NASA/TM—2017-219496

Integrated Power, Avionics, and Software (iPAS) Space Telecommunications Radio System (STRS) Radio User's Guide—Advanced Exploration Systems (AES)

Rigoberto Roche and Mary Jo Shalkhauser
Glenn Research Center, Cleveland, Ohio

June 2017

NASA/TM—2017-219496

Integrated Power, Avionics, and Software (iPAS) Space Telecommunications Radio System (STRS) Radio User's Guide—Advanced Exploration Systems (AES)

Rigoberto Roche and Mary Jo Shalkhauser
Glenn Research Center, Cleveland, Ohio

National Aeronautics and
Space Administration

Glenn Research Center
Cleveland, Ohio 44135

June 2017

Contents

Integrated Power, Avionics, and Software (iPAS) Space Telecommunications Radio System (STRS) Radio User's Guide—Advanced Exploration Systems (AES)

Rigoberto Roche and Mary Jo Shalkhauser
National Aeronautics and Space Administration
Glenn Research Center
Cleveland, Ohio 44135

Summary

The Space Telecommunications Radio System (STRS) provides a common, consistent framework for software defined radios (SDRs) to abstract the application software from the radio platform hardware. The STRS standard aims to reduce the cost and risk of using complex, configurable, and reprogrammable radio systems across NASA missions. To promote the use of the STRS architecture for future NASA advanced exploration missions, NASA Glenn Research Center developed an STRS-compliant SDR on a radio platform used by the Advanced Exploration System program at NASA Johnson Space Center in their Integrated Power, Avionics, and Software (iPAS) laboratory. This document provides information to program, operate, and configure the iPAS STRS radio on the Reconfigurable, Intelligently-Adaptive Communication System (RIACS) platform.

1.0 Introduction

The Integrated Power, Avionics, and Software (iPAS) Space and Telecommunications Radio System (STRS) radio is being implemented on the Reconfigurable, Intelligently-Adaptive Communication System (RIACS) platform, currently being used for radio development at NASA Johnson Space Center. The platform consists of a Xilinx® Virtex®-6 ML605 Evaluation Kit, an Analog Devices AD–FMCOMMS1–EBZ radiofrequency (RF) front-end board, and an Axiomtek™ eBOX620–110–FL embedded personal computer (PC) running the Ubuntu® 12.04 LTS operating system (OS). Figure 1 shows the RIACS platform hardware.

1.1 System Overview

The goal of this development is to implement the STRS standard for software defined radios (SDRs) on the RIACS platform at Johnson. At the conclusion of the development, the software and hardware description language (HDL) code will be delivered to Johnson for their use in their iPAS testbed and for development of their own STRS waveforms on the RIACS platform.

1.2 Document Overview

The purpose of this document is to describe how to configure and operate the iPAS STRS radio platform with its delivered test waveform. Throughout this document, commands are always shown in *italics*.

2.0 Design Overview

This section presents a design overview of the iPAS STRS radio components.

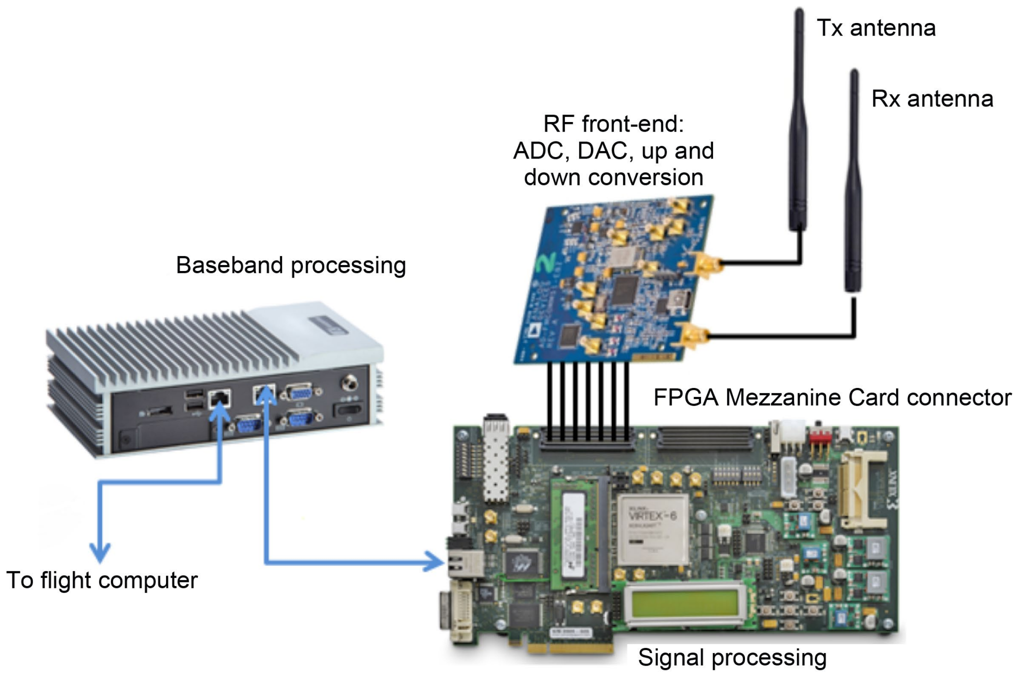


Figure 1.—Reconfigurable, Intelligently-Adaptive Communication System (RIACS) platform. ADC, analog-to-digital converter; DAC, digital-to-analog converter; FPGA, field-programmable gate array; RF, radiofrequency; Rx, receive; Tx, transmit; UDP, User Datagram Protocol.

2.1 General Purpose Module (GPM)

The GPM is the implementation of the STRS command infrastructure on the iPAS radio. It houses the operating environment (OE) and presents a communication conduit for commands and data to and from the signal processing module (SPM). The GPM is where the general purpose processor (GPP) hardware is contained and accessed by the OS running the STRS project files.

The GPP hardware (eBOX620–110–FL) is used under the STRS architecture. This architecture can perform transmit-side (Tx-side) streaming to the SPM, receive-side (Rx-side) streaming from the SPM, command generation, command transmission to the SPM, and command processing of responses from the SPM, all simultaneously. The GPP also controls the Ethernet communication protocol (configured in Transmission Control Protocol (TCP)/Internet Protocol (IP)v4), scheduler and dynamic memory allocation of the hardware, and systemic pointing for STRS command and control interfaces.

This architecture handles the use of specific waveform characteristics that are commanded in the iPAS test waveform. These parameters include a pseudorandom bit sequence (PRBS) generator in the Tx side for parallel streaming to the SPM, an in-phase (I) and quadrature (Q) channel data source for parallel streaming to the SPM, an Rx-side streaming bit error rate tester (BERT), and a plotting tool for a graphical representation of the incoming data. Additional features include commands to display the bit error rate (BER) from SPM BERT and GPM BERT. Debugging tools included in this software architecture allow the user to query status bits and display any issues indicated by such bits. Additional queries are possible to obtain the status of the first in first out (FIFO) buffer during Tx-side streaming and observation of automatic speed adjustments of Tx-side packet streaming.

2.2 Signal Processing Module (SPM)

The purpose of the field-programmable gate array (FPGA) design is the implementation of the SPM functions of the STRS radio architecture in the iPAS RIACS platform. The FPGA design consists of two parts: the FPGA wrapper and the test waveform. The FPGA wrapper implements each platform interface as follows:

(1) Ethernet communication to the embedded processor for commanding and data streaming
(2) Digital-to-analog converter (DAC) and analog-to-digital converter (ADC) interface to the RF front-end board (AD–FMCOMMS1–EBZ)
(3) RF front-end board (AD–FMCOMMS1–EBZ) control and configuration
(4) FPGA clocking

The test waveform does not fully implement all the signal processing functionality for a radio, but it exercises and demonstrates each interface in the FPGA wrapper. A future user of the platform for an STRS radio would use the FPGA wrapper and replace the test waveform with their own radio signal processing functions.

The FPGA design is required to receive and process commands and provide command control and data to the test waveform. It must also receive and transmit streaming data from and to the embedded processor. The test waveform demonstrates each FPGA wrapper interface. To test Tx-side streaming, it can perform BER testing on Tx-side PRBS streaming data. It can also generate PRBS streaming data packets for an Rx-side streaming data source. The test waveform generates sine and cosine waves for the I and Q inputs to the RF transceiver. Captured I and Q outputs of the RF transceiver can be streamed to the embedded processor where it can be plotted to demonstrate proper functionality of the RF board and its interfaces.

2.3 Radiofrequency Module (RFM)

The RF front-end board (AD–FMCOMMS1–EBZ) provides the analog and RF signal processing for the iPAS STRS radio. On the Tx side, the RF front-end board (AD–FMCOMMS1–EBZ) takes complex I and Q inputs (16 bits) into a high-speed DAC to create an analog signal. The DAC output signal is up-converted to the desired RF by a Q modulator. On the Rx side, the received RF signal is demodulated using direct conversion to create I and Q analog signals. The analog signals are converted to digital data using a 14-bit ADC.

2.4 Top-Level Design Description

Figure 2 shows how the STRS standard is implemented on the RIACS platform. The SPM encompasses the FPGA design, which consists of the following:

- The FPGA wrapper that implements all the interfaces to the FPGA and abstracts them from the waveform.
- The waveform that is the FPGA implementation of the radio signal processing functions.

2.5 Concept of Operation

The flight computer (FC) graphical user interface (GUI) simulates the STRS commands that would originate from a typical FC. The GPM is implemented on the embedded PC (eBOX620–110–FL) and includes the STRS OE and waveform application software. The STRS OE communicates with the waveform application through standard STRS application programming interfaces (APIs) to control and configure the waveform.

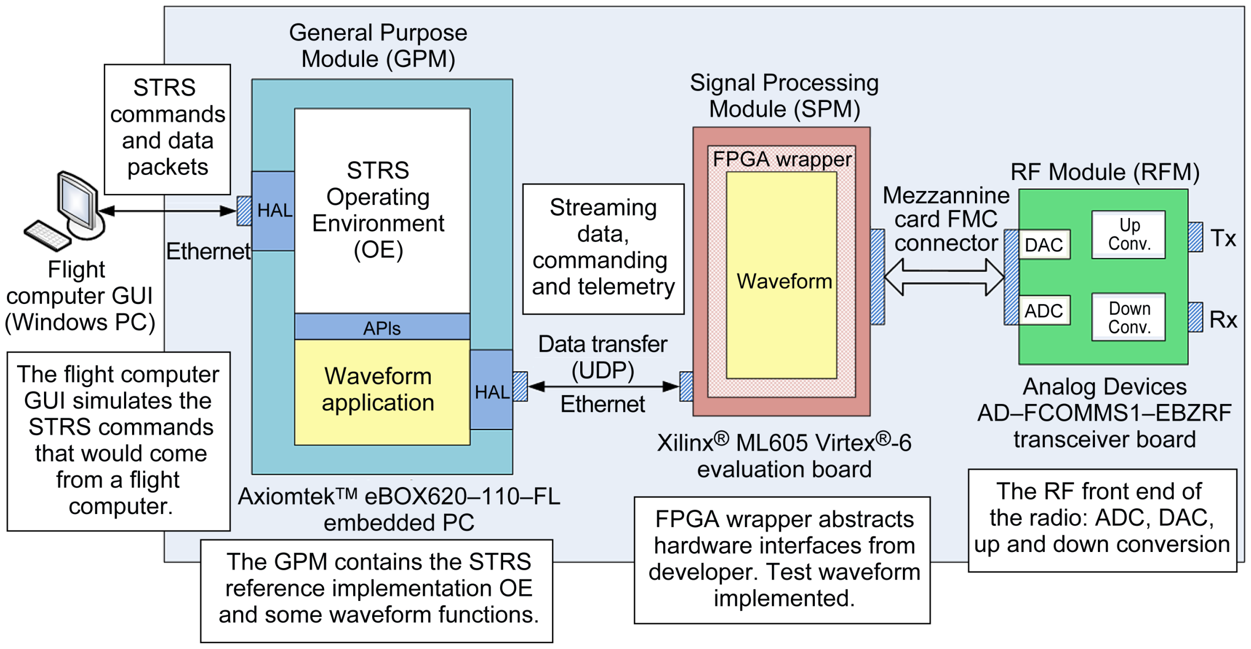

General Purpose Module (GPM)
STRS commands and data packets
Flight computer GUI (Windows PC)
Ethernet
HAL
STRS Operating Environment (OE)
APIs
Waveform application
HAL
The flight computer GUI simulates the STRS commands that would come from a flight computer.
Axiomtek™ eBOX620–110–FL embedded PC
The GPM contains the STRS reference implementation OE and some waveform functions.
Streaming data, commanding and telemetry
Data transfer (UDP)
Ethernet
Signal Processing Module (SPM)
FPGA wrapper
Waveform
Xilinx® ML605 Virtex®-6 evaluation board
FPGA wrapper abstracts hardware interfaces from developer. Test waveform implemented.
Mezzannine card FMC connector
RF Module (RFM)
DAC
ADC
Up Conv.
Down Conv.
Tx
Rx
Analog Devices AD–FCOMMS1–EBZRF transceiver board
The RF front end of the radio: ADC, DAC, up and down conversion

Figure 2.—Space Telecommunications Radio System (STRS) implementation on Reconfigurable, Intelligently-Adaptive Communication System (RIACS) platform. ADC, analog-to-digital converter; APIs, application programming interfaces; DAC, digital-to-analog converter; FPGA, field-programmable gate array; GUI, graphical user interface; HAL, hardware abstraction layer; PC, personal computer; RF, radiofrequency; Rx, receive; Tx, transmit; UDP, User Datagram Protocol.

The SPM is implemented in the Xilinx® ML605 FPGA board. The FPGA consists of two parts: an FPGA wrapper and a test waveform. The FPGA wrapper abstracts the hardware interfaces from the waveform developer. The test waveform utilizes each of the hardware interfaces within the wrapper to demonstrate that the wrapper is correctly implemented. The GPM sends commands over an Ethernet port to the FPGA to control and configure the waveform. The GPM also streams packetized data to the FPGA and receives packetized streaming data from the FPGA over the same Ethernet port.

The RF front-end board (AD–FMCOMMS1–EBZ) contains a DAC, up-converter, down-converter, and an ADC. The FPGA configures the RF front-end board (AD–FMCOMMS1–EBZ) using the Xilinx® Microblaze™ 32-bit Reduced Instruction Set Computer soft processor and sends I and Q data to the DAC. The FPGA also receives down-converted and sampled I and Q data from the RF front-end board (AD–FMCOMMS1–EBZ).

The test waveform can demonstrate STRS commands for configuration and control of the test waveform, Tx-side streaming data operation, RF front-end board (AD–FMCOMMS1–EBZ) configuration, Rx-side streaming data, and STRS telemetry querying.

3.0 General Purpose Module (GPM)

This section describes the GPM.

3.1 Software Control Description

The GPM houses the STRS reference implementation. This implementation runs the STRS OE. The STRS OE allows for applications to reference the STRS APIs, which are used to standardize the communication protocols on NASA's space SDRs. The software implementation presented in this design is an adaptation of a test implementation written for VxWorks and ported onto a Linux system.

There are two methods for controlling the iPAS STRS radio. Each method is representative of external control from an FC, where the GPM acts only as a passthrough interface that is standardized by the STRS APIs, sending commands, getting responses, and controlling data transfer between the FC and the SPM.

The first method of control is an external computer connected via Ethernet. This computer runs a flight computer simulator (FCS) that can send commands, receive and interpret command responses, and send and receive data. This method is described in Section 4.0. Additionally, an application for simulating data transfer from an FC called IPASDataIO was written. This application also resides in the external control computer and it is described in Section 4.0. A clone of the IPASDataIO application is also present in the STRS_Architecture_RI directory of the GPM.

The second method of control is internal to the GPM, where a separate software process is implemented to send commands to the process running the STRS reference implementation commanded via Ethernet. The TCP/IP configuration allows for a software socket to be created and any other process can communicate to that interface via a static IP address. This can be used to control the radio using both internal processes in the GPM and an external FCS connected via Ethernet. The GPM is running the Ubuntu Linux 12.04 LTS OS. This OS allows for the communication interfaces in the different parts of the radio to use the standard protocols implemented in the UNIX environment out of the box.

Both control methods described in this document implement the STRS commands described in Table I. The anatomy of these commands is described in the STRS iPAS Hardware Interface Description document (Ref. 1). The command header specified in Table I is only relevant to control the sample waveform developed for this implementation, “WFIPAS2.”

The commands for this radio implementation have the following structure:

Source; Destination; Property; Property Name; Property Value

In this implementation the *Source = FC (Flight Computer)* and *Destination = WFIPAS2 (Sample Waveform).* This can be changed when other waveforms are developed to run on the platform. However, WFIPAS2 requires two commands to be sent so that the SDR can be configured and controlled through the STRS. These commands are:

(1) FC;FC;STRS_InstantiateApp;%STRS_BASE%/WFIPAS2/WFIPAS2.cfg
(2) FC;WFIPAS2;STRS_Initialize

In order to facilitate the automation of standard control commands for WFIPAS2, several bash script files were created with the “.init” extension. These files will be discussed in more detail in Section 3.4. For now, it is only important to know that these files can be run and modified to automatically configure the SDR for a particular task. This way the user is not required to enter command after command and explicitly write out each of the properties. The user may simply edit one of these “.init” files and either copy and paste commands onto the terminal or run the scripts as a whole to achieve a specific configuration.

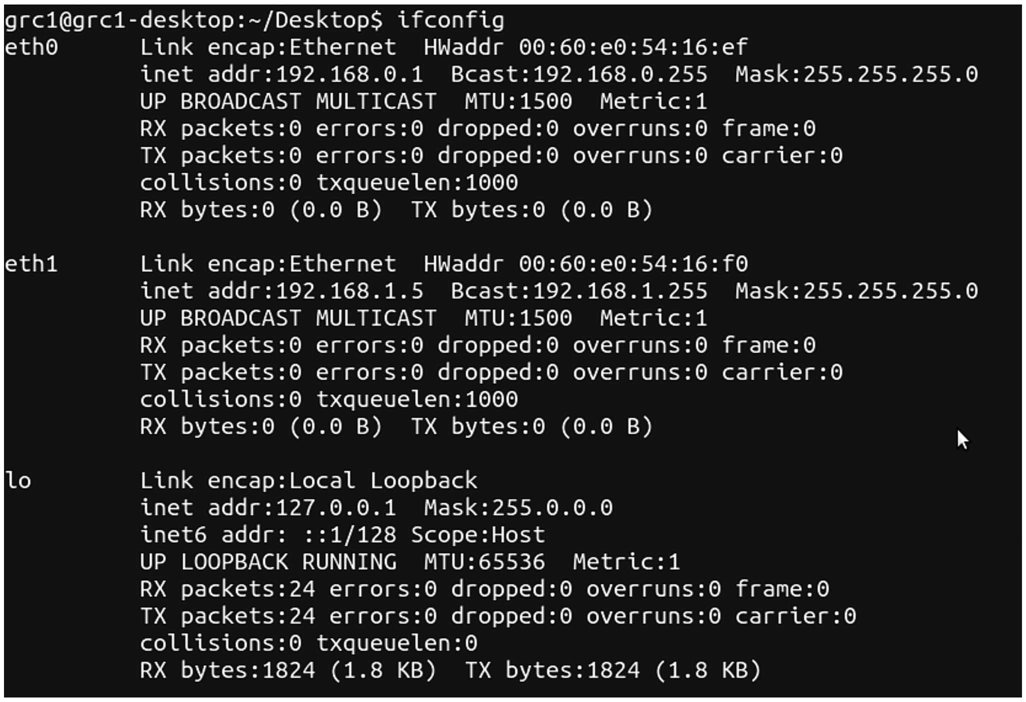
Other commands are implemented in WFIPAS2. However, the relevant commands for control and interface testing of the SDR are listed in Table I. These are the only commands necessary to exercise all the interfaces and demonstrate the capabilities of WFIPAS2.

One important aspect to note is that some commands can only be executed while the waveform is not running. The SOURCETX and SOURCERX property values can only be set while the waveform is stopped. All the other commands should be used after the waveform is running (after executing STRS_Start).

TABLE I.—SPACE TELECOMMUNICATIONS RADIO SYSTEM (STRS) INTEGRATED POWER, AVIONICS, AND SOFTWARE (IPAS) GENERAL PURPOSE MODULE (GPM) COMMAND LIST AND DESCRIPTION

Command source and destination	Command property	Property name	Property value	Description
FC[a]; FC;	STRS_ InstantiateApp	%STRS_BASE% /WFIPAS2 /WFIPAS2.cfg	NA	Instantiate base object for waveform configuration
FC; WFIPAS2[b];	STRS_Initialize;	NA	NA	Initialize environment control interface
FC; WFIPAS2;	STRS_Start;	NA	NA	Start waveform
FC; WFIPAS2;	STRS_Stop;	NA	NA	Stop waveform
FC; WFIPAS2;	STRS_Configure;	SOURCETX	STREAMING_PRBS	Set data source to external PRBS[c] generator
			STREAMING_SINE	Set data source to external sine wave generator
			PRBS	Set data source to internal (from SPM[d]) PRBS generator
			SINEWAVE	Set data source to internal (from SPM) sine wave generator
		SOURCERX	NORMAL	Set data sink to forward data to RF[e] module
			LOOPBACK	Set data sink to return data to SPM bypassing the RFM[f]
		STREAMTX	ENABLE	Enable Tx-side streaming
			DISABLE	Disable Tx-side streaming
		STREAMRX	ENABLE	Enable Rx-side streaming
			DISABLE	Disable Rx-side streaming
		PRBS	ENABLE	Enable internal (from SPM) PRBS generator
			DISABLE	Disable internal (from SPM) PRBS generator
		BERT[g]	ENABLE	Enable internal (from SPM) BER[h] tester
			DISABLE	Disable internal (from SPM) BER tester
		INSERTERROR	1, 2, any integer	Insert errors in internal (from SPM) PRBS generator
FC; WFIPAS2;	STRS_Query;	BERDATA	NA	Queries internal (from SPM) BER tester

[a]Flight computer.
[b]iPAS sample waveform.
[c]Psuedorandom bit sequence.
[d]Signal processing module.
[e]Radiofrequency.
[f]Radiofrequency module.
[g]Bit error rate tester.
[h]Bit error rate.



```
grc1@grc1-desktop:~/Desktop$ ifconfig
eth0      Link encap:Ethernet  HWaddr 00:60:e0:54:16:ef
          inet addr:192.168.0.1  Bcast:192.168.0.255  Mask:255.255.255.0
          UP BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

eth1      Link encap:Ethernet  HWaddr 00:60:e0:54:16:f0
          inet addr:192.168.1.5  Bcast:192.168.1.255  Mask:255.255.255.0
          UP BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:24 errors:0 dropped:0 overruns:0 frame:0
          TX packets:24 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:1824 (1.8 KB)  TX bytes:1824 (1.8 KB)
```

Figure 3.—Typical output from *iconfig* in the general purpose module (GPM).

3.2 Verifying Correct Interface Configuration

The GPM's Ethernet interfaces should automatically configure at login. However, it is good practice to make sure the interfaces are configured correctly before starting operations. To verify the configuration, perform the following actions:

(1) Log in to the jsc1 user.
(2) Open a terminal window.
(3) Type the command *ifconfig*.
(4) If the output from *ifconfig* is not the same as illustrated in Figure 3, restart the GPM.

3.3 Operating the General Purpose Module (GPM) To Control the Software Defined Radio (SDR)

The STRS reference implementation source code is housed in the GPM in the directory path: /home/jsc1/code/strs/STRS_Architecture_RI. In this directory, an executable shell script has been created to run the Ethernet control protocol for the STRS OE to run WFIPAS2. This shell script is called runGUI.sh.

The following procedure describes how to initialize the control interface in the GPM.

(1) Connect the power converter to the embedded PC (eBOX620–110–FL) and to the Xilinx® ML605 FPGA board, respectively. Make sure that the power supplies are not live (not connected to the grid socket).
(2) Connect the included Ethernet cable from the slot on the embedded PC (eBOX620–110–FL), labeled "FPGA," to the Ethernet connector on the Xilinx® ML605 FPGA board.

(3) Connect a Universal Serial Bus (USB) mouse (not included) and USB keyboard (not included) to the embedded PC (eBOX620–110–FL).
(4) Connect a Video Graphics Array (VGA) input display (not included) to the embedded PC (eBOX620–110–FL).
(5) If using an external control computer, connect an Ethernet cable (not included) from the remaining open slot of the embedded PC (eBOX620–110–FL) to the Ethernet connector in the control computer.
(6) Connect the power converters to the grid socket.
(7) Turn on the Xilinx® ML605 FPGA board by operating power On/Off slide switch 2 (SW2) to the On position. The DS25 light-emitting diode (LED) will illuminate green to indicate that the board is powered on.
(8) Turn on the embedded PC (eBOX620–110–FL) by depressing the power switch in the back and releasing it after 3 s.
(9) Once steps (1) to (8) are complete, log in to Linux. The password to log in to the Linux desktop is “strs” (without quotes).
(10) Open a terminal window after login.
(11) Type the following commands into the terminal:

```
cd /home/jsc1/code/strs/STRS_Architecture_RI
./runGUI.sh
```

Once the control script executes, the GPM is ready to receive commands.

In order to send commands internally, use the *nc* command. The *nc* (or netcat) utility is used for controlling actions involving TCP or User Datagram Protocol (UDP). It can open TCP connections, send UDP packets, listen on arbitrary TCP and UDP ports, perform port scanning, and process both IPv4 and IPv6. In the case of the STRS iPAS Radio, the IPv4 standard is used.

With the runGUI.sh script running, static IP address 192.168.1.5 is open with a socket created in the GPM. This socket communicates through port 1120 and listens for inputs from external, connected devices (via Ethernet) or any other process that communicates to that static IP/port combination. For internal control, use the *echo* command and pipe |, the output of *echo* to *nc*. Using Table I, a command that has the following structure can be constructed to be executed under runGUI.sh:

```
echo "FC;WFIPAS2;STRS_Start" | nc 192.168.1.5 1120
```

To send internal commands, follow these steps:

(1) Open a second terminal window (different from the one running runGUI.sh).
(2) Type a command in the way specified above and press Enter.
(3) Look at the runGUI.sh terminal window for output related to the command.

3.4 Examples

Several examples were written to facilitate the use of internal control in the GPM. These examples are files with the extension “.init” and they are located in the following directory path:

```
/home/jsc1/code/strs/STRS_Architecture_RI/WFIPAS2
```

This section focuses on understanding what is happening in each of the lines of one of the examples provided so that the user can have a clear picture of the command structures and procedures used to control the SDR. The example files located in the previously mentioned path are

- WFIPAS.init-normal.sh
- WFIPAS.init-tx.sh
- WFIPAS.init.txt
- WFIPAS.init+loopback.sh
- WFIPAS.init+rxprbs.sh
- WFIPAS.init+test.sh
- WFIPAS.init+txprbs.sh
- WFIPAS.stop.sh

This section explores WFIPAS.init+loopback.sh. The contents of this file are shown below:

```
0   #!/bin/bash
1
2   NC="nc 192.168.1.5 1120"
3   set -x
4
5   echo "FC;FC;STRS_InstantiateApp;%STRS_BASE%/WFIPAS2/WFIPAS2.cfg" | $NC
6   sleep 1
7   echo "FC;WFIPAS2;STRS_Initialize" | $NC
8   sleep 1
9   echo "FC;WFIPAS2;STRS_Configure;SOURCETX;STREAMING_PRBS;SOURCERX;LOOPBACK" | $NC
10  sleep 1
11  echo "FC;WFIPAS2;STRS_Start" | $NC
12  sleep 1
13  echo "FC;WFIPAS2;STRS_Configure;STREAMTX;ENABLE;STREAMRX;ENABLE" | $NC
```

This file configures the SDR to receive PRBS23 data from an external source, pipe the data through the GPM, send the data over to the SPM, and then loopback the data in the SPM back to the GPM, bypassing the RFM. Then the GPM is configured to forward that same data out to an external sink. The external source/sink being used is IPASDataIO (see Sec. 3.1).

Line 0 is a convention, so the *nix shell knows what type of interpreter is necessary in order to run this script. This is not relevant to SDR control; it is simply a convention. Line 2 sets up a variable with the static IP address and port to communicate with the process initiated by runGUI.sh. Line 3 handles environment variable setting in the instance of the shell script. This is not relevant to SDR control; it is only needed for the script to work well if run simultaneously. Line 5 and Line 7 are the initial commands needed to instantiate the base object representative of the waveform and then to initialize the control environment.

The SDR configuration commends are in Lines 9, 11, and 13. Line 9 is a combination of two commands: *FC;WFIPAS2;STRS_Configure;SOURCETX; STREAMING_PRBS* and *FC;WFIPAS2;STRS_Configure;SOURCERX;LOOPBACK*. Since properties with different names and value pairs are being configured under the same parameter control (STRS_Configure), the command can be combined into one STRS_Configure parameter control with the two name and value pairs and it will function the same as sending two independent commands.

The waveform is started using the STRS_Start command. Now, bidirectional streaming can be enabled. Similar to the commands in line 9, the commands in line 13 are combined into one parameter control with two sets of name and value pairs.

The user is encouraged to look through these example files to gain a thorough understanding of the internal operation of the SDR and how these files can be used to configure and command the platform with ease.

3.5 IPASDataIO Internal to the General Purpose Module (GPM)

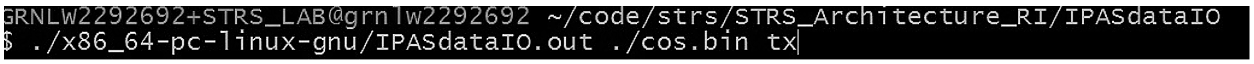
To exercise the external data path to the STRS iPAS Radio, an application called IPASDataIO was written. This is a very simple application. Its main function is to read two files, one PRBS24.bin file, containing a PRBS of 10^{23}–1 unique bits repeated eight times, and a cos.bin file containing a sampled sine wave.

This application takes two inputs: input one is the file to be read and input two is the mode of operation. As previously stated, the files to be read are either PRBS23.bin or cos.bin and the modes of operation are tx (transmit only), rx (receive only), and txrx (transmit and receive). Additionally, this application provides a BERT when receiving PRBS23 data and there are three user commands available while the application is running. These commands are *e* to enter an error into the PRBS23 sequence, *z* to clear the status display of the BERT, and *s* to save a file of sample data containing 2 MB of the Rx stream.

In order to run IPASDataIO follow these steps:

(1) Once logged in to the GPM and running runGUI.sh, configure the SDR to use external data sources using the example files in Section 3.4.
(2) Open a second terminal window (other than the one running runGUI.sh) and navigate to the IPASDataIO directory:
    cd /home/jsc1/code/strs/STRS_Architecture_RI/IPASDataIO
(3) Now the IPASDataIO application can be run in any of these modes:
    (a) Command: *./x86_64-pc-linux-gnu/IPASDataIO.out ./cos.bin tx* is used to send sine wave data to the GPM, but also to receive sine wave data. An example of executing this command is shown in Figure 4.
    (b) Command: *./x86_64-pc-linux-gnu/IPASDataIO.out ./PRBS23.bin rx* is used to receive PRBS23 data to the GPM and pass it through a BERT.
    (c) Command: ./x86_64-pc-linux-gnu/IPASDataIO.out ./PRBS23.bin tx is used to send PRBS23 data to the GPM.
    (d) Command: *./x86_64-pc-linux-gnu/IPASDataIO.out ./PRBS23.bin txrx* is used to send PRBS23 data to the GPM, receive PRBS23 data from the GPM, and pass that data through a BERT.

Note: The IPASDataIO is an independent process to the SDR control interface. It sends data to a specific IP address and port combination; whether that results in sending data to another process running in the same OS or to a process running on a different machine has no effect on the actions of IPASDataIO.



```
GRNLW2292692+STRS_LAB@grnlw2292692 ~/code/strs/STRS_Architecture_RI/IPASdataIO
$ ./x86_64-pc-linux-gnu/IPASdataIO.out ./cos.bin tx
```

Figure 4.—Command execution.

(4) Now that IPASDataIO is running, the terminal window where it is running should be populated with the statistics of the transmitting and receiving of bits, packets, and errors. The user can now enter one of three key strokes to initiate the following actions:
   (a) Press *e* and then Enter to insert an error in the PRBS23 sequence being transmitted.
   (b) Press *z* and then Enter to reset the counters for the statistics being displayed periodically by IPASDataIO.
   (c) Press *s* and then Enter to save a 2 MB sample for the incoming data form an external source to IPASDataIO. The file will be saved under the IPASDataIO directory and it will be called saved_data.bin.

   Note: When IPASDataIO is used in txrx mode there are some initial errors displayed in the BERT statistics, this is due to the asynchronous nature of the external connection to the GPM. Perform step (4)(b) (above) to clear these errors. There should not be any other errors after this initial burst (usually less than 100 errors).

(5) To terminate the application, press the "Ctrl + C" key combination.

4.0 External Interfaces

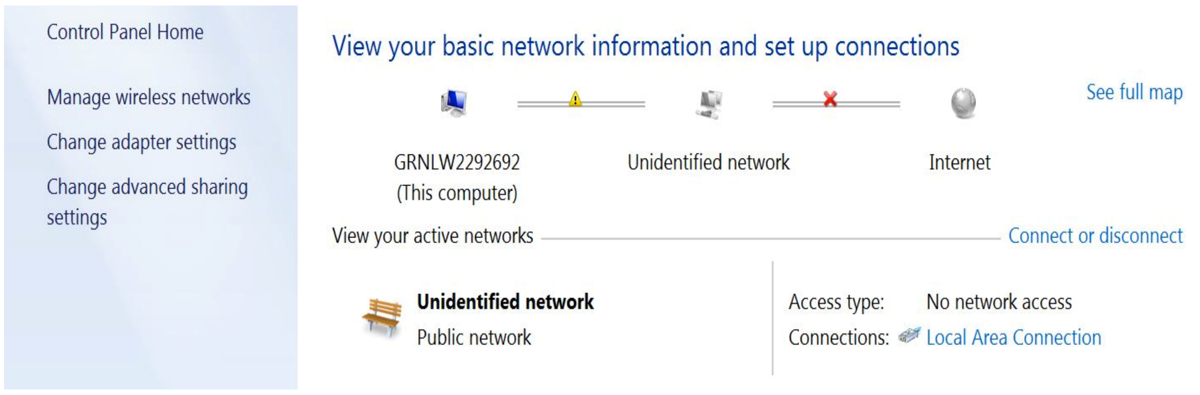
This section describes the system's external interfaces.

4.1 Command and Control Graphical User Interface (GUI)

A command and control GUI can be used as an FCS to send commands to the SDR and receive command responses from the SDR via 1 Gbps Ethernet. The purpose of this interface is to demonstrate the capability for an external physical device (another computer, an FC, or another SDR) to control the STRS iPAS SDR.

This GUI operates on a Windows 7 64-bit OS PC. The user may choose to install the interface in a similar configuration. However, make sure to follow these steps to configure static IP communication via IPv4 on the OS.

(1) To configure this interface, navigate to Control Panel→Network Settings.
(2) If the Ethernet cable is not connected from the control PC to the GPM, connect it now. Once this is done, a screen similar to Figure 5 will appear.



View your basic network information and set up connections

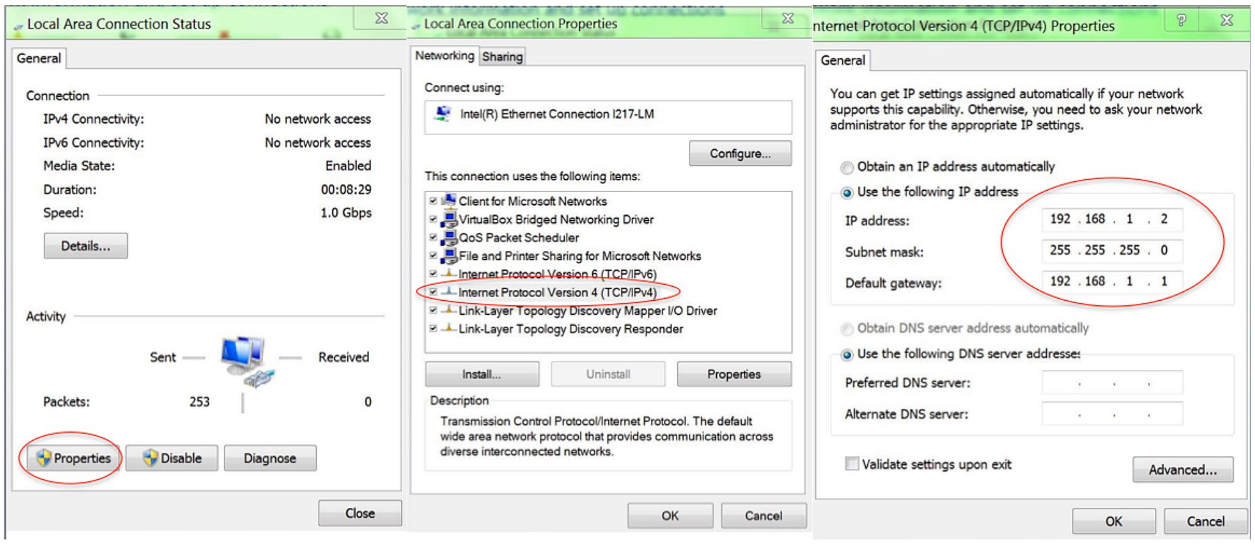
GRNLW2292692 (This computer) — Unidentified network — Internet

View your active networks

Unidentified network
Public network

Access type: No network access
Connections: Local Area Connection

Figure 5.—Ethernet configuration for the flight computer simulator (FCS) for the Integrated Power, Avionics, and Software (iPAS) radio on the Windows 7 personal computer (PC).


Local Area Connection Status

General

Connection

IPv4 Connectivity:	No network access
IPv6 Connectivity:	No network access
Media State:	Enabled
Duration:	00:08:29
Speed:	1.0 Gbps

Activity

	Sent	Received
Packets:	253	0

Local Area Connection Properties

Networking | Sharing

Connect using:

Intel(R) Ethernet Connection I217-LM

This connection uses the following items:

- ☑ Client for Microsoft Networks
- ☑ VirtualBox Bridged Networking Driver
- ☑ QoS Packet Scheduler
- ☑ File and Printer Sharing for Microsoft Networks
- ☑ Internet Protocol Version 6 (TCP/IPv6)
- ☑ Internet Protocol Version 4 (TCP/IPv4)
- ☑ Link-Layer Topology Discovery Mapper I/O Driver
- ☑ Link-Layer Topology Discovery Responder

Description

Transmission Control Protocol/Internet Protocol. The default wide area network protocol that provides communication across diverse interconnected networks.

nternet Protocol Version 4 (TCP/IPv4) Properties

General

You can get IP settings assigned automatically if your network supports this capability. Otherwise, you need to ask your network administrator for the appropriate IP settings.

Obtain an IP address automatically [✓ Use the following IP address]

IP address:	192 . 168 . 1 . 2
Subnet mask:	255 . 255 . 255 . 0
Default gateway:	192 . 168 . 1 . 1

Obtain DNS server address automatically [✓ Use the following DNS server addresses]

Preferred DNS server:	. . .
Alternate DNS server:	. . .

☐ Validate settings upon exit

Figure 6.—Internet Protocol (IP) configuration for the flight computer simulator (FCS) for the Integrated Power, Avionics, and Software (iPAS) radio on the Windows 7 personal computer (PC). DNS, Domain Name System; I/O, input/output.

To configure the local area network (LAN) connection, follow these steps (Figure 6):

(1) Double-click Local Area Connection. A window will appear with connection settings.
(2) Click Properties and then double-click Internet Protocol Version 4.
(3) Select the radio button “Use the following IP address.”
(4) Enter 192.168.1.2 for the IP address and 192.169.1.1 for the Default Gateway. These settings are illustrated in Figure 6.

Once the interface has been configured, the user may choose to copy and paste the STRS_IPAS_Radio_External_Control.iso onto a local directory in their host system and then navigate to the STRS_FCS folder. To run the application and control the STRS iPAS SDR, follow these steps:

(1) Navigate to the local directory where the application is saved.
(2) Navigate to STRS_FCS→STRS_FCS→bin→Debug and run STRS_FCS.exe. A new window opens (Figure 7).

The GUI needs to be configured through this window to communicate with the Ethernet interface that was configured at the beginning of this section.

(3) Click Settings and then click Target Setup (Figure 8). A new window will open.
(4) Enter the following information (Figure 8):
    (a) Name: User’s choice
    (b) IP address: 192.168.1.5
    (c) Tx Port: 1120
    (d) Rx Port: 1130
(5) Click Save New and then click OK.
(6) Click Connect in the top left quadrant of the window (Figure 9). Once connected, the Disconnect button will become active (not shown in Figure 9).

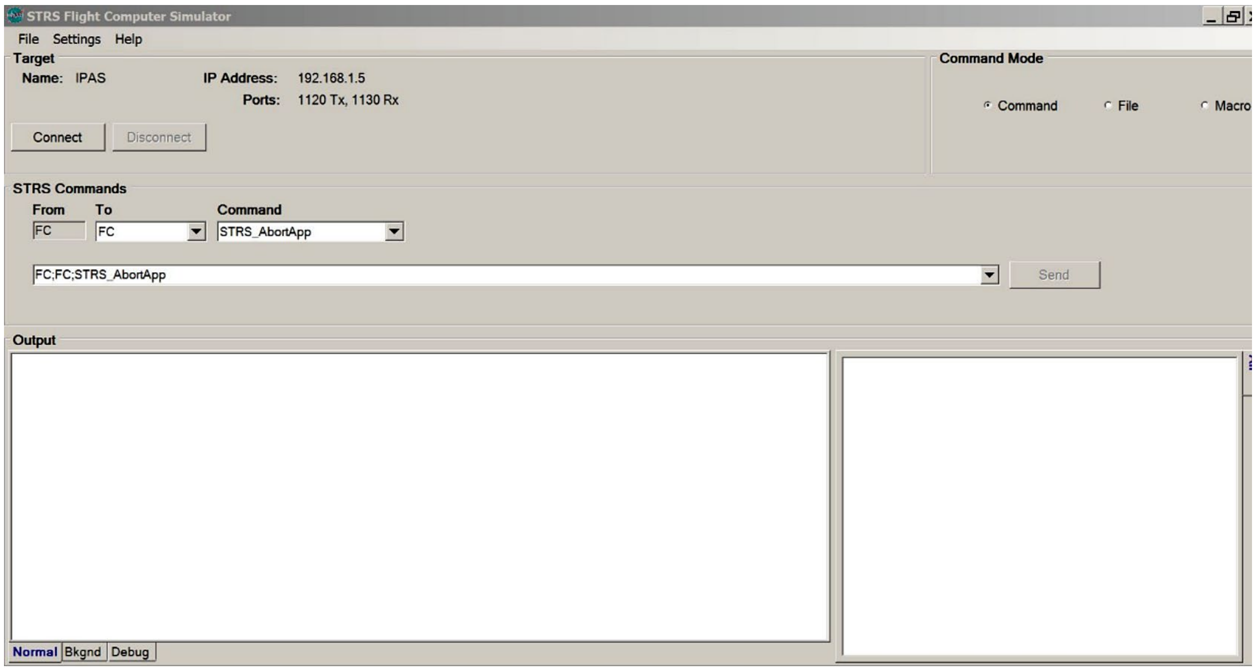


Figure 7.—Space Telecommunications Radio System (STRS) flight computer simulator (FCS). FC, flight computer; IP, Internet Protocol.

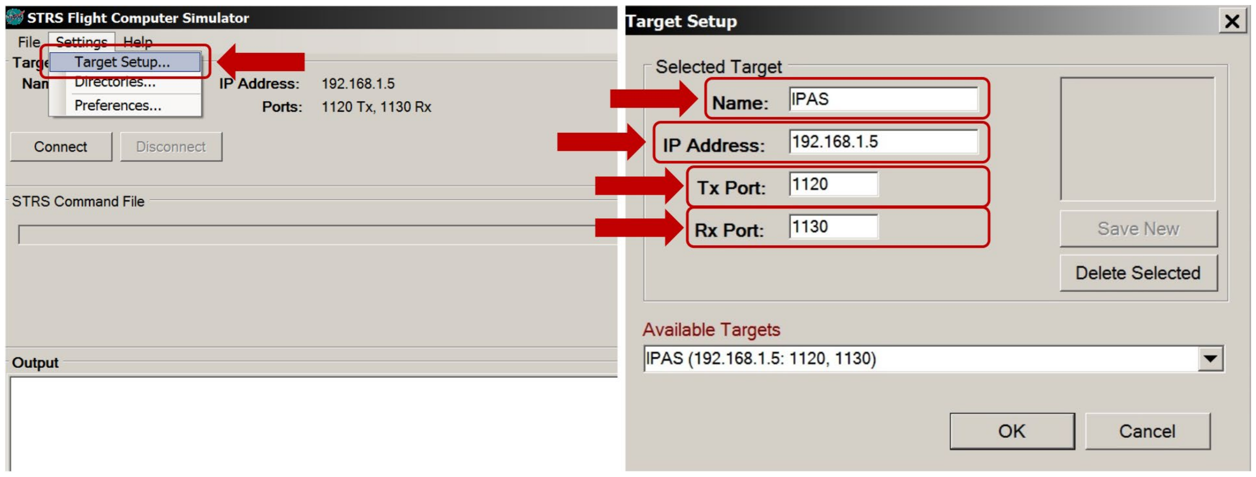


Figure 8.—Space Telecommunications Radio System (STRS) flight computer simulator (FCS) configuration. IP, Internet Protocol; Rx, receive; Tx, transmit.

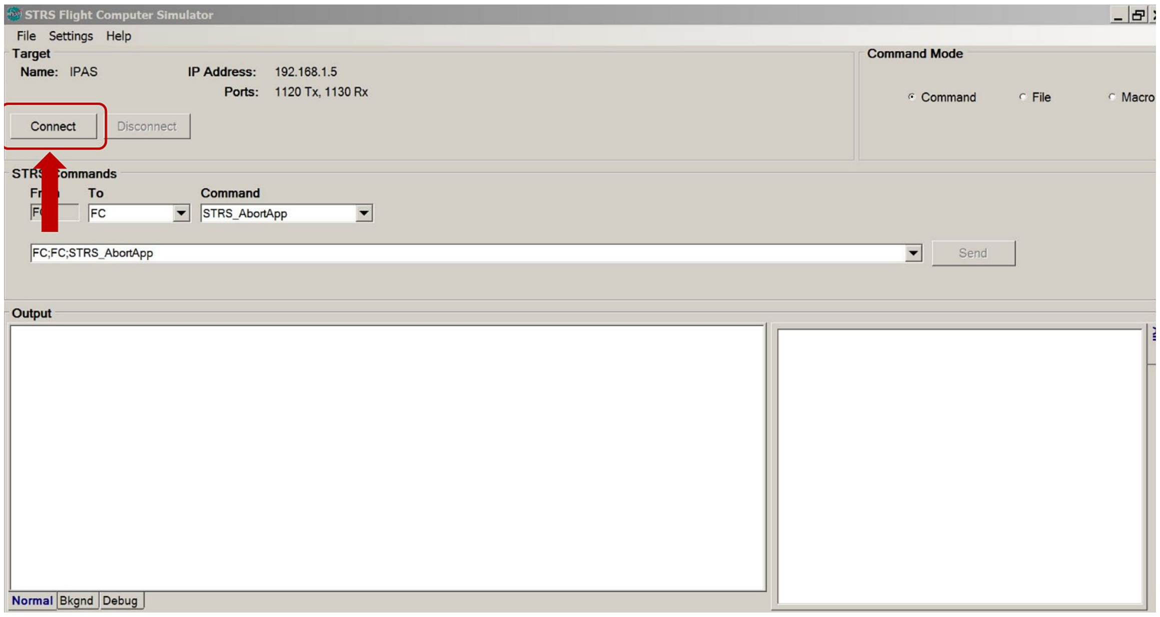


Figure 9.—Space Telecommunications Radio System (STRS) Flight Computer Simulator Connect.

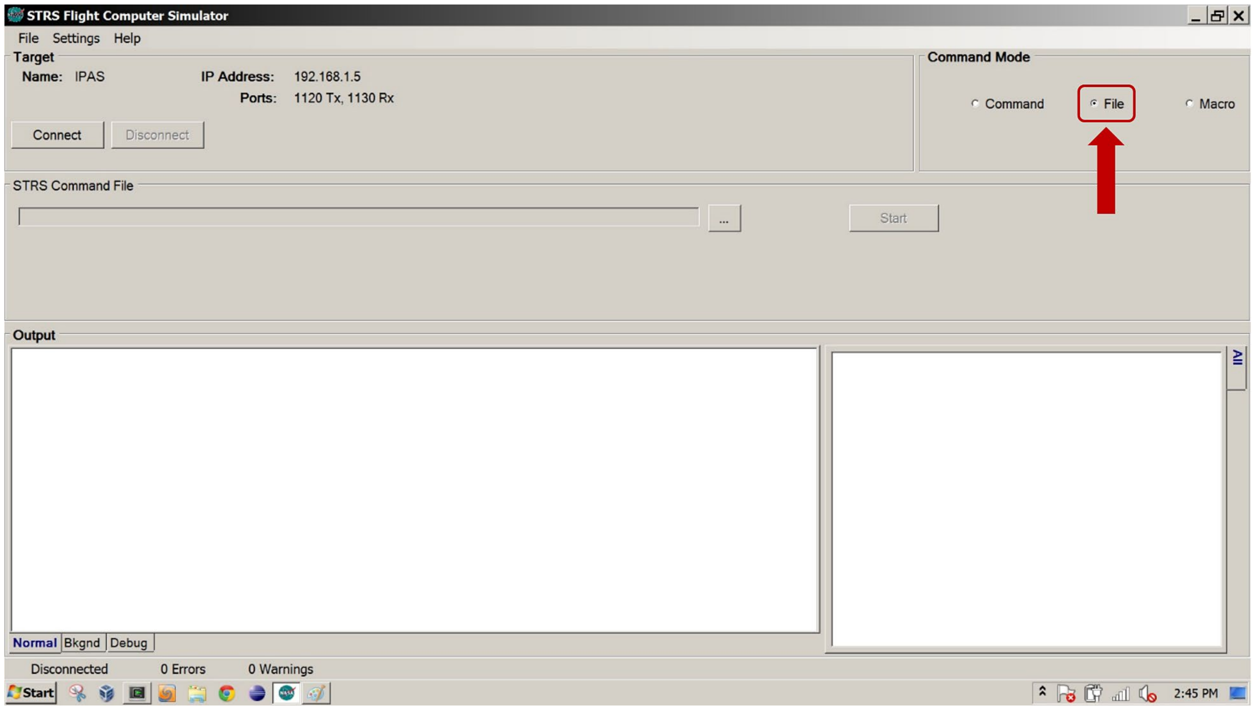


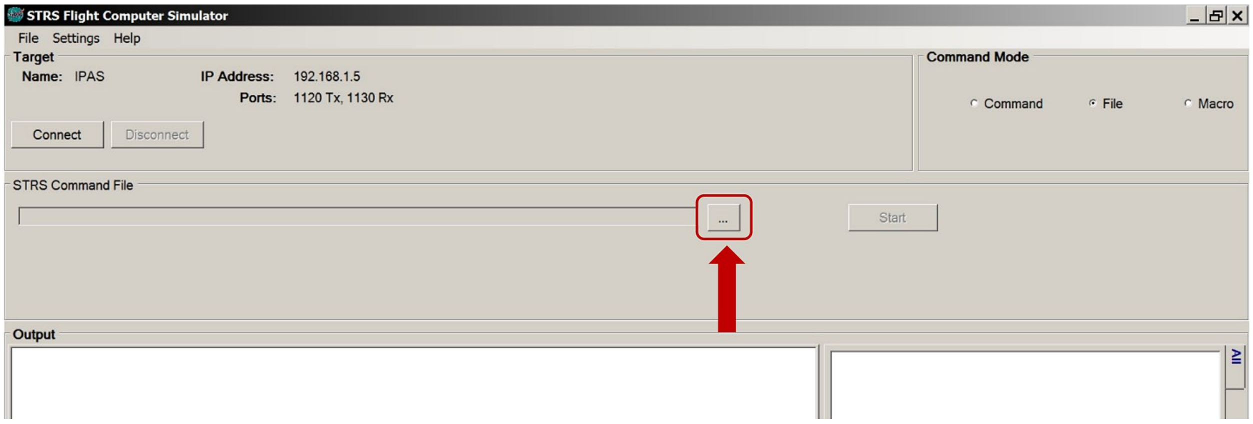
Figure 10.—File Command Mode selection. IP, Internet Protocol; Rx, receive; Tx, transmit.

Now the initialization file must be configured.

(7) Select the File radio button in the top right quadrant of the Command Mode panel. This will change the available options in the window (Figure 10).

(8) Now click the … (ellipsis) button next to the STRS Command File field (Figure 11). An instance of Windows Explorer appears (Figure 12).

(9) Select the appropriate ".init" file. The file is located in STRS_FCS→STRS_FCS→bin→Debug. This configuration uses WFIPAS_JSC.init.

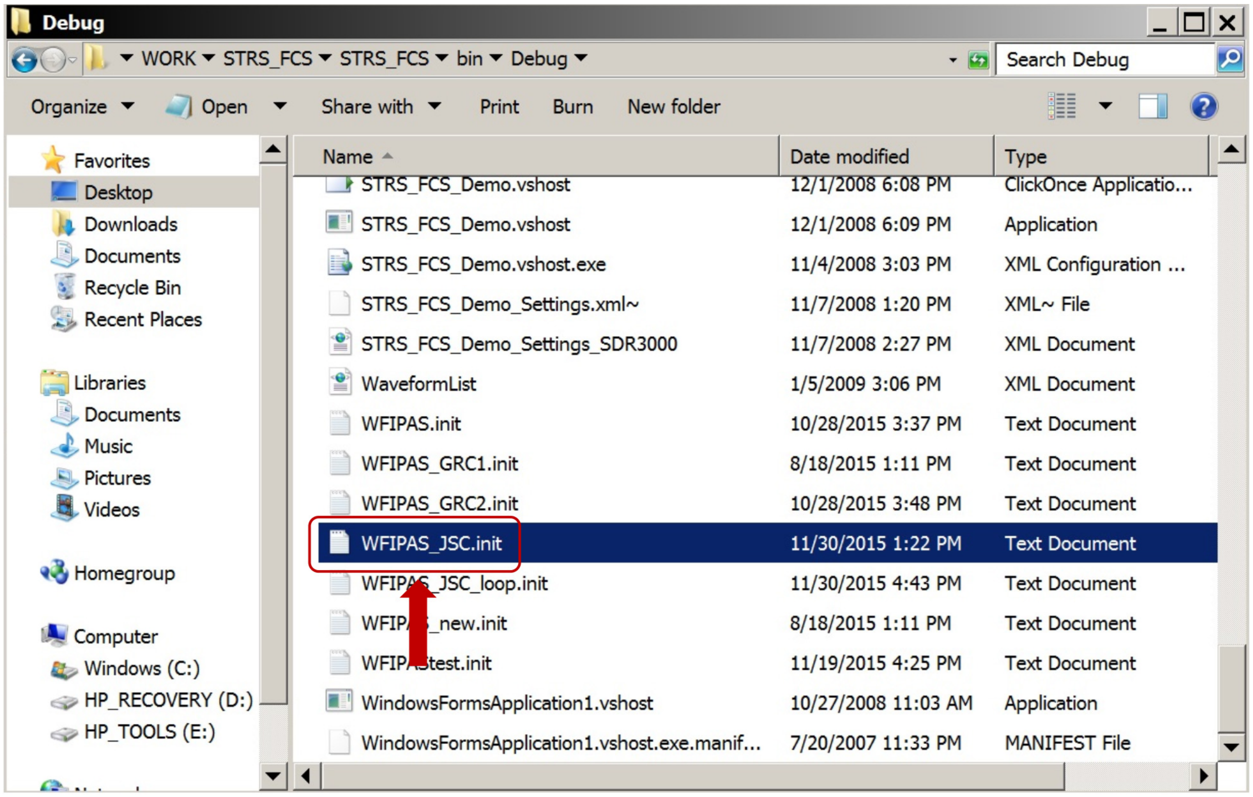


Target
Name: IPAS
IP Address: 192.168.1.5
Ports: 1120 Tx, 1130 Rx

Command Mode: Command [✓ File] Macro

STRS Command File

Figure 11.—… (ellipsis) button selection. IP, Internet Protocol; Rx, receive; Tx, transmit.



Name	Date modified	Type
STRS_FCS_Demo.vshost	12/1/2008 6:08 PM	ClickOnce Applicatio...
STRS_FCS_Demo.vshost	12/1/2008 6:09 PM	Application
STRS_FCS_Demo.vshost.exe	11/4/2008 3:03 PM	XML Configuration ...
STRS_FCS_Demo_Settings.xml~	11/7/2008 1:20 PM	XML~ File
STRS_FCS_Demo_Settings_SDR3000	11/7/2008 2:27 PM	XML Document
WaveformList	1/5/2009 3:06 PM	XML Document
WFIPAS.init	10/28/2015 3:37 PM	Text Document
WFIPAS_GRC1.init	8/18/2015 1:11 PM	Text Document
WFIPAS_GRC2.init	10/28/2015 3:48 PM	Text Document
WFIPAS_JSC.init	11/30/2015 1:22 PM	Text Document
WFIPAS_JSC_loop.init	11/30/2015 4:43 PM	Text Document
WFIPAS_new.init	8/18/2015 1:11 PM	Text Document
WFIPAStest.init	11/19/2015 4:25 PM	Text Document
WindowsFormsApplication1.vshost	10/27/2008 11:03 AM	Application
WindowsFormsApplication1.vshost.exe.manif...	7/20/2007 11:33 PM	MANIFEST File

Figure 12.—Windows Explorer instance.

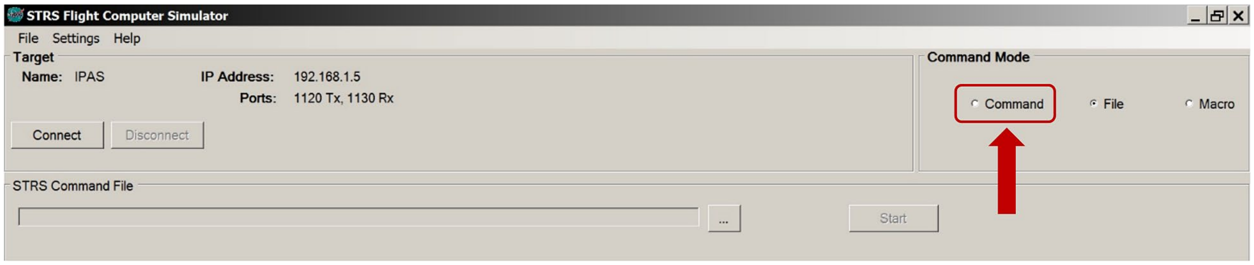
The file path populates to the STRS Command File field.

(10) Click Start. This will cause the output window to be populated and the status window (right bottom quadrant) to display the status of the waveform. In the normal window, green colors represent successful command implementation.

(11) After the initialization step completes, select the Command radio button (Figure 13). The window switches back to display Command functions.

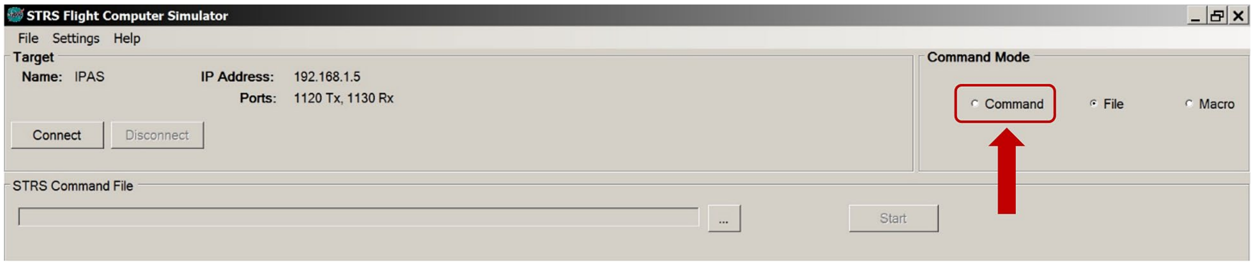
(12) At the command window, perform the following:

- (a) Select WFIPAS2 from the To pull-down menu.
- (b) Select a command from the Command pull-down menu. This example uses STRS_AbortApp (Figure 14).



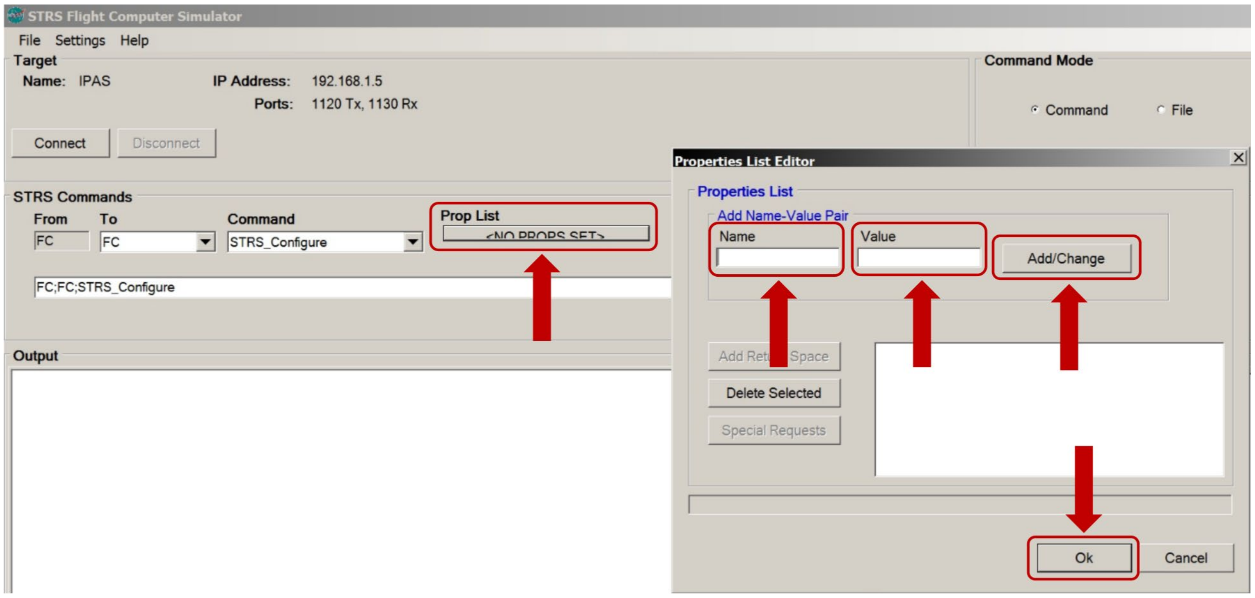
Target
Name: IPAS
IP Address: 192.168.1.5
Ports: 1120 Tx, 1130 Rx
Command Mode: Command [✓ File] Macro

Figure 13.—File path populated to Space Telecommunications Radio System (STRS) Command File field. IP, Internet Protocol; Rx, receive; Tx, transmit.



Target
Name: IPAS
IP Address: 192.168.1.5
Ports: 1120 Tx, 1130 Rx
Command Mode: Command [✓ File] Macro

Figure 14.—To and Command selections.



Target
Name: IPAS
IP Address: 192.168.1.5
Ports: 1120 Tx, 1130 Rx
Command Mode: [✓ Command] File

STRS Commands
From: FC
To: FC
Command: STRS_Configure
Prop List: <NO PROPS SET>
FC;FC;STRS_Configure

Name:
Value:

Figure 15.—Properties List Editor configuration.

This application allows for control of other STRS implementations that perform many functions beyond what the sample waveform WFIPAS2 is intended to do. This application uses only STRS_Start, STRS_Stop, STRS_Configure, and STRS_Query. The Source, FC; the Destination, WFIPAS2; and the Property, Selected Command; have been selected.

Follow this step to specify the property name and the property value, if the selected property requires it.

(13) For example, to change the property name SOURCETX to STREAMING_PRBS:

(a) Select STRS_Configure from the Command pull-down menu.

(b) Click Prop List. A new window titled “Properties List Editor” opens (Figure 15).

   (c) Enter the name (SOURCETX) and the value (STREAMING_PRBS) in their respective fields. Enter the names and values in all caps exactly as they are spelled in Table I.
   (d) Click Add Change.
   (e) Click OK.
(14) To complete this procedure, click Send to send the command to the SDR.

This procedure can be repeated with all the properties, the property names, and property values in Table I. The GUI’s output windows give the user a series of messages about what is happening in the background as commands are sent over Ethernet and command responses are received from the SDR. The user is encouraged to explore the different options in this GUI to grasp a better understanding of the inner workings of this interface.

4.2 IPASDataIO External to the General Purpose Module (GPM)

The IPASDataIO.exe application is a clone of the IPASDataIO code internal to the GPM. The main difference is that the IPASDataIO.exe application in the external computer had to be run under Cygwin (www.cygwin.com) so that the same commands available on the Linux system can be used in the Windows OS.

To install and run IPASDataIO.exe, follow these steps:

(1) Download and install Cygwin in a PC running a Windows 7 64-bit OS with at least 520 GB of random-access memory (RAM) and 2 GB of read-only memory (ROM). Follow the Cygwin installation steps from the download website (www.cygwin.com). Make sure to include the development packages (gcc, cmake, etc.).
(2) Open the STRS_IPAS_Radio_External_Control.iso using a compatible virtual disk utility and copy the IPASDataIO directory to the local drive.
(3) Open a Cygwin terminal and use the *cd* command to navigate to where the IPASDataIO directory was saved.
(4) Once there, run the *make* command.
(5) After *make* has finished, run the “.out” file.

Make sure to stay in the same directory where PRBS23.bin and cos.bin are located. The relative path of the executing program in the command must be specified to run IPASDataIO. The command will look something like this:

```
/path/to/the/directory/IPASDataIO/x86_64-pc-linux-gnu ./IPASDataIO PRBS23.bin txrx
```

This instance is running in transmit/receive mode for a PRBS23 data stream. The mode of operation and input file will differ based upon specific test configurations.

4.3 Plotting a Captured Sine Wave

When using IPASDataIO, the user is able to capture data coming from the SDR. This data can be either PRBS, which is usually passed through a BERT to test for errors, or it can be a sinusoid wave. The data coming from the SDR is a set of 16-bit words (2 bytes) representing signed integers. This data is read by IPASDataIO and saved to the “saved_data.bin” file when the user presses the S key and then Enter. This data can be plotted to make sure it outputs a sine wave with one caveat. Because of the asynchronicity of the data stream and the pairwise representation of each word (by 2 bytes), it is not known whether the portion of data saved in “saved_data.bin” is representing the words starting with its first or second byte. Therefore, plot the data as is, shifted by 1 byte. One of these plots should look like a sinusoid.

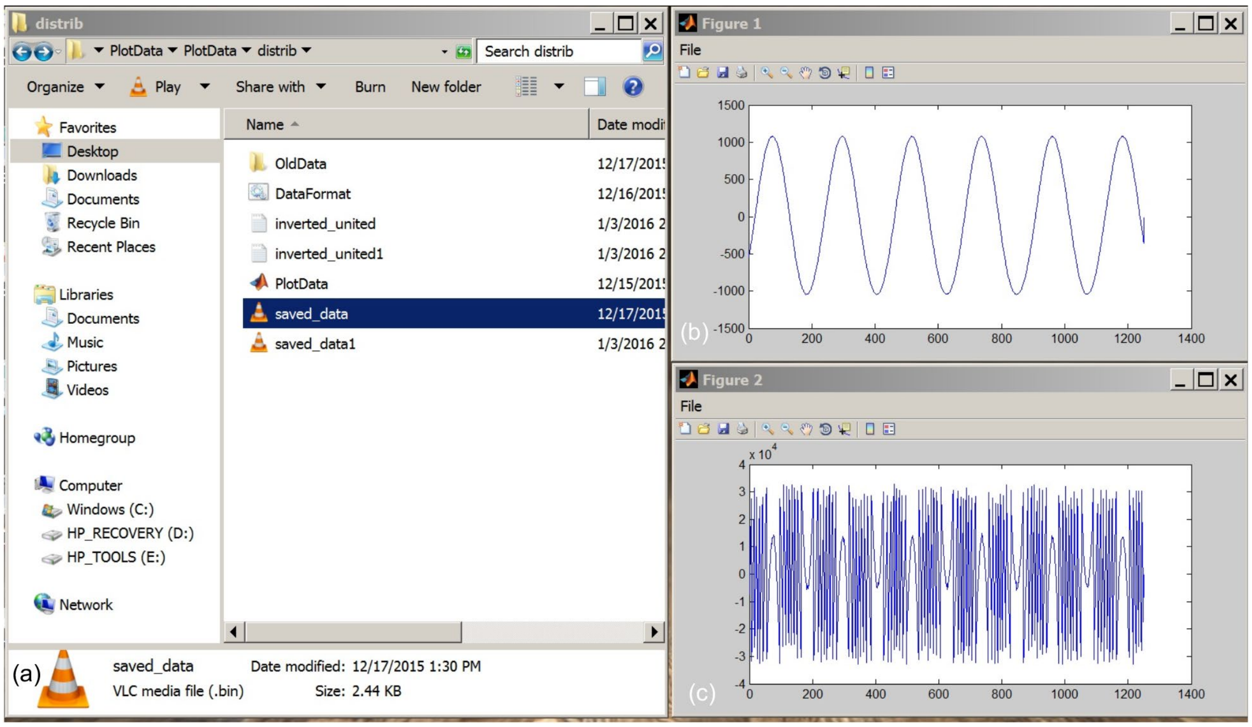
The plotting process is done in two steps. The first step converts the data into a readable format for the plotting tool. The second step inputs those files into the plotting tool to generate the graphs.

The user can copy and paste the formatting and plotting tools from the STRS_IPAS_Radio_External_Control.iso in the External Interface/Plot Data directory. Once these files are saved to the host drive, a data set can be plotted, assuming IPASDataIO has successfully run and the user has saved a “saved_data.bin” file.

To plot the content of the file, follow these steps:

(1) Copy the “saved_data.bin” file from the IPASDataIO directory to the PlotData/PlotData/distrib directory.
(2) Run the DartaFormat.sh script using Cygwin. This will generate three files: “saved_data1.bin,” “inverted_united.txt,” and “inverted_united1.txt.”
(3) Now run the PlotData.exe application. This is a MathWorks® MATLAB® stand-alone app, so the user needs to install the Mathworks® MATLAB® Runtime Compiler (MRC) in order to run this executable. This is a free application from www.mathworks.com.
(4) Once the app has run (it takes about 10 to 40 s to generate the plots), the user can observe something similar to what is shown in Figure 16.

As Figure 16 shows, the top right plot (Figure 16(b)) is a sinusoid, indicating that this instance of data collection is aligned by word to the collection time. Figure 16(c) represents a plot of the data that is shifted by one, which is shown by the uneven plot.



Name	Date modi
OldData	12/17/201
DataFormat	12/16/201
inverted_united	1/3/2016 2
inverted_united1	1/3/2016 2
PlotData	12/15/201
saved_data	12/17/201
saved_data1	1/3/2016 2

saved_data Date modified: 12/17/2015 1:30 PM
VLC media file (.bin) Size: 2.44 KB

Figure 16.—PlotData.exe results. (a) saved_data file selection. (b) Aligned sinusoidal plot. (c) Plot with data shifted by 1 byte.

5.0 Signal Processing Module (SPM)

This section provides a detailed description of how to configure and use the SPM.

5.1 Hardware Identification

This FPGA portion of the iPAS STRS radio design is implemented on a Xilinx® ML605 Rev D evaluation board that contains a Xilinx® Virtex®-6 XC6VLX240T–1FFG1156C FPGA.

5.2 Development Tools

The development tool used for this design is the Xilinx® Integrated Synthesis Environment® (ISE) Design Suite: System Edition, version 14.4, which includes the Embedded Development Kit and Software Development Kit (SDK). The Xilinx® ISE® Simulator (ISim) was used for design simulation of most of the Very High Speed Integrated Circuit (VHSIC) Hardware Description Language (VHDL) modules.

5.3 Xilinx® ML605 Field-Programmable Gate Array (FPGA) Board Configuration—Jumpers and Dual In-Line Package (DIP) Switch Settings

To use the wrapper and test waveform as is, insert jumpers and set DIP switches according to the listings in Table II. Some of the jumpers are in their default locations, and are not needed for the iPAS STRS radio (i.e., Peripheral Component Interconnect Express (PCIe) lane size). If a jumper connector is not listed, no jumper is needed. Table II shows the jumper and switch configurations for the delivered platform.

5.4 CompactFlash (CF)

The Xilinx® ML605 FPGA board contains a system archiver compression file (ACE) CF configuration controller, which is compatible with Type I or II CF cards. The CF configuration controller can be used to program the FPGA with an ACE file on the CF card (inserted in connector U73) during the board power-on cycle. Reprogramming of the FPGA can also be initiated by pushing SYSACE_RESET_B SW3 on the FPGA board.

To use the CF configuration controller to program the FPGA, S4 on DIP switch S1 must be set to On. The switches S1, S2, and S3 on DIP switch S1 select which folder location on the CF card to use for programming the FPGA. The CF card contains eight subfolders, each of which can contain only one ACE file.

5.5 Joint Test Action Group (JTAG) Programming

During waveform development, the Xilinx® Virtex®-6 FPGA on the Xilinx® ML605 FPGA board can be programmed using the Xilinx® iMPACT program, the ISE® Design Suite generated.bit file, a USB (Windows PC) to mini USB cable (Xilinx® ML605 FPGA board side), and the USB JTAG connector (J22) on the board.

5.6 Archiver Compression File (ACE) File

The final iPAS STRS radio FPGA design has been programmed with an ACE file loaded onto the CF configuration controller located at U73. The FPGA device will automatically be programmed from the CF card at power-on. If the radio does not appear to be working properly, reprogram the FPGA by pressing SYSACE_RESET_B SW3.

TABLE II.—JUMPER AND SWITCH CONFIGURATION FOR DELIVERED PLATFORM

Jumper purpose/DIP[a] switch ID[b]	Location	Jumper connections or switch position	Description
Ethernet PHY[c] configuration	J66	Jumper pins 1 and 2	To use GMII[d] (1 Gigabit) interface to Ethernet PHY
	J67	Jumper pins 1 and 2	
	J68	No jumper	
For JTAG[e] access to the board	J17	Jumper pins 1 and 2	Enables JTAG access without FMC[f] modules installed
	J18	Jumper pins 1 and 2	
System ACE[g] error LED[h] disable Jumper	J69	Jumper pins 1 and 2	Enables LED, which will flash if there is an ACE problem
SFP[i] module control	J54	Jumper pins 1 and 2	SFP_RT_SEL (full bandwidth)
	J65	Jumper pins 1 and 2	SFP_TX_DISABLE (SFP enabled)
PCIe[j] lane size	J42	Jumper pins 5 and 6	X8 lane size
System monitor	J19	Jumper pins 1 and 2	Use on-chip reference
	J35	Jumper pins 1 and 3	Not connected
		Jumper pins 2 and 4	FPGA[k] thermal diode access
DIP switch S1	S1	CFGAddr 0—Off	Selects which CF[l] image is downloaded to the FPGA (selects subfolder cfg 4)
	S2	CFGAddr 1—Off	
	S3	CFGAddr 2—On	
	S4	SysAce Mode—On	Enable ACE boot
DIP switch S2	S1	EXT CCLK—On	Oscillator enable
	S2	CS_SEL—On	Boot EPROM[m] select
	S3	M0—Off	FPGA mode (slave SelectMAP)
	S4	M1—On	
	S5	M2—On	
	S6	Flash_A23—Off	Flash address select (lower)

[a]Dual in-line package.
[b]Identification.
[c]Physical layer.
[d]Gigabit media-independent interface.
[e]Joint Test Action Group.
[f]FPGA Mezzanine Card.
[g]Archiver compression file.
[h]Light-emitting diode.
[i]Small form-factor pluggable.
[j]Peripheral Component Interconnect Express.
[k]Field-programmable gate array.
[l]ControlFlash.
[m]Erasable programmable read-only memory.

5.7 Field-Programmable Gate Array (FPGA) Mezzanine Card (FMC) Connector Interface to Radiofrequency (RF) Board

The RF front-end board (AD–FM–COMMS1–EBZ) connects to the FGPA in via the FMC Low Pin Count (LPC) connector (J63). See the Radiofrequency (RF) Front-End Board to Field-Programmable Gate Array (FPGA) Interface through Low Pin Count (LPC) Mezzanine Connector table in the iPAS STRS Radio Hardware Interface Description document (Ref. 1) for details about the FMC connector interface.

5.8 Power Supply

The Xilinx® Virtex®-6 ML605 Evaluation Kit comes with a 12 V alternating current (AC)-to-direct current (DC) power supply. The power supply has a 6-pin plug that mates with the Xilinx® ML605 FPGA board 6-pin Molex Mini-Fit-type connector at J60. The power can be turned on and off using the power On/Off slide SW2 mounted on the Xilinx® ML605 FPGA board. When the power supply is connected and the power On/Off slide SW2 is On, the DS25 LED will be green.

5.9 Viewing the Analog-to-Digital Converter (ADC) and Digital-to-Analog Converter (DAC) Signals Using the Xilinx® ChipScope™ Pro Tool

A ChipScope™ Pro tool Integrated Logic Analyzer (ILA) core was included with the STRS_Waveform module to allow the user to view particular signals when operating the delivered design. The user is able to view the I channel, ADC, and DAC samples as well as the input data for Rx-side streaming packets. Table III shows the viewable signals and the ChipScope™ Pro tool configuration.

To view the signals, open the ChipScope™ Pro tool 64-bit version under ISE® DesignSuite 14.4. Click the Open Cable/Search JTAG Chain symbol in the upper left corner of the ChipScope™ Pro tool window. When the open cable process is finished, click File→Open Project and browse to select ChipScopeProjects→WaveformADCViewer.cpj to open the saved ChipScope™ Pro tool configuration file. Open the Trigger Setup and Bus Plot windows.

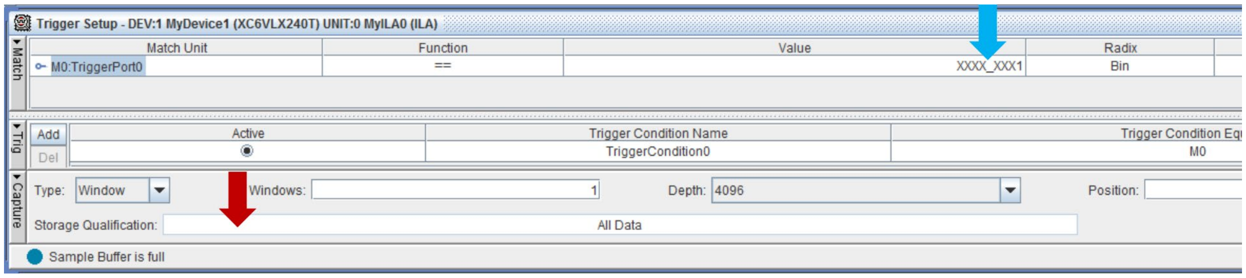
Configure the Trigger Setup according to Table III, based upon the signal required for viewing. Figure 17 shows the Trigger Setup window, where a blue arrow points to the TriggerPort Value and a red arrow points to the Storage Condition block. Edit the TriggerPort0 Value to match the M0:TriggerPort0 column in Table III. To change the Storage Condition, click in the field pointed to by the red arrow, and the window in Figure 18 will open. Edit the radio buttons and check boxes according to the Storage Qualification column in Table III. When the Storage Qualification is edited as shown for the Rx-side streaming data rows in Table III, data is stored only when the RxWFClockEn2 signal is high (i.e., data is stored at the waveform parallel data rate, which is the Serial Data Rate ÷ 16).

Figure 19 to Figure 24 show examples for each of the configurations shown in Table III.

TABLE III.—CHIPSCOPE™ PRO TOOL CONFIGURATION

Signal to view	Rx[a]-side data source	Tx[b]-side data source	M0:TriggerPort0	Storage condition selection	Figure reference
ADC[c] samples, I channel	ADC	Internal sine wave	XXXX_XXX1	● All Data	Figure 19
ADC samples, I channel	ADC	PRBS[d] (either internal or streaming)	XXXX_XXX1	● All Data	Figure 20
DAC[e] samples, I channel	N/A	Internal sine ave	XXXX_XXX1	● All Data	Figure 21
DAC samples, I channel	N/A	PRBS (either internal or streaming)	XXXX_XXX1	● All Data	Figure 22
Rx-side streaming data	ADC	Internal sine wave	XXXX_XXX1	● AND Equation ✓ Enable	Figure 23
Rx-side streaming data	Loopback	Internal sine wave	XXXX_XXX1	● AND Equation ✓ Enable	Figure 24

[a]Receive.
[b]Transmit.
[c]Analog-to-digital converter.
[d]Psuedorandom bit sequence.
[e]Digital-to-analog converter.


Trigger Setup - DEV:1 MyDevice1 (XC6VLX240T) UNIT:0 MyILA0 (ILA)

Match Unit	Function	Value	Radix
M0:TriggerPort0	==	XXXX_XXX1	Bin

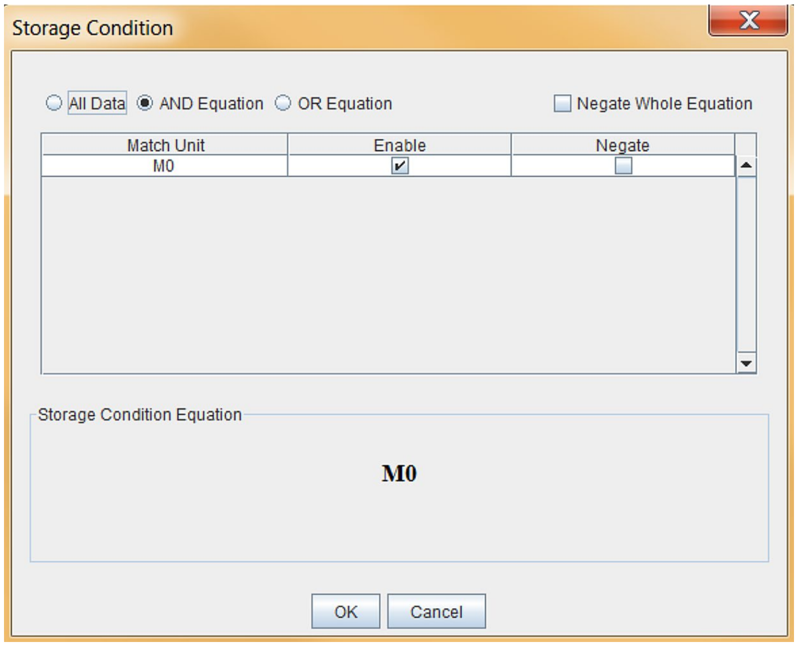
Active	Trigger Condition Name	Trigger Condition Eq
●	TriggerCondition0	M0

Type: Window Windows: 1 Depth: 4096 Position:

Storage Qualification: All Data

Sample Buffer is full

Figure 17.—ChipScope™ Pro tool Trigger Setup.


Storage Condition

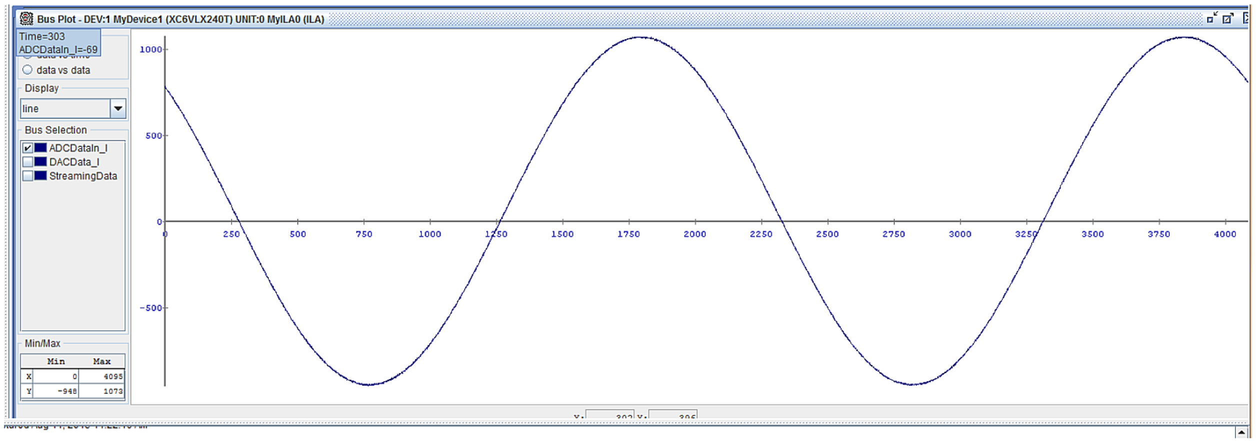
All Data [✓ AND Equation] OR Equation   Negate Whole Equation

Match Unit	Enable	Negate
M0	✓	

Storage Condition Equation

M0

Figure 18.—ChipScope™ Pro tool Storage Condition.


Bus Plot - DEV:1 MyDevice1 (XC6VLX240T) UNIT:0 MyILA0 (ILA)

Time=303
ADCDataIn_I=-69

data vs data

Display: line

Bus Selection: [✓ ADCDataIn_I] DACData_I StreamingData

Min/Max

	Min	Max
X	0	4095
Y	-948	1073

Figure 19.—Sine wave through radiofrequency module (RFM), analog-to-digital converter (ADC) output.

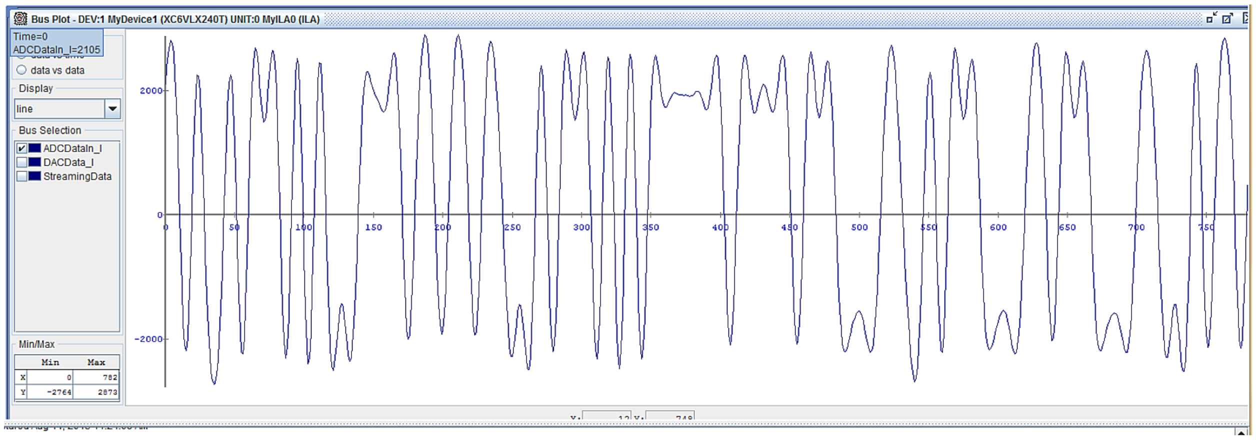


Figure 20.—Modulated pseudorandom bit sequence (PRBS) data through radiofrequency (RF) board; analog-to-digital converter (ADC) output.

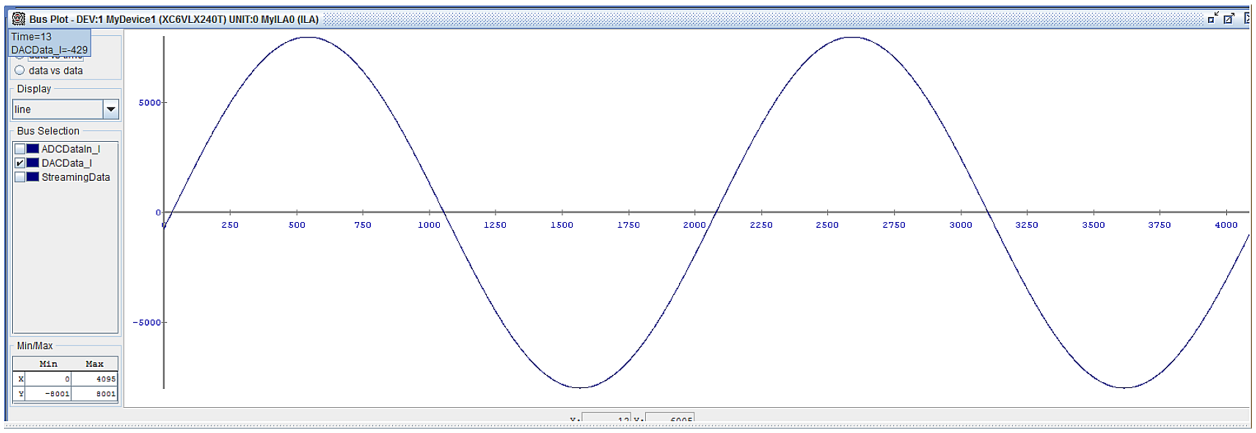


Figure 21.—Sine wave to radiofrequency module (RFM); digital-to-analog converter (DAC) input.

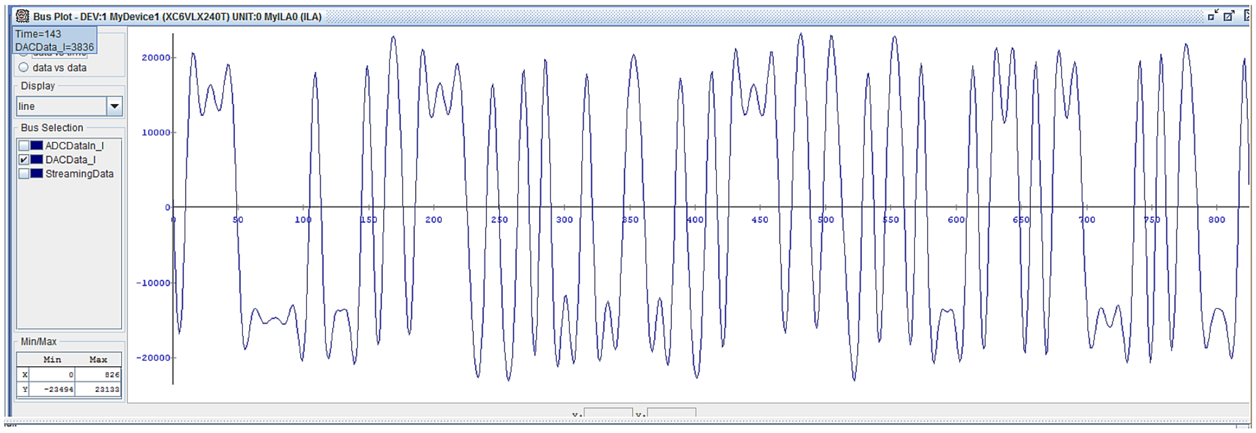


Figure 22.—Modulated pseudorandom bit sequence (PRBS) data to radiofrequency module (RFM); digital-to-analog converter (DAC) input.

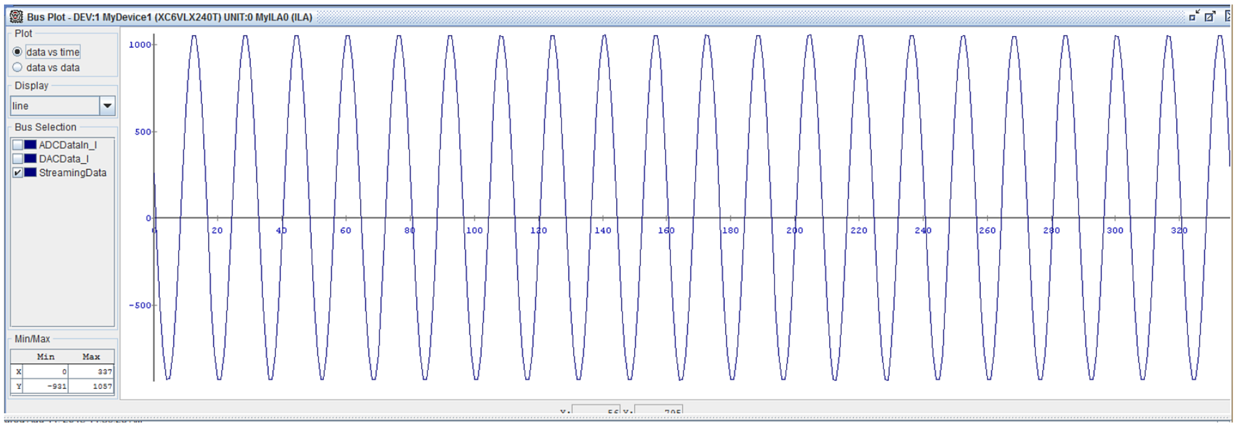


Figure 23.—Receive (Rx)-side streaming data from sine wave through radiofrequency (RF) board.

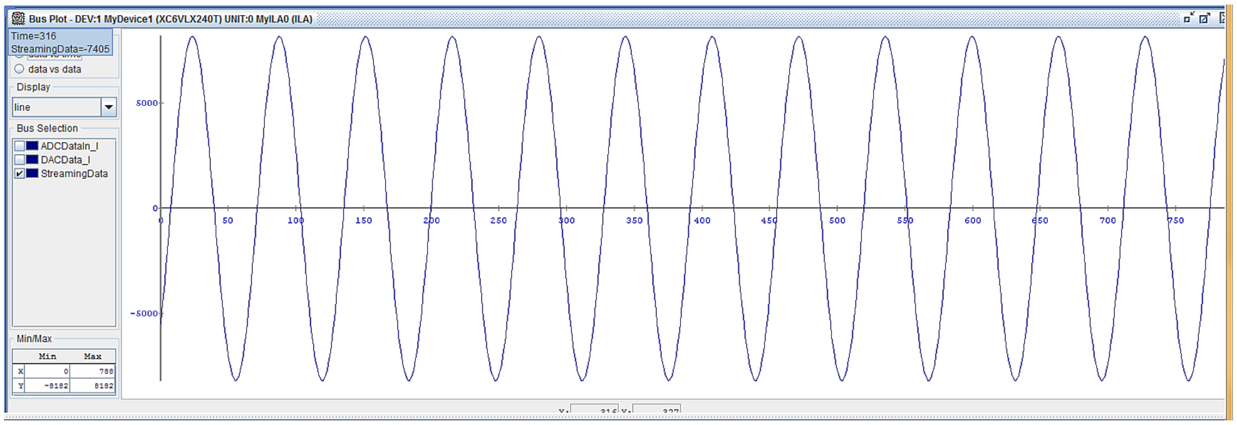


Figure 24.—Receive (Rx)-side streaming data from sine wave using internal loopback.

TABLE IV.—DUAL IN-LINE PACKAGE (DIP) SWITCH CONFIGURATIONS

DIP switch	Position	Result
S1	CFGAddr 0—On	Selects which CF[a] image is downloaded to the FPGA[b] (selects subfolder cfg 7)
S2	CFGAddr 1—On	
S3	CFGAddr 2—On	

[a]ControlFlash.
[b]Field-programmable gate array.

5.10 Non-Return-to-Zero-Mark (NRZ-M) Data Format Option

Demodulation of a binary phase-shift keying (BPSK) signal can sometimes result in a data inversion at the output of the demodulator. To avoid this problem, the serial data in the modulator input can be converted to NRZ-M bit encoding. The iPAS radio has the option of using either NRZ-M or non-return-to-zero-line (NRZ-L) encoding. The type of encoding can be selected using S1 at DIP switch S1 on the Xilinx® ML605 FPGA board. When S1 is set to On, NRZ-M is selected.

5.11 Optional Lower Data Rate Design

A second iPAS STRS radio design was developed with a lower data rate due to the low availability of high rate demodulators for verification of the modulator operation. This optional design generates serial data at a rate of 9.8304 Mbps. The ACE file for this optional design was put in subfolder cfg 7 on the CF card. The user can select this design by changing the DIP switches on DIP switch S1 according to Table IV.

6.0 Radiofrequency Module (RFM)

This section provides a description of how to configure and interface the RFM.

6.1 Hardware Identification

The RFM consists of an RF front-end board (AD–FMCOMMS1–EBZ). The RFM was designed to provide the analog front end for FPGA-based radio applications. Detailed information about the RF front-end board (AD–FMCOMMS1–EBZ) can be found in the Analog Devices wiki at http://wiki.analog.com/resources/eval/user-guides/ad-fmcomms1-ebz/hardware/functional_overview.

The RF front-end board (AD–FMCOMMS1–EBZ) is a mezzanine board compatible with the Xilinx® ML605 Rev D evaluation board that can plug into either FMC connector on the Xilinx® ML605 Rev D evaluation board. The iPAS STRS radio requires the RF front-end board (AD–FMCOMMS1–EBZ) to be inserted in only the FMC LPC connector. The Tx side of the high-speed analog board contains a 16-bit DAC, followed by an up-converter and a 20 dB linear amplifier. The Rx side of the board contains a down-converter, followed by a variable gain amplifier and an ADC. The board also contains clock generators and synchronizers as well as frequency synthesizers needed for the operation of the analog components. The RF front-end board (AD–FMCOMMS1–EBZ) is configured by the FPGA board through an Inter-Integrated Circuit (IIC) interface. The IIC interface is converted to a Serial Peripheral Interface (SPI), which is used to configure the components on the RF front-end board (AD–FMCOMMS1–EBZ).

6.2 Using the Radiofrequency (RF) Board

The RF front-end board (AD–FMCOMMS1–EBZ) provides the analog and RF signal processing for the iPAS STRS radio. On the Tx side, the RF front-end board (AD–FMCOMMS1–EBZ) takes complex I and Q inputs (16 bits) into a high-speed DAC to create an analog signal. A Q modulator up-converts the DAC output signal to the desired RF.

On the Rx side, the received RF signal is demodulated using direct conversion to create I and Q analog signals. The analog signals are converted to digital data using a 14-bit ADC.

The MicroBlaze™ is a soft processor core for the Xilinx® Virtex®-6 FPGA that is used in the iPAS STRS radio design to configure the RF front-end board (AD–FMCOMMS1–EBZ). Analog Devices provides a reference design to help users interface their RF front-end board (AD–FMCOMMS1–EBZ) with the Xilinx® ML605 FPGA board. The reference design is available through their online Wiki (http://wiki.analog.com/resources/eval/user-guides/ad-fmcomms1-ebz). The reference design contains functionality that was not needed for the iPAS STRS radio, so that functionality was removed from the MicroBlaze.xmp Xilinx® Platform Studio portion of the reference design, leaving only the functionality necessary to configure and provide clocking to the RF board and the universal asynchronous receiver/transmitter (UART). This provides the waveform developer with the ability to directly connect to the RF board's DAC and ADC with VHDL.

The SDK portion of the reference design SDK was retained in the iPAS STRS radio. The `main.c` file was edited to configure the ADC sampling rate (200 MHz), the DAC sampling rate (200 MHz), and the Rx RF gain (10,000). The default SDK configuration is provided in the `CompiledDefaultProgram.elf` file.

The MicroBlaze™ processor uses a UART peripheral to display the configuration of the RF front-end board (AD–FMCOMMS1–EBZ). When the MicroBlaze™ processor starts after a power-on or a reset, the UART will send the following information to a terminal—ADC and DAC sampling rates, variable-gain amplifier gain, and RF signal frequency. The UART configuration necessary for the PC-based receiver is as follows: 57,600 baud, 8-bit data, no parity bit, 1 stop bit, and no flow control. Use of the UART requires the installation of a driver on the PC for the Silicon Labs CP210x USB to UART Bridge Virtual COM Port (VCP) driver located on the Xilinx® ML605 FPGA board. The driver is provided with the Xilinx® Virtex®-6 ML605 Evaluation Kit.

The `main.c` program in the SDK files of the reference design contains the assignments to a structure variable called *defInit* (line 70 of `main.c`). The waveform developer can easily customize the reference design by editing this variable. The developer can change the ADC sampling rate, DAC sampling rate, RF gain, and RF signal frequency. The *defInit* variable for the iPAS STRS radio wrapper is configured as follows:

```
XCOMM_DefaultInit defInit = {   FMC_LPC,           //  fmcPort
                                XILINX_ML605,      //  carrierBoard
                                200000000,         //  adcSamplingRate
                                200000000,         //  dacSamplingRate
                                10000,             //  rxGain1000
                                2400000000ull,     //  rxFrequency
                                2400000000ull};    //  txFrequency
```

6.3 Interfaces

The external interfaces of the RFM are described in the Integrated Power, Avionics, and Software (iPAS) Space Telecommunications Radio System (STRS) Radio Hardware Interface Description document (Ref. 1).

7.0 Hardware Initialization Procedure

This section describes the procedure to configure and initialize the hardware.

(1) Connect an Ethernet cable from the eBOX620–110–FL Linux PC to the port marked with a handwritten F on the Xilinx® ML605 FPGA board.
(2) Connect an Ethernet cable between the eBOX620–110–FL Linux PC and Windows PC 1.
(3) Connect a VGA cable between the eBOX620–110–FL Linux PC and the Axiomtek™ monitor.
(4) Verify that the RFM is inserted into the FMC LPC connector on the Xilinx® ML605 FPGA board.
(5) Connect the RF output of the AD–FMCOMMS1–EBZ RF front-end board to the input on the STE down-converter with a SubMiniature version A (SMA) coaxial cable. Make sure that a 20 dB attenuator is inserted at the input of the down convertor.
(6) Connect the Hittite signal generator output to the mixer input on the down-converter (Figure 25). Set signal generator frequency to 2330.00 MHz. The signal generator power level should be set to +7 dBm.
(7) Connect the down-converter output to the Rx Bayonet Neill-Concelman (BNC) connector on the back of the demodulator unit.

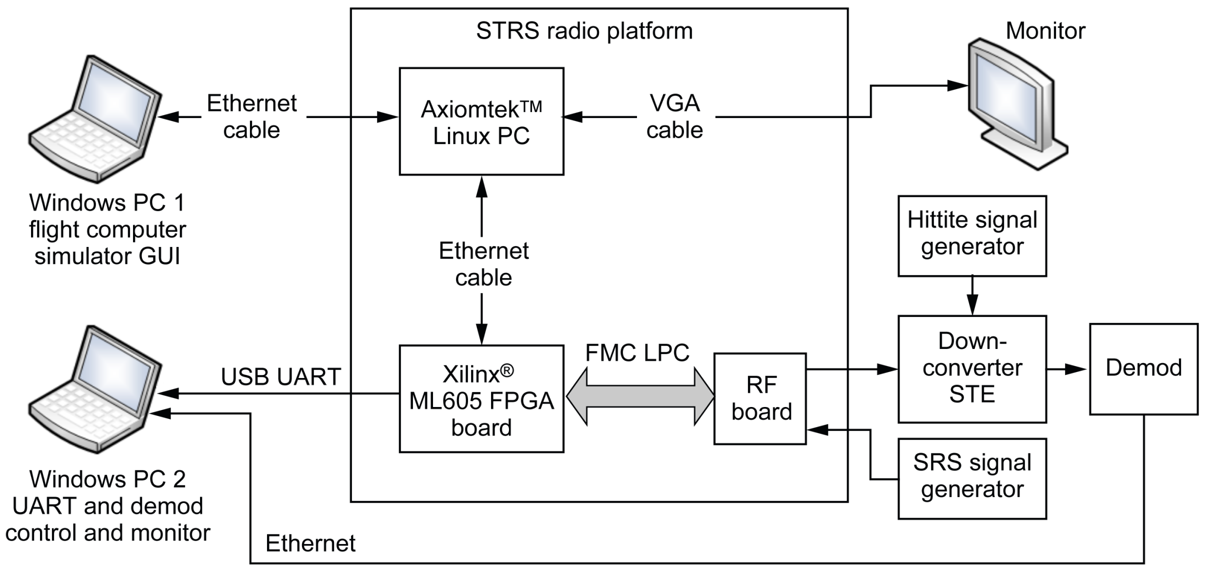


Figure 25.—Hardware setup diagram. Demod, demodulator; FMC, FPGA Mezzanine Card; FPGA, field-programmable gate array; GUI, graphical user interface; LPC, Low Pin Count; PC, personal computer; RF, radiofrequency; SRS, Stanford Research Systems; STRS, Space Telecommunications Radio System; UART, universal asynchronous receiver/transmitter; VGA, Video Graphics Array.

(8) Make sure that there is a 20 dB attenuator connected to the input of the AD–FMCOMMS1–EBZ RF front-end board. Connect the Stanford Research Systems (SRS) signal generator output to the RF input of the AD–FMCOMMS1–EBZ RF front-end board. The SRS signal generator output power should come up to the correct level when powered on, but to be sure, measure the power with a power meter before connecting it to the AD–FMCOMMS1–EBZ RF front-end board. The SRS signal generator output power should be around +7 dBm. If necessary, adjust the output voltage until the output power is correct. Set the frequency to 2399.90 MHz.

(9) Confirm that the programmed CF card is inserted in the Xilinx® ML605 FPGA board connector (U73). The high data rate (24.576 Mbps) FPGA waveform can be selected by setting S3, S2, and S1 to "100" (or On, Off, and Off) on DIP switch S1, in the order given. The low data rate (9.8304 Mbps) FPGA waveform can be selected by setting S3, S2, and S1 to "111" (or On, On, and On) on DIP switch S1.

(10) Check and configure the jumpers and switch settings on the Xilinx® ML605 FPGA board.

(11) If necessary, connect the power cable to the Xilinx® ML605 FPGA board and operate the power On/Off slide SW2 to the On position.

(12) Power on Windows PC 2.

(13) Power on the eBOX620–110–FL Linux PC.

(14) Connect the demodulator to Windows PC 2 with an Ethernet cable. This link contains the manual for the demodulator: http://www.paradisedata.com/files/trnsfr/modems-documentation/evolution/EvolutionSeriesHandbook.pdf.

(15) To connect to the demodulator GUI, open a browser on Windows PC 2 and enter 10.10.10.110 for the IP address.

(16) On the demodulator GUI, select Edit→Memory→Recall.

(17) Click the Recall button next to "iPAS_Radio_High_Rate" or "iPAS_Radio_Low_Rate," depending on the data rate selected in step (9) above. It takes a little while for the configuration to load, so be patient.

(18) To view the current settings, go to View→TxRx. The demodulator is locked onto the received signal when the Receive Traffic light on the left side of the GUI is green.
(19) To view the spectrum, go to View→Graphs→Spectrum.
(20) To view the constellation, go to View→Graphs→Constellation.
(21) To view the BER, go to Test→BERT. The demodulator BERT is locked to the received PRBS data when the light next to the RESTART button is green and the text next to the green light says "Sync OK."
(22) Click the RESTART button to clear the errors in the BER results.
(23) To initialize the WFIPAS2 app on the eBOX620–110–FL Linux PC, run script RunSetup1.
(24) To start the Flight Simulator GUI on the second Windows PC, double-click the FCS shortcut.
(25) Click the "File" radio button and select the WFIPAS.
(26) Select the file "WFIPAS_GRC1.int."

8.0 Conclusions

The Integrated Power, Avionics, and Software (iPAS) Space Telecommunications Radio System (STRS) radio was implemented on the Reconfigurable, Intelligently-Adaptive Communication System (RIACS) platform, currently being used for radio development at NASA Johnson Space Center. The platform consists of a Xilinx® Virtex®-6 ML605 Evaluation Kit, an Analog Devices AD–FMCOMMS1–EBZ radiofrequency (RF) front-end board, and an Axiomtek™ eBOX620–110–FL embedded personal computer (PC) running the Ubuntu® 12.04 LTS operating system (OS). The result of this development is a low-cost, STRS-compliant platform that can be used for waveform development for multiple radio applications. This document provides users with information on how to develop a new waveform using the RIACS platform. Details about the STRS reference implementation and how it interacts with other software components are presented, along with information on how to use the software waveform templates to develop new radio waveforms. Details about the field-programmable gate array (FPGA) platform wrapper are provided to help the user understand how to interface to the platform and the FPGA wrapper.

Appendix—Abbreviations and Acronyms

The following abbreviations and acronyms are used within this document.

AC	alternating current
ACE	archiver compression file
ADC	analog-to-digital converter
API	application programming interface
BER	bit error rate
BERT	bit error rate tester
BNC	Bayonet Neill-Concelman
BPSK	binary phase-shift keying
CF	CompactFlash
DAC	digital-to-analog converter
DC	direct current
demod	demodulator
DIP	dual in-line package
EPROM	erasable programmable read-only memory
FC	flight computer
FCS	flight computer simulator
FIFO	first in first out
FMC	FPGA Mezzanine Card
FPGA	field-programmable gate array
GMII	gigabit media-independent interface
GPM	general purpose module
GPP	general purpose processor
GUI	graphical user interface
HAL	hardware abstraction layer
HDL	hardware description language
I	in-phase
IIC	Inter-Integrated Circuit
ILA	Integrated Logic Analyzer
I/O	input/output
IP	Internet Protocol
iPAS	Integrated Power, Avionics, and Software
ISE®	Integrated Synthesis Environment®
ISim	ISE® Simulator
JTAG	Joint Test Action Group
LAN	local area network
LPC	Low Pin Count
LED	light-emitting diode
MRC	MATLAB® Runtime Compiler
NRZ-L	non-return-to-zero line
NRZ-M	non-return-to-zero mark
OE	operating environment
OS	operating system
PC	personal computer

PCIe	Peripheral Component Interconnect Express
PHY	physical layer
PRBS	pseudorandom bit sequence
Q	quadrature
RIACS	Reconfigurable, Intelligently-Adaptive Communication System
RAM	random-access memory
RF	radiofrequency
RFM	radiofrequency module
ROM	read-only memory
Rx	receive
SDK	Software Development Kit
SDR	software defined radio
SFP	small form-factor pluggable
SMA	SubMiniature version A
SPI	Serial Peripheral Interface
SPM	signal processing module
SRS	Stanford Research Systems
STRS	Space Telecommunications Radio System
SW	switch
TCP	Transmission Control Protocol
Tx	transmit
UART	universal asynchronous receiver/transmitter
UDP	User Datagram Protocol
USB	Universal Serial Bus
VCP	Virtual COM Port
VGA	Video Graphics Array
VHDL	VHSIC Hardware Description Language
VHSIC	Very High Speed Integrated Circuit
WFIPAS2	iPAS sample waveform

References

1. Shalkhauser, Mary Jo W.; and Roche, Rigoberto: Hardware Interface Description for the iPAS STRS Radio. NASA/TM—2017-219432, 2017.