into these hybrids.

1.1.3 Sideband Separation

In the closed modules on the far right are the down-converters, which have complex mixers putting out two IFs from 0 to 1000 MHz, with a 90° phase difference. Both IFs go from -1000 MHz to +1000 MHz but the negative and positive frequencies are mixed together. These may be considered the real and imaginary components of a complex signal. There are switches

which can direct each I/Q pair into a quadrature hybrid to convert them to an upper and lower sideband pair.

1.2 Monitor and Control

Monitor and control is done with two or three LabJack® general purpose I/O devices.

1.2.1 LabJack 1

This LabJack with local ID 1 (on the center left of the motherboard) controls the WBDC motherboard. The motherboard has groups of eight latches which are used to control and sense switches, and to select analog monitoring points. The LabJack's ADC channels read currents, voltages, and temperatures.

1.2.2 Labjack 2 (and 3)

LabJack 2 (and 3 for WBDC2), on the box walls at the ends of the row of filters, control PIN diode attenuators. There are four voltage-controlled RF attenuators (PIN diodes) for each down-converter sub-band. The voltages are generated in LabJack TickDACs attached to the terminals of LabJack 2 (and 3 for WBDC2).

Chapter 2

Command Line

The program `wbdc2_prompt.py`[1] provides the user with a direct Python command line interface in a terminal session on the embedded Raspberry Pi. It simply configures the hardware and software as described in Chapter 4.

The programming style used is "object oriented", so that a command takes the form `object.method()` where `object` is an "instance" of a "class'. In this implementation, classes may have private classes. Think of systems and subsystems, so that a command can also have the form `parent.child.method()`. Classes can also have instances of other public classes as "attributes", which leads to the same command structure. In the program `wbdc2_prompt.py`, `receiver` is an instance of the `WBDC2` class.

The Python `help()` command can be very useful at the command line. It can be invoked at any level of the hierarchy. These examples

```
help(receiver)
help(receiver.get_crossover)
help(receiver.crossSwitch)
help(receiver.crossSwitch.get_state)
```

will become clearer below.

2.1 High-level Commands

2.1.1 Feed Switch Control

The WBDC class of receivers have two feed horns with two linear polarizations, "E" and "H". There are four input channels called "F1E", "F1H", "F2E" and "F2H". These feed a pair of switches through which the receivers connect to the feeds. In the default state (logic state

[1] In `/usr/local/lib/python2.7/DSN-Sci-packages/MonitorControl/Receivers/WBDC/apps`.

0), "F1E"→"R1E" and "F1H"→"R1H". In the *set* state (logic state 1), "F1E"→"R2E" and "F1H"→"R2H". The commands for this are shown in Snippet 2.1. The keyword `crossover` is

```
receiver.get_crossover()
receiver.set_crossover(crossover=False)
```

Snippet 2.1: Sensing and controlling the transfer switch.

not required in this case but is mnemonically useful. The value `True` sets the switch.

2.1.2 Polarization Mode

Each of the four receiver sections feeds five bandpass filters. "Band 18" goes from 17 to 19 GHz, "Band 20" from 19 to 21, and so on to "Band 26" from 25 to 27 GHz. There are ten sections which can convert the linearly polarized modes "E" and "H" produced by the front end to circularly polarized modes "L" and "R".

The command for sensing and controlling the polarization mode are given in Snippet 2.2. `receiver.get_pol_modes()` returns a Python `dict` (short for "dictionary") which is a key:value

```
receiver.get_pol_modes()
receiver.set_pol_modes(circular=False)
```

Snippet 2.2: Sensing and controlling the polarization mode.

array with the states of all the polarization sections. The command `receiver.set_pol_modes()` sets all the polarization sections in the entire receiver to the designated state. The default state, that is, if `set_pol_modes()` is invoked without specifying an argument, is `False`. This command automatically invokes `get_pol_modes()` as its last step.

2.1.3 Down-conversion Mode

The outputs from the polarization sections go into down-converters which have names like "DC['R1-18P1']" (receiver 1, band 18, polarization 1) which reflects the fact that the polarization is now ambiguous. "P1" can be "E" or "L" while "P2" can be "H" or "R" depending on the state of the polarization section providing the signal.

There are quadrature hybrids which may be used to convert the in-phase and quad-phase IFs to lower sideband (-1000 to 0 MHz) and upper sideband (0 to 1000 MHz) IFs. The default state (logic 0) of these sections is L/U mode; the optional I/Q mode is logic state 1.

The commands for sensing and controlling the down-converter section IF mode is shown in Snippet 2.3. The first command returns a Python `dict` with the states of all the downconverters.

```
receiver.get_IF_mode()
set_IF_mode(SB_separated=False)
```

Snippet 2.3: Sensing and controlling the down-converter mode.

All the downconverters can be set to the same state with the command `set_IF_mode()`. It also returns a Python `dict` with the states of all the downconverters.

2.1.4 Analog Monitoring

The analog inputs to LabJack 1 are connected to various analog monitoring points. The power supply voltages, currents to various subsystems, temperatures at three locations and the power levels into the RF sections can all be monitored.

Analog inputs 0 and 1 return currents and voltages respectively. AIN2 and AIN3 return RF power and temperature. The address of the current and voltage monitor point is sent to latchgroup 1 at address 0. The address of the RF power and temperature is sent to latchgroup 1 at address 2.

The command `receiver.readAnalogs(latchAddress)` returns a Python dict with all the analog data available through that latch group.

2.1.5 Other

For completeness we note here one other high-level command which can be useful for debugging the equipment.

```
receiver.check_IO_config()
```

reports on the configuration of the WBDC2's three LabJack multi-purpose I/O devices. Section A.3 shows a typical output and the proper configuration of the LabJacks.

2.2 Mid-level Commands

2.2.1 Feed Switch Control

The crossover switch subsystem can be addressed directly. The commands to check and set the state of the cross-over switch are shown in Snippet 2.4. These have the same effect as the commands shown in Snippet 2.1.

It can happen that the two switches are not in the same state. Then `get_crossover()` will give an error message. The state of the individual switches can be queried with the commands in Snippet 2.5. The switches can be set individually with similar `set_state()` commands.

```
receiver.crossSwitch.get_state()
receiver.crossSwitch.set_state(True)
```

Snippet 2.4: Sensing and controlling the crossover switch directly.

```
receiver.crossSwitch['E'].get_state()
receiver.crossSwitch['H'].get_state()
```

Snippet 2.5: Sensing the state of the individual crossover switches.

2.2.2 Polarization Mode

The state of the individual polarization sections can be queried and set with the commands in Snippet 2.6. The *set* state (logic level 1) means that the conversion is made. The default state

```
receiver.pol_sec['R120'].get_state()
receiver.pol_sec['R120'].set_state(True)
receiver.pol_sec['R1-18'].atten['H'].set_atten(10)
```

Snippet 2.6: Commands to sense and set the mode of individual polarization sections and to balance their inputs.

is `False` (logic level 0). For the polarization sections to function properly, the two incoming signals must be balanced. For that purpose, each polarization section has two attenuators which can be adjusted. (The balancing procedure is described elsewhere.) The attenuation can be queried with a corresponding `get_atten()` command but it does not read the hardware. It only reports what the last `set_atten()` command did. The attenuation specified is relative to the minimum attenuation of the attenuator, which is in the range of 7 to 9 dB.

The state of a single down-converter hybrid can be queried and set with commands like those in Snippet 2.7.

```
receiver.DC['R1-18P1'].get_state()
receiver.DC['R1-18P1'].set_state(True)
```

Snippet 2.7: Commands for sensing and setting a down-converter I/Q hybrid.

2.3 Low Level Commands

2.3.1 LabJacks

At the heart of the monitor and control subsystem are two (or three) LabJacks. (They can also provide other functions, like timers, which we don't use.)

There are eight I/O lines available at screw terminals on the unit. These can all function as digital inputs or outputs, or analog inputs, according to the value of some configuration bytes. They are called FIO0-FIO7 when functioning as digital ports and AIN0-AIN7 when functioning as analog ports. In addition, the DB25 connector on the LabJacks provides an additional 12 digital IO lines, called EIO0-EIO7 and CIO0-CIO3. Each LabJack also has two analog outputs called DAC0 and DAC1, which we don't use.

Communication with the LabJack consists mainly of reading and writing bytes to the FIO, DIO and CIO groups.

2.3.2 Digital Monitor and Control

All the switches described above are controlled by latches. The latches are logically grouped, as shown in Table 2.1, with up to eight latches per group. Latch groups are read as bytes, as

Table 2.1: Latch Groups for Digital Monitor and Control

Latch Group	Function
X	controls the cross-over switches
R1P	controls the receiver 1 polarization sections
R2P	controls the receiver 1 polarization sections
PLL	monitors the cross-over switch state and LO phase lock
P1I1	pol 1 IF group 1 (bands 18 and 20)
P1I2	pol 1 IF group 2 (bands 22, 24 and 26)
P2I1	pol 2 IF group 1 (bands 18 and 20)
P2I2	pol 2 IF group 2 (bands 22, 24 and 26)

shown by the examples in Snippet 2.8. The bit assignment for each latch group is described in

```
In [7]: Math.decimal_to_binary(receiver.lg['P2I2'].read())
Out[7]: '00000000'
In [12]: Math.decimal_to_binary(receiver.lg['PLL'].read())
Out[12]: '01110100'
```

Snippet 2.8: Setting a latch group.

Subsection 3.1.1. There is a corresponding `write()` command which could be used in this way

```
receiver.lg['X'].write(int('00000011',2))
```

to set the cross-over switches.

2.3.3 Attenuator Control

The attenuators are PIN diodes controlled with a voltage (or more strictly, the current resulting from an applied voltage). The voltages are supplied by dual-channel ("A" and "B") digital-to-analog converters. The DACs are connected to a LabJack multi-function IO interface.

In the example in Snippet 2.9 an instance of a DAC is created and the A side is set to -2 V. The first way avoids creating an enduring instance of the DAC. However, it is pretty

```
from Receiver.Interfaces.LabJack import LJTickDAC

LJTickDAC(receiver.LJ[2],'R1-18')['A'].setVoltage(-2)

vs = LJTickDAC(receiver.LJ[2],'R1-18')['A']
vs.setVoltage(-2)
```

Snippet 2.9: Two ways of instantiating and setting a DAC.

cumbersome so the second way, which creates an instance of a DAC, although taking two steps, is usually preferred. And there might be need to use `vs` again.

The LJTickDAC is an "attribute" of the polarization section and so also be addressed as `receiver.pol_sec['R1-18'].tdac['A']`.

Using this command to control the attenuation requires knowledge of the attenuation *vs* control voltage curve. This is discussed elsewhere.

Chapter 3

Monitor and Control System

3.1 Motherboard Control

Motherboard signals are monitored and controlled with digital latches (logical devices which preserve a specified logic state). The latches are connected to serial-in/parallel-out registers. These registers are addressed using the EIO out bits of LabJack 1. The data are then clocked into the register using the SDI and SCK signals. Table 3.1 shows how the LabJack ports are used to control the motherboard signals. The state of NLOAD is normallly high. When checking status (or reading bits), NLOAD is toggled low for at least 10 mS then high to store the information for reading. CS-BUS is the global enable. The normal state is high. This is set to low when programming latches or reading data. It is set high when done.

LabJack 1 is capable of addressing 256 latches on a WBDC motherboard. They are grouped in 32 sets of 8 latches. The first four of the latch groups, at addresses 0–3, 8–11, 16–19, ..., 80–83, ..., 248–251, are writing to control bits. The second four of the latch group, at addresses 4–7, 12–15, 20–23, ...,84–87, ..., 252–255, are for reading from control or status bits (*e.g.* switch position indicators).

Latch group address bits A0-A7 (ports EIO0–EIO7 on LabJack 1) are used to address a specific 8-bit latch group for writing or reading. The latch data are bytes sent or read serially, MSB first. The latch group address encoded with the EIO bits consists of three parts:

EIO7-EIO3 encodes the motherboard digital module (DM = 1, 2, or 3) address.
EIO2 indicates write if 0 and read if 1.
EIO1-EIO0 along with EIO2, selects the latch group (LG = 1-4) in a digital module.

So latches in write-mode have addresses ending in 0–3. Latches in read-mode have addresses ending in 4–7.

The DM 1 bit address assignment differs for WBDC1 and WBDC2. DM 1 LG 1 has a EIO value (address) of 80 (0101 0000) in WBDC1 and 8 (0000 1000) in WBDC2. LG 2 in any DM has EIO0=1. WBDC2 DM 2 LG 1 has address 16 (0001 0000).

Table 3.1: Motherboard Signals and LabJack Ports

Labjack		**WBDC Motherboard**	
Name	**Channel**	**Name**	**Function**
AIN0	0	IMON	Analog input measuring supply currents
AIN1	1	VMON	Analog Input measuring supply voltages
AIN2	2	TEMP	Analog input measuring temperatures
AIN3	3	RF	Analog input measuring RF power
FI04	4		
FI05	5		
FI06	6		
FI07	7	SDO	Digital input from readback
EI00	8	A0	Latch address LSB
EI01	9	A1	..
EI02	10	A2	..
EI03	11	A3	..
EI04	12	A4	..
EI05	13	A5	..
EI06	14	A6	..
EI07	15	A7	Latch address MSB
CI00	16	SCK	Serial Clock input
CI01	17	SDI	Data input
CI02	18	NLOAD	Load data to/from latches
CI03	19	CS-BUS	Eable writing to or reading from latch

3.1.1 Digital Module 1

Seven latch groups are assigned to digital monitoring, that is, the program reads the latch states.

Feed Crossover Switch

The feed crossover switches are controlled with DM 1 LG 1 using bits L1A0 and L1A1 (A0, A1 on latchgroup 1): logic 0 for through and logic 1 for crossed. The commanded state can be read at the same bits (A0, A1 at LG address 12). However, the actual states of the cross-over switches are read at L4A0 and L4A1 (LG 3).

Polarization Hybrids

WBDC1 (to be provided).

WBDC2 receiver chain 1 polarization hybrids are controlled by latchgroups 2 and 3. The receiver chain 1 polarization hybrids are controlled by L2A0–L2A4. The receiver chain 2 polarization hybrids are controlled by L3A0–L3A4.

Band	Control Bits
18	L2A0 L3A0
20	L2A1 L3A1
22	L2A2 L3A2
24	L2A3 L3A3
26	L2A4 L3A4

Logic level 0 is for bypassing the hybrids, that is, linear polarization. Logic level 1 is for converting linear to circular, that is, X and Y polarization to L and R.

Local Oscillator Lock

Latch Group 4 monitors the state of the transfer switches and the local oscillator phase-locked loops

Pol	Status Bit
X	L4A0
Y	L4A1
Band	**Status Bit**
18	L4A2
20	L4A3
22	L4A4
24	L4A5
26	L4A6

Latch LEDs

The monitor bits should match the LEDs in the respective digital module. In WBDC1 the DM is to the top right With the hinge of the lid at the bottom, the LabJacks are upside down. The LEDs are in LSB -> MSB order and grouped as

```
LG 3 address 82 or 86    LG 4 address 83 or 87
LG 1 address 80 or 84    LG 2 address 81 or 85
```

If the box is mounted on a wall or ceiling and the lid is hanging down, the order is more conventional.

In WBDC2 there are three DMs at the top left of the lids, numbered from left to right. DM 3 is a spare. The (upside down) LED ordering is

```
LG addr 10 or 14   LG addr 11 or 15         LG addr 18 or 22   LG addr 19 or 23
LG addr  8 or 12   LG addr  9 or 13         LG addr 16 or 20   LG addr 17 or 21
```

3.1.2 Digital Module 2

This only applies to WBDC2. Four latch groups are used to control the I/Q hybrids. Receiver chain 1 I/Q hybrids are controlled by latch groups 1 and 2 (low address). Receiver chain 2 I/Q hybrids are controlled by latch groups 3 and 4 (low address). The two bypass switches for each hybrid are controlled with one bit. Logic level 1 is for IQ; logic level 0 is for LU. This is the bit assigment:

	Pol 1		Pol 2	
Band	Rec1	Rec2	Rec1	Rec2
18	L1A0	L1A1	L3A0	L3A1
20	L1A2	L1A3	L3A2	L3A3
22	L2A0	L2A1	L4A0	L4A1
24	L2A2	L2A3	L4A1	L4A2
26	L2A4	L2A5	L4A3	L4A4

3.1.3 WBDC1 Analog Monitoring

Latch addresses 0 and 2 address the registers used to select analog monitoring points. bit pattern sent to latch address 0 selects a current and a voltage to be connected to AIN0 and AIN1. A bit pattern sent to latch address 2 selects a thermistor to be connected to AIN2. AIN3 is not used in this receiver.

(Register addresses to be provided.)

3.1.4 WBDC2 Analog Monitoring

Latch 1 (Address 0)

This latch addresses the register used to select voltage and current monitoring points. Voltage points are selected using bits 0-2 and read at AIN1. Currents are selected using bits 3-6 and read at AIN0. The addresses can be ANDed to select any voltage point with any current point.

The monitor point address bits for voltage are shown in Table 3.2. The actual voltages are the byte read times the scale factor shown in Table 3.2

The monitor point addresses for currents are shown in Table 3.3. The actual currents are the value read offset by -0.026.

Table 3.2: Monitor Points for Voltages

Latch					
Address	**Bits**	**Input**	**Supply**		**Scale**
0	xxxxx000	AIN1	+6 V	digital	4.0211
0	xxxxx001	AIN1	+6 V	analog	4.0278
0	xxxxx010	AIN1	+16 V		10.5446
0	xxxxx011	AIN1	+12 V		10.5827
0	xxxxx100	AIN1	-16 V		-10.5446

Table 3.3: Monitor Points for Currents

Latch				
Address	**Bits**	**Input**	**Supply**	
0	x0000xxx	AIN0	+6 V	MB digital
0	x0001xxx	AIN0	+6 V	MB analog
0	x0010xxx	AIN0	-16 V	MB
0	x0011xxx	AIN0	+16 V	R1 FE
0	x0100xxx	AIN0	+16 V	R2 FE
0	x0101xxx	AIN0	+16 V	R1 BE
0	x0110xxx	AIN0	+16 V	R2 BE
0	x0111xxx	AIN0	+16 V	LDROs
0	x1000xxx	AIN0	+16 V	MB
0	x1001xxx	AIN0	+6 V	R1 FE
0	x1010xxx	AIN0	+6 V	R2 FE
0	x1011xxx	AIN0	-16 V	R1 FE
0	x1100xxx	AIN0	-16 V	R2 FE
0	x1101xxx	AIN0	-16 V	R1 BE
0	x1110xxx	AIN0	-16 V	R2 BE

Latch 2 (Address 1)

This latch selects temperature monitor point with bits 0–2 to be read at AIN3 and RF detector voltages with bits 3–6 to be read at AIN2. Table 3.4 gives the monitor address bits for temperature. The conversion from measured volts to temperature is

$$T = (V_{AIN3} + 0.2389275) * 23.549481$$

The monitor points for RF detector power are given by Table 3.5. The actual reading is (data-0.004)*2.0064.

Table 3.4: Monitor Points for Temperatures

Address	Bits	Input	Supply |	Location
1	xxxxx000	AIN3		R1 RF plate
1	xxxxx001	AIN3		R2 RF plate
1	xxxxx010	AIN3		BE plate

Table 3.5: Monitor Points for RF Detectors

Address	Bits	Input	Location
1	x0000xxx	AIN2	R1 E-plane
1	x0001xxx	AIN2	R2 E-plane
1	x0010xxx	AIN2	R1 H-plane
1	x0011xxx	AIN2	R2 H-plane

Chapter 4

Configurations

A WBDC is one element of a receiving system, specifically a `Receiver`. The receiver obtains its signals from a front end and delivers signals to a backend. A configuration is a description of the signal flow from the point where the antenna focuses radiation on a feed to the point where data are processed by a computer. At the very minimum, a receiver must know something about where its signal comes from and what properties were imposed by the preceding equipment.

Configurations are stored in the `DSN-Sci-packages` directory under `MonitorControl/Configurations`. The configuration used by the program `wbdc2_prompt.py` is shown in Snippet 4.1. This shows that the `Receiver` is a class `WBDC2` device with the name "WBDC2". It has four inputs named "F1E", "F1H", "F2E" and "F2H" which were introduced in Subsection 2.1.1. They obtain their signals from front end outputs with the same names. They don't have to be the same but it was natural in this case. In general, port names can be the label for the port on the device.

In this case the description stops at the WBDC. However, a phantom `FrontEnd` object (one that can't be controlled or monitored) had to be defined because the `Receiver` expects a signal source. The `FrontEnd` needs a `Telescope` for a signal.

The `Observatory` provides a context for the `Telescope`. There are devices which depend on the observatory, rather than a particular telescope. The `Telescope` object has one output which provides the signal to one or more input ports of the `FrontEnd`. The `Observatory` object has information not specific to a `Telescope`, such as weather data.

```
from MonitorControl import Observatory, Telescope, ClassInstance
from MonitorControl.FrontEnds import FrontEnd
from MonitorControl.FrontEnds.K_band import K_4ch
from MonitorControl.Receivers import Receiver
from MonitorControl.Receivers.WBDC.WBDC2 import WBDC2

import logging
module_logger = logging.getLogger(__name__)

def station_configuration(roach_loglevel=logging.WARNING):
  """
  Configuration for the K-band system on DSS-43 using WBDC2

  Feed 1 (F1) is at 024-0.016, F2 at 024+0.016.  The polarizations are linear,
  E and H.  There are so many receiver outputs that it is simpler to let the
  software generate them.
  """
  observatory = Observatory("Canberra")
  telescope = Telescope(observatory, dss=43)
  front_end = ClassInstance(FrontEnd, K_4ch, "K",
                    inputs = {'KF1': telescope.outputs[telescope.name],
                              'KF2': telescope.outputs[telescope.name]},
                    output_names = [['F1E','F1H'],
                                    ['F2E','F2H']])
  IFswitch = None
  receiver = ClassInstance(Receiver, WBDC2, "WBDC-2",
                           inputs = {'F1E': front_end.outputs["F1E"],
                                     'F1H': front_end.outputs["F1H"],
                                     'F2E': front_end.outputs["F2E"],
                                     'F2H': front_end.outputs["F2H"]})
  clock = None
  backend = None
  return observatory, telescope, front_end, receiver, IFswitch, clock, backend
```

Snippet 4.1: Configuration description used by the program `wbdc2_prompt.py` to test WBDC2 in the lab.

Appendix A

LabJack Configuration

A.1 LabJack 1

This monitors and controls the motherboard.

```
FIO0-FIO3 - AIN: monitor voltages, current, temperatures
FIO4-FIO6 -      not used
FIO7      - DO:  read digital in
EIO0-EIO7 - DO:  select latch by address
CIO0-CIO3 - DO:  set WBDC signals
```

So this is the LabJack Configuration

```
CIOBitDir     1111
EIOBitDir 11111111
FIOBitDir 00000000
FIOAnalog 00001111
```

A.2 LabJack 2 (and 3)

This controls the LJTickDAC® voltage sources. All EIO and CIO ports are assigned to controlling this function and so are digital outputs.

A.3 Configuration Check

An initial checkout might be as shown in Snippet A.1. The normal state of the WBDC1 LabJacks is something like

```
In [2]: from MonitorControl.Electronic.Interfaces.LabJack import *
In [3]: lj = connect_to_U3s()
In [4]: report_IO_config(lj)
```

Snippet A.1: Commands for checking the LabJack I/O configuration.

```
=============== U3 I/O Configurations ================
          U3 local ID          1        2
          ------------- -------- --------
              CIOBitDir 00001111 00000000
               CIOState 00001111 00001111
             DAC1Enable 00000000 00000000
              EIOAnalog 00000000 00000000
               EIOState 01010111 11111111
         EnableCounter0    0.000    0.000
         EnableCounter1    0.000    0.000
                  FAIN0    0.001
                  FAIN1    1.455
                  FAIN2    0.000
                  FAIN3    0.000
              FIOAnalog 00001111 00000000
              FIOBitDir 00000000 00000000
               FIOState 11110000 11111111
  NumberOfTimersEnabled        0        0
          Temperature  295.569  295.507
     TimerCounterConfig       64       64
  TimerCounterPinOffset        4        4
```