RFControl User Manual

Red Oak Canyon LLC

terry@redoakcanyon.com

Overview

The RFControl Package contains two programs, RFControl.exe, an enhanced version of Synthmachine, and RFShell.exe, a command interface that issues commands to RFControl.exe.

RFControl.exe receives external commands through a TCP/IP network connection when it is launched in "server mode". When RFControl is launched with no parameters, it displays the same GUI and behaves identically to SynthMachine.

RFShell reads commands from a text file or from a command line, processes them, and sends them to RFControl. The shell provides loop control, delay, pause, and variable substitution prior to passing commands to RFControl. The "attach" launches RFControl, while the "host" command connects to an exisitng instance of RFControl.

RFShell can be replaced with your own application so the synthesizer can be controlled along with other components in your system.

RFControl is opened in "server mode" by launching it with runtime parameters. The parameters include a key that corresponds to the serial number of an attached TPI RF Signal Generator board, an option to display or hide the GUI, and an optional network port number to use for communication.

Installing RFControl and RFShell

RFControl and RFShell are both installed into the /Program Files (x86)/SynthMachine folder (or on 32 bit machines, /Program Files/SynthMachine) by launching RFControl_Setup.

RFControl uses the same calibration file as SynthMachine. The calibration file is specific to the TPI RF board and is installed in /Program Files (x86)/SynthMachine when SynthMachine_Setup is run, so it is necessary to execute SynthMachine_Setup and then RFControl_setup.

RFControl also uses the same defaults preferences file as SynthMachine, so any changes to the defaults preferences will be picked up by RFControl when it is launched.

RFControl expects the calibration and defaults preferences files to be located in the same folder where the RFControl executable resides. RFShell expects RFControl to be located in the same folder as RFShell.

As with SynthMachine, the FTDI drivers must also be installed, and the TPI RF board must be attached.

Note: In demo/test mode, it is not necessary to load the FTDI drivers or have a TPI RF Synthesizer board attached since the USB connection is disabled.

RFControl Startup, Communication and Usage

RFControl is launched in server mode by launching it with prameters. The TPI board to control is specified with a key that represents the board serial number as a method software copy control.

If RFControl was launched from a command window, launching it in command mode would look like this:

Command> RFcontrol HEXKEY -hide/-show portnumber logfile

Where:

HEXKEY --- is the key corresponding to the TPI RF board serial number

-hide or -show --- hides or shows the GUI in server mode

portnumber --- is the TCP/IP port to use for communication. If portnumber is 0 or not supplied, a port number will be assigned by the system.

logfile -- specifies the name of an optional log file where recevied commands can be logged.

RFControl decodes the HEXKEY and tries to open the corresponding TPI RF board serial number. If it is opened, then RFControl tries to open the board specific calibration file and defaults preferences file.

To use RFControl in test/demo mode use the hexkey d04017edd70894f7. In test mode, the USB channel is disabled.

If a suitable calibration file is not found, a file menu window will open so the calibration file can be located, even if the –show option is not specified. To avoid this dialog, place the calibration file in the same folder as RFControl. If a defaults preferences file is not found in the folder containing RFControl, the internally programmed defaults will be used. The defaults file will be created in the folder containing RFControl when RFControl shuts down (usually /Program Files (x86)/SynthMachine).

Once the TPI RF board is successfully opened, RFControl will reply with a message via STDOUT containing the TCP port number for command communications:

0.0.0.0 0.0.0.0 50192

The first two fields correspond to the IP address and network name of “localhost” The third field is the TCP port number RFControl is listening to for further commands. If the port number was not specified at launch time, it will be dynamically assigned by Windows and could be different for each instantiation of RFControl. If the port number was specified, that port number will be

used if it is available.

If the board cannot be opened, RFControl will respond with:

Can’t attach using HEXKEY where HEXKEY is the value passed when the program was launched.

Programming Examples: Launching RFControl

The application should open RFControl with a pipe to STDOUT so it can capture the response. Note that these are code segments for demonstration of the main concept but don’t code for the possible exceptions and error returns.

In TCL launching RF control and capturing the port number might look like:

```
set appname “|RFControl HEXKEY”   # note the | to open the pipe
set ahandle [open $appname]
gets $ahandle buf
lassign $buf ipname ipaddr portnum
```

In perl:

```
$appname = “RF_Control.exe HEXKEY”
open (AHANDLE, “<$appname”);  # in perl, < redirects STDOUT to us
$buf = <AHANDLE>;
($ipname, $ipaddr, $portnum) = split (“ “,$buf);
```

In c#:

```
Process p = new Process();
// Redirect STDOUT from RFControl to us
p.StartInfo.UseShellExecute = false;
p.StartInfo.RedirectStandardOutput = true;
p.StartInfo.FileName = "C:/Program Files (x86)/SynthMachine/RFControl.exe";
// Launch it with Hexkey it and get the first line of output from STDOUT
p.StartInfo.Arguments = Hexkey;
p.Start();
string output = p.StandardOutput.ReadLine();
// Output will have port number in third parameter or an error message.
char[] delim = { ' ' };
string[] fields = output.Split(delim);
Int32 portnum = Convert.ToInt32(fields[2]);
```

The application would then open the TCP port using the appropriate form of socket command. For example,

In TCL this would be:

```
set handle [socket “localhost” $portnum]
gets $handle buf                    # buf has RF control response
```

In perl this might be:

```
use scrict;
use IO::Socket;
my $handle;
$handle = IO::Socket::INET->new (Proto => “tcp”,
   PeerAddr = “localhost”,
   PeerPort = $portnum);
$buf  = <$handle>;
```

In c#:

```
// the port must be opened, and a stream must also be allocated
client = new TcpClient(“localhost”, port);
     stream = client.GetStream();
     buf = GetMessage();
```

When the connection is established, RFControl will respond via the TCP connection with:

SynthMachine Ready\n

RFControl Commands

Commands are sent to RFControl using the handle (or object) for the open channel. Note that all commands must be terminated with a newline (\n).

RFControl recognizes the following commands:

constant mode:

rfon - turn the RF on

rfoff - turn the RF off

frequency scan modes:

freqscan - start frequency scan

freqonce - start frequency once mode. Issue again to scan once "again"

freqstep - start frequency step mode. Issue again to take next step

power scan modes:

pwrscan - start power scan

pwronce - start power once mode. Issue again to scan once "again"

pwrstep - start power step mode. Issue again to take next step

stop scan frequency or scan power operations:

stop - cancels the current scan mode and turns off the RF if it is on

set up frequency and power

setfreq value - set the frequency to value

setpwr value - set the power to value

set up frequency scan modes

setfreqfrom value - set the scan frequency "from" field to value

setfreqto value - set the scan frequency "to" field to value

setfreqstep value - set the scan frequency step size field to value

setfreqdelay value- set the scan frequency delay field to value

set up power scan modes

setpwrfrom value - set the scan power "from" field to value

setpwrto value - set the scan power "to" field to value

setpwrstep value - set the scan power step size field to value

setpwrdelay value - set the scan power delay field to value

other:

help - display commands. The returned string is a message that uses embeded “,,” as a new line separator. To properly display this string replace “,,” with “\n”. The length of this string may change in the future.

shutdown - terminates RFControl

For each command sent to RFControl, the server responds with a string containing a status, an error, or the help message as below:

The possible responses:

rfon @ {frequency} MHz and power {power}
good - freqscan from {f1} to {f2} by {step} every {delay}
good - freqonce from {f1} to {f2} by {step} every {delay}
good - freqstep from {f1} to {f2} by {step}
good - pwrscan from {p1} to {p2} by {step} every {delay}
good - pwrstep from {p1} to {p2} by {step}
good - stopped scan
good - stopped once/step
error - already scanning – stop first: issued if a scan operation is already in progress
nothing scanning - if a stop is issued when scanning is not active
value* is not a valid frequency
value* is not a valid power setting
power step must be non-zero integer
abs (frequency step) must be > .001
delay must be >= .01 seconds
unknown command – send help
a "help string" that contains the valid RFControl commands.
* value is the value parameter that was passed to RFControl with the command.

Programming Examples: Command processin

In TCL, the interaction between the application and RFControl may look like this, assuming the socket had been opened as above:

```
puts $handle "rfon"   # send the command

        flush $handle
        gets $handle buf    # get the response
```

In perl it may look like this:

```
        print $handle "rfon"; #send the command
        $buf = <$handle>;  # get the response
```

In c#:

```
        // allocate no less than the longest reply message expected from RFControl (364 bytes)
        or multiple reads  will have to be performed to read the full help reply.
                Byte[] data = new Byte[364]
        // append the newline to the command
                data = System.Text.Encoding.ASCII.GetBytes(message + "\n");
         // Send the message to the connected TcpServer via the stream
                stream.Write(data, 0, data.Length);
        // Read full response from stream
```

```
Int32 bytes = stream.Read(data, 0, data.Length);
string buf = System.Text.Encoding.ASCII.GetString(data, 0, bytes);
```

RFShell Usage

RFShell is the scripting front end to RFControl.

RFShell can be started from a command window or from another application. If a file is specified, commands will be read from the file, otherwise, commands will be read interactively.

RFShell provides commands to:

Launch RFControl and attach to a TPI RF Synthesizer board conected to a USB port

Connect to an existing instance of RFControl

Define variables and reference them in subsequent RFControl and Script commands

Increment variables by an arbitrary positive or negative number

Define loops that iterate for a specified number of times

Delay for a specified number of milliseconds

Pause a script running from a file until the user presses the <return> key

Commands for Launching /or or connecting to RFControl

The attach command launches RFControl opens the communication channel between RFControl and RFShell, and attempts to attach to the TPI RF board that corresponds to the provided key.

attach HEXKEY -hide/-show portnumber

where:

HEXKEY is the encrypted form of the TPI board serial number

hide/-show will hide/display the synthMachine front panel

portnumber is the TCP/IP port to use for communication

The host command opens the communication channel between RFShell and an existing instance of RFcontrol on the machine named hostname (or at the ipaddress).

host hostname/ip_address portnumber

where:

hostname/ipaddress is either the name of, or the ip address of, a machine on the network where RFControl is running

portnumber is the port to use with RFControl

RFShell programming commands:

variables:

setvar varname number - set varname to number

incvar varname number - increment varname by number

loops:

loop start name count - mark beginng of a loop labeled "name" that will loop "count" times

loop end name - mark end of the loop labeled "name"

delay and wait:

delay timeval - delay "timeval" ms

wait - wait for the user to hit the RETURN key when executing commands from a file

Issuing RFControl Command

command* number - send an RF command with number as the "value" parameter

command* $varname - send an RF command with varname substitued for the "value" parameter

*command: Available commands are listed in the RFControl Section above

exit:

exit - exit the RFShell

Demonstration Mode

RFControl demonstration mode can be initiated by launching RFControl with Hexkey d04017edd70894f7. In this mode, all USB I/O is disabled so it is not necessary to attach a board or load the FTDI drivers.

Use the -show option after the Hexkey to display the front panel to display the commands issued to RFControl.

It is still necessary to load a calibration file. For demonstration mode, the default caltable.csv file can be used. It should be placed into /Program Files (x86)/SynthMachine.